Media Player Framework Reference

Audio & Video

2010-12-09

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

Contents

Tables and Listings

Introduction

Framework	/System/Library/Frameworks/MediaPlayer.framework
Header file directories	/System/Library/Frameworks/MediaPlayer.framework/Headers
Declared in	MPMediaEntity.h MPMediaItem.h MPMediaItemCollection.h MPMediaLibrary.h MPMediaPickerController.h MPMediaPlayback.h MPMediaPlaylist.h MPMediaQuery.h MPMediaQuerySection.h MPMoviePlayerController.h MPMoviePlayerViewController.h MPMusicPlayerController.h MPVolumeSettings.h MPVolumeView.h

The Media Player framework provides facilities for playing movie, music, audio podcast, and audio book files. This framework also gives your application access to the iPod library, letting you find and play audio-based media items synced from iTunes on the desktop. iPod library access is read-only.

This framework's `MPVolumeView` class lets you present a control to let the user adjust system audio output volume and choose among the available output routes, such as for sending audio to an AirPlay-enabled device. The `MPMoviePlayerController` and `MPTimedMetadata` classes let you play streamed video content and respond to time-based metadata contained in the stream. Starting in iOS 4.3, you can configure a movie player to support wireless movie playback to AirPlay-enabled hardware.

Introduction

Classes

PART I

Classes

MPMediaEntity Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 4.2 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide

Overview

The `MPMediaEntity` class serves as the abstract superclass for `MPMediaItem` and `MPMediaItemCollection` instances, and in turn for `MPMediaPlaylist` instances. As the superclass, `MPMediaEntity` defines methods used by those subclasses.

Tasks

Working with Media Properties

`+ canFilterByProperty:` (page 14)
Indicates if a media property key can be used to construct a media property predicate.

`- valueForProperty:` (page 15)
Gets the value for a specified media property key.

`- enumerateValuesForProperties:usingBlock:` (page 14)
Executes a provided block with the fetched values for the given item properties.

Class Methods

canFilterByProperty:

Indicates if a media property key can be used to construct a media property predicate.

```
+ (BOOL)canFilterByProperty:(NSString *)property
```

Parameters

property

The key for the media property that you want to examine.

Return Value

`YES` if the property you are testing can be used to construct a media property predicate (of type `MPMediaPropertyPredicate`); otherwise, `NO`.

Discussion

The media property keys you can use with this property are listed in this document and in `General Media Item Property Keys` (page 20), `Podcast Item Property Keys` (page 26), `Playlist Property Keys` (page 44), and `User-Defined Property Keys` (page 26).

Availability

Available in iOS 4.2 and later.

Declared In

`MPMediaEntity.h`

Instance Methods

enumerateValuesForProperties:usingBlock:

Executes a provided block with the fetched values for the given item properties.

```
- (void) enumerateValuesForProperties: (NSSet *) properties usingBlock: (void
    (^)(NSString *property, id value, BOOL *stop)) block;
```

Parameters

properties

A set of property keys that you want the values for.

block

A block object that executes for each fetched property value. If a value is not available, your block is sent `nil`.

Discussion

Use this method to get property values in a batch fashion. In some cases, enumerating over a set of property keys can be more efficient than fetching each individual property with `valueForProperty:` (page 15).

The media property keys you can use with this property are listed in this document and in `General Media Item Property Keys` (page 20), `Podcast Item Property Keys` (page 26), `Playlist Property Keys` (page 44), and `User-Defined Property Keys` (page 26).

Availability
Available in iOS 4.2 and later.

Declared In
`MPMediaEntity.h`

valueForProperty:

Gets the value for a specified media property key.

```
- (id) valueForProperty: (NSString *) property
```

Parameters

property

The media property key that you want the corresponding value of.

Return Value
The value for the media *property* key.

Discussion
The media property keys you can use with this property are listed in this document and in `General Media Item Property Keys` (page 20), `Podcast Item Property Keys` (page 26), `Playlist Property Keys` (page 44), and `User-Defined Property Keys` (page 26).

Availability
Available in iOS 4.2 and later.

Declared In
`MPMediaEntity.h`

Constants

Media Entity Property Keys

You obtain metadata for a media entity by calling the `valueForProperty` method with these property keys. Some properties can also be used to build media property predicates, as described in *MPMediaPropertyPredicate Class Reference*. These properties are marked here as *filterable*.

```
NSString *const MPMediaEntityPropertyPersistentID;     // filterable
```

Constants

`MPMediaEntityPropertyPersistentID`

The persistent identifier for a media entity. Value is an NSNumber object containing a `uint64_t` (unsigned long long).

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 4.2 and later.

Declared in `MPMediaEntity.h`.

MPMediaItem Class Reference

Inherits from	MPMediaEntity : NSObject
Conforms to	NSCoding (MPMediaEntity) NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	MPMediaItem.h
Companion guide	iPod Library Access Programming Guide
Related sample code	AddMusic

Overview

A media item represents a single piece of media (such as one song) in the iPod library. A media item has an overall unique identifier, accessed using the `MPMediaItemPropertyPersistentID` (page 21) property key, as well as specific identifiers for its metadata. These identifiers persists across application launches.

A media item can have a wide range of metadata associated with it. You access this metadata using the `valueForProperty:` (page 15) method along with the property keys described in this document. The You can also access metadata in a batch fashion using the `enumerateValuesForProperties:usingBlock:` (page 14) method. In some cases, this is more efficient. Both of these methods are defined in `MPMediaEntity`, the abstract superclass of `MPMediaItemCollection`, and described in *MPMediaEntity Class Reference*.

You use attributes of media items to build media queries for searching the iPod library. These attributes are described in "Media Item Type Flags" (page 19), "General Media Item Property Keys" (page 20), and "Podcast Item Property Keys" (page 26). In addition, the `MPMediaEntityPropertyPersistentID` (page 15) property is described in `Media Entity Property Keys` (page 15). Media queries are described in *MPMediaQuery Class Reference*.

Tasks

Obtaining Group Properties

`+ persistentIDPropertyForGroupingType:` (page 18)
Obtains the persistent identifier key for a specified grouping type.

`+ titlePropertyForGroupingType:` (page 18)
Obtains the title key for a specified grouping type.

Class Methods

persistentIDPropertyForGroupingType:

Obtains the persistent identifier key for a specified grouping type.

```
+ (NSString *) persistentIDPropertyForGroupingType: (MPMediaGrouping) groupingType;
```

Parameters

groupingType
The grouping type that you want the persistent identifier key for.

Discussion
Use this convenience method to obtain the key for a specific persistent identifier based on a grouping type. You can use that key, in turn, to obtain the value of a specific persistent ID of a media item, such as album title or artist name. Using this method simplifies such tasks as drilling down from an artist, to albums by that artist, to a specific album.

For example, the following statement returns the persistent identifier key for the album grouping type:

```
NSString *albumIDKey = [MPMediaItem persistentIDPropertyForGroupingType:
MPMediaGroupingAlbum];
```

You could then obtain the specific persistent ID that you want by using the `valueForProperty:` (page 15)method. Grouping keys are described in `Media Item Collection Grouping Keys` (page 63).

Availability
Available in iOS 4.2 and later.

Declared In
`MPMediaQuery.h`

titlePropertyForGroupingType:

Obtains the title key for a specified grouping type.

```
+ (NSString * )titlePropertyForGroupingType: (MPMediaGrouping) groupingType;
```

Parameters

groupingType

The grouping type that you want the title key for.

Discussion

Use this convenience method to obtain the key for the title that corresponds to a specified grouping type. For example, the following statement obtains the title key for the album grouping type:

```
NSString *titleIDKey = [MPMediaItem titlePropertyForGroupingType:
MPMediaGroupingAlbum];
```

You could then obtain the specific title that you want by using the `valueForProperty:` (page 15) method. Grouping keys are described in `Media Item Collection Grouping Keys` (page 63).

Availability

Available in iOS 4.2 and later.

Declared In

`MPMediaQuery.h`

Constants

Media Item Type Flags

Media item types, used as possible values for the `MPMediaItemPropertyMediaType` (page 23) property. A media item can have more than one media item type.

```
enum {
    // audio media types
    MPMediaTypeMusic        = 1 << 0,
    MPMediaTypePodcast      = 1 << 1,
    MPMediaTypeAudioBook    = 1 << 2,
    MPMediaTypeAnyAudio     = 0x00ff,


    // generic media type
    MPMediaTypeAny          = ~0
};
typedef NSInteger MPMediaType;
```

Constants

`MPMediaTypeMusic`

If set, the media item contains music.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaTypePodcast`

If set, the media item contains a podcast.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaTypeAudioBook`
: If set, the media item contains an audio book.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaTypeAnyAudio`
: If set, the media item contains an unspecified type of audio content.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaTypeAny`
: If set, the media item contains an unspecified type of audio.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

General Media Item Property Keys

You obtain metadata for a media item by calling the `valueForProperty` method with these property keys. Some properties can also be used to build media property predicates, as described in *MPMediaPropertyPredicate Class Reference*. These properties are marked here as *filterable*.

```
NSString *const MPMediaItemPropertyPersistentID;            // filterable
NSString *const MPMediaItemPropertyAlbumPersistentID;       // filterable
NSString *const MPMediaItemPropertyArtistPersistentID;      // filterable
NSString *const MPMediaItemPropertyAlbumArtistPersistentID; // filterable
NSString *const MPMediaItemPropertyGenrePersistentID;       // filterable
NSString *const MPMediaItemPropertyComposerPersistentID;    // filterable
NSString *const MPMediaItemPropertyPodcastPersistentID;     // filterable
NSString *const MPMediaItemPropertyMediaType;               // filterable
NSString *const MPMediaItemPropertyTitle;                   // filterable
NSString *const MPMediaItemPropertyAlbumTitle;              // filterable
NSString *const MPMediaItemPropertyArtist;                  // filterable
NSString *const MPMediaItemPropertyAlbumArtist;             // filterable
NSString *const MPMediaItemPropertyGenre;                   // filterable
NSString *const MPMediaItemPropertyComposer;                // filterable
NSString *const MPMediaItemPropertyPlaybackDuration;
NSString *const MPMediaItemPropertyAlbumTrackNumber;
NSString *const MPMediaItemPropertyAlbumTrackCount;
NSString *const MPMediaItemPropertyDiscNumber;
NSString *const MPMediaItemPropertyDiscCount;
NSString *const MPMediaItemPropertyArtwork;
NSString *const MPMediaItemPropertyLyrics;
NSString *const MPMediaItemPropertyIsCompilation;           // filterable
NSString *const MPMediaItemPropertyReleaseDate;
NSString *const MPMediaItemPropertyBeatsPerMinute;
NSString *const MPMediaItemPropertyComments;
NSString *const MPMediaItemPropertyAssetURL;
```

Constants

`MPMediaItemPropertyPersistentID`

The persistent identifier for the media item. Value is an `NSNumber` object containing a `uint64_t` (unsigned long long).

The value of the `MPMediaItemPropertyPersistentID` identifier persists across application launches and across syncs that do not change the sync status of the media item. The value is not guaranteed to persist across a sync/unsync/sync cycle.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyAlbumPersistentID`

The persistent identifier for an album. Value is an `NSNumber` object containing a `uint64_t` (unsigned long long).

The value of the `MPMediaItemPropertyAlbumPersistentID` identifier persists across application launches and across syncs that do not change the sync status of the media item. The value is not guaranteed to persist across a sync/unsync/sync cycle.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 4.2 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyArtistPersistentID`

The persistent identifier for an artist. Value is an `NSNumber` object containing a `uint64_t` (unsigned long long).

The value of the `MPMediaItemPropertyArtistPersistentID` identifier persists across application launches and across syncs that do not change the sync status of the media item. The value is not guaranteed to persist across a sync/unsync/sync cycle.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 4.2 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyAlbumArtistPersistentID`

The persistent identifier for an album artist. Value is an `NSNumber` object containing a `uint64_t` (unsigned long long).

The value of the `MPMediaItemPropertyAlbumArtistPersistentID` identifier persists across application launches and across syncs that do not change the sync status of the media item. The value is not guaranteed to persist across a sync/unsync/sync cycle.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 4.2 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyGenrePersistentID`

The persistent identifier for a genre. Value is an `NSNumber` object containing a `uint64_t` (unsigned long long).

The value of the `MPMediaItemPropertyGenrePersistentID` identifier persists across application launches and across syncs that do not change the sync status of the media item. The value is not guaranteed to persist across a sync/unsync/sync cycle.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 4.2 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyComposerPersistentID`

The persistent identifier for a composer. Value is an `NSNumber` object containing a `uint64_t` (unsigned long long).

The value of the `MPMediaItemPropertyComposerPersistentID` identifier persists across application launches and across syncs that do not change the sync status of the media item. The value is not guaranteed to persist across a sync/unsync/sync cycle.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 4.2 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyPodcastPersistentID`

The persistent identifier for an audio podcast. Value is an `NSNumber` object containing a `uint64_t` (unsigned long long).

The value of the `MPMediaItemPropertyPodcastPersistentID` identifier persists across application launches and across syncs that do not change the sync status of the media item. The value is not guaranteed to persist across a sync/unsync/sync cycle.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 4.2 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyMediaType`

The media type of the media item. Value is an an `NSNumber` object representing an `NSInteger` data type. The `NSInteger` value represents a bit field flag, or set of flags, from "Media Item Type Flags" (page 19).

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyTitle`

The title (or name) of the media item. This property is unrelated to the `MPMediaItemPropertyAlbumTitle` (page 23) property. Value is an `NSString` object.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyAlbumTitle`

The title of an album, such as "Live On Mars", as opposed to the title of an individual song on the album, such as "Crater Dance (radio edit)" (which you specify using the `MPMediaItemPropertyTitle` (page 23) property). Value is an `NSString` object.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyArtist`

The performing artist(s) for a media item—which may vary from the primary artist for the album that a media item belongs to. For example, if the album artist is "Joseph Fable," the artist for one of the songs in the album may be "Joseph Fable featuring Thomas Smithson". Value is an `NSString` object.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyAlbumArtist`
: The primary performing artist for an album as a whole. Value is an `NSString` object.

  Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyGenre`
: The musical or film genre of the media item. Value is an `NSString` object.

  Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyComposer`
: The musical composer for the media item. Value is an `NSString` object.

  Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyPlaybackDuration`
: The playback duration of the media item. Value is an `NSNumber` object representing a duration in seconds as an `NSTimeInterval`.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyAlbumTrackNumber`
: The track number of the media item, for a media item that is part of an album. Value is an `NSNumber` object representing an `NSUInteger` data type.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyAlbumTrackCount`
: The number of tracks in the album that contains the media item. Value is an `NSNumber` object representing an `NSUInteger` data type.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyDiscNumber`
: The disc number of the media item, for a media item that is part of a multi-disc album. Value is an `NSNumber` object representing an `NSUInteger` data type.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyDiscCount`
: The number of discs in the album that contains the media item. Value is an `NSNumber` object representing an `NSUInteger` data type.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyArtwork`
: The artwork image for the media item. Value is a media item image, described in *MPMediaItemArtwork Class Reference*.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyLyrics`
: The lyrics for the media item. Value is an `NSString` object.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyIsCompilation`
: A Boolean value indicating whether the media item is part of a compilation (`YES`), or not (`NO`). Corresponds to the the "Part of a compilation" checkbox in the Info tab in the Get Info dialog in iTunes. Value is an `NSNumber` object representing a `BOOL` data type.

  Available in iOS 3.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyReleaseDate`
: The date on which the media item was first publicly released. Value is an `NSDate` object.

  Available in iOS 4.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyBeatsPerMinute`
: The number of musical beats per minute for the media item, corresponding to the "BPM" field in the Info tab in the Get Info dialog in iTunes. Value is an `NSNumber` object representing an `NSUInteger` data type.

  Available in iOS 4.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyComments`
: Textual information about the media item, corresponding to the "Comments" field in in the Info tab in the Get Info dialog in iTunes. Value is an `NSString` object.

  Available in iOS 4.0 and later.

  Declared in `MPMediaItem.h`.

`MPMediaItemPropertyAssetURL`
: A URL pointing to the media item, from which an `AVAsset` object (or other URL-based AV Foundation object) can be created, with any options as desired. Value is an `NSURL` object.

  The URL has the custom scheme of `ipod-library`. For example, a URL might look like this:

  ```
  ipod-library://item/item.m4a?id=12345
  ```

  Usage of the URL outside of the AV Foundation framework is not supported.

  Available in iOS 4.0 and later.

  Declared in `MPMediaItem.h`.

Podcast Item Property Keys

You obtain metadata for a podcast media item by calling the `valueForProperty` method with these property keys. So-called *filterable* properties can also be used to build media property predicates, as described in *MPMediaPropertyPredicate Class Reference*.

```
NSString *const MPMediaItemPropertyPodcastTitle;     // filterable
```

Constants

`MPMediaItemPropertyPodcastTitle`

The title of a podcast, such as "This Martian Drudgery", as opposed to the title of an individual episode of a podcast such as "Episode 12: Another Cold Day At The Pole" (which you specify using the `MPMediaItemPropertyTitle` (page 23) property). Value is an `NSString` object.

Can be used to build a media property predicate as described in *MPMediaPropertyPredicate Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

User-Defined Property Keys

You obtain user-defined metadata for a media item by calling the `valueForProperty:` (page 15) method with these property keys. User-defined properties cannot be used to build media property predicates.

```
NSString *const MPMediaItemPropertyPlayCount;
NSString *const MPMediaItemPropertySkipCount;
NSString *const MPMediaItemPropertyRating;
NSString *const MPMediaItemPropertyLastPlayedDate;
NSString *const MPMediaItemPropertyUserGrouping;
```

Constants

`MPMediaItemPropertyPlayCount`

The number of times the user has played the media item. Value is an `NSNumber` object representing an `NSUInteger` data type.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertySkipCount`

The number of times the user has skipped playing the item. Value is an `NSNumber` object representing an `NSUInteger` data type.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyRating`

The user-specified rating of the object in the range `[0...5]`, where a value of `5` indicates the most favorable rating. Value is an `NSNumber` object representing an `NSUInteger` data type.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyLastPlayedDate`

The most recent calendar date on which the user played the media item. Value is an `NSDate` object.

Available in iOS 3.0 and later.

Declared in `MPMediaItem.h`.

`MPMediaItemPropertyUserGrouping`

Corresponds to the “Grouping” field in the Info tab in the Get Info dialog in iTunes. Value is an `NSString` object.

Available in iOS 4.0 and later.

Declared in `MPMediaItem.h`.

MPMediaItemArtwork Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide
Related sample code	AddMusic

Overview

An `MPMediaItemArtwork` object, or media item artwork, represents a graphical image, such as music album cover art, associated with a media item. Media items are described in *MPMediaItem Class Reference*.

Tasks

Using a Media Item Image

`- imageWithSize:` (page 30)
Creates and returns a `UIImage` object of a specified size.

`bounds` (page 30) *property*
The overall bounds, in points, of the image associated with the media item artwork.

`imageCropRect` (page 30) *property*
The bounds, in points, of the content area for the full size image associated with the media item artwork.

Properties

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

bounds

The overall bounds, in points, of the image associated with the media item artwork.

```
@property (nonatomic, readonly) CGRect bounds;
```

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaItem.h`

imageCropRect

The bounds, in points, of the content area for the full size image associated with the media item artwork.

```
@property (nonatomic, readonly) CGRect imageCropRect;
```

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaItem.h`

Instance Methods

imageWithSize:

Creates and returns a `UIImage` object of a specified size.

```
- (UIImage *)imageWithSize:(CGSize)size
```

Parameters

size

The size, in points, for the new `UIImage` object.

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
`MPMediaItem.h`

MPMediaItemCollection Class Reference

Inherits from	MPMediaEntity : NSObject
Conforms to	NSCoding (MPMediaEntity) NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide
Related sample code	AddMusic

Overview

A media item collection is a sorted set of media items (instances of the `MPMediaItem` class) from the iPod library. Typically, you use this class by requesting an array of collections from a media query by way of its `collections` property. Media queries are described in *MPMediaQuery Class Reference*.

The grouping type for the media query determines the arrangement of the media items you obtain. You also use the media query `collections` property to obtain synced playlists, as described in *MPMediaPlaylist Class Reference*.

A media item collection can have a wide range of metadata associated with it. You access this metadata using the `valueForProperty:` (page 15) method along with the property keys described in this document. The You can also access metadata in a batch fashion using the `enumerateValuesForProperties:usingBlock:` (page 14) method. In some cases, this is more efficient. Both of these methods are defined in `MPMediaEntity` (the abstract superclass of `MPMediaItemCollection`) and described in *MPMediaEntity Class Reference*.

Tasks

Creating a Media Item Collection

`+ collectionWithItems:` (page 33)
Creates a media item collection by copying an array of media items.

- initWithItems: (page 34)
  Initializes a media item collection with an array of media items.

Using a Media Item Collection

items (page 32) *property*
  The media items in a media item collection.

representativeItem (page 33) *property*
  A media item whose properties are representative of the other media items in a collection.

count (page 32) *property*
  The number of media items in a collection.

mediaTypes (page 33) *property*
  The types of the media items in a collection.

Properties

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

count

The number of media items in a collection.

```
@property (nonatomic, readonly) NSUInteger count;
```

Discussion
In some cases, using this property is more efficient than fetching the *items* array and asking for the count.

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
MPMediaItemCollection.h

items

The media items in a media item collection.

```
@property (nonatomic, readonly) NSArray *items;
```

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
`MPMediaItemCollection.h`

mediaTypes

The types of the media items in a collection.

```
@property (nonatomic, readonly) MPMediaType mediaTypes;
```

Discussion
The media item types are listed in the `Media Item Type Flags` (page 19) enumeration in *MPMediaItem Class Reference*.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaItemCollection.h`

representativeItem

A media item whose properties are representative of the other media items in a collection.

```
@property (nonatomic, readonly) MPMediaItem *representativeItem;
```

Discussion
The media items in a collection typically share common property values, owing to how the collection was built. For example, if you build a collection based on a predicate that uses the `MPMediaItemPropertyArtist` property, all items in the collection share the same artist name. You can use the `representativeItem` property to efficiently obtain values for such common properties—often more efficiently than fetching an item from the *`items`* array.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaItemCollection.h`

Class Methods

collectionWithItems:

Creates a media item collection by copying an array of media items.

```
+ (MPMediaItemCollection *) collectionWithItems: (NSArray *) items;
```

Parameters

items

The array of media items you are assigning to the media item collection.

Return Value
A media item collection.

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
`MPMediaItemCollection.h`

Instance Methods

initWithItems:

Initializes a media item collection with an array of media items.

```
- (id) initWithItems: (NSArray *) items;
```

Parameters

items

The array of items you are assigning to the media item collection.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaItemCollection.h`

MPMediaLibrary Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide

Overview

An `MPMediaLibrary` object, or media library, represents the state of the set of synced media items (such as songs) on a device. The complete library of media items itself is called the *iPod library*.

A user may sync their device, changing the content of the iPod library, while your application is running. You can use the notification provided by this class to ensure that your application's cache of the iPod library is up-to-date.

To retrieve media items from the iPod library, build a custom query as described in *MPMediaPropertyPredicate Class Reference* and *MPMediaQuery Class Reference*.

Tasks

Using the Default Media Library

`+ defaultMediaLibrary` (page 36)
Gets an instance of the default media library.

`lastModifiedDate` (page 36) *property*
The calendar date on which a media library was last modified.

`- beginGeneratingLibraryChangeNotifications` (page 36)
Asks a media library to turn on notifications for device-to-computer synchronizations.

`- endGeneratingLibraryChangeNotifications` (page 37)
Asks a media library to turn off notifications for device-to-computer synchronizations.

Properties

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

lastModifiedDate

The calendar date on which a media library was last modified.

```
@property (nonatomic, readonly) NSDate *lastModifiedDate;
```

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaLibrary.h`

Class Methods

defaultMediaLibrary

Gets an instance of the default media library.

```
+ (MPMediaLibrary *)defaultMediaLibrary
```

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaLibrary.h`

Instance Methods

beginGeneratingLibraryChangeNotifications

Asks a media library to turn on notifications for device-to-computer synchronizations.

```
- (void)beginGeneratingLibraryChangeNotifications
```

Discussion
This method is nestable—that is, you can call it multiple times. To turn off notifications, you must call `endGeneratingLibraryChangeNotifications` (page 37) the same number of times that you called `beginGeneratingLibraryChangeNotifications`.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaLibrary.h`

endGeneratingLibraryChangeNotifications

Asks a media library to turn off notifications for device-to-computer synchronizations.

```
- (void)endGeneratingLibraryChangeNotifications
```

Availability
Available in iOS 3.0 and later.

See Also
`- beginGeneratingLibraryChangeNotifications` (page 36)

Declared In
`MPMediaLibrary.h`

Notifications

MPMediaLibraryDidChangeNotification

When `MPMediaLibraryDidChangeNotification` is posted, your application should reevaluate items or playlists that you previously cached.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaLibrary.h`

MPMediaPickerController Class Reference

Inherits from	UIViewController : UIResponder : NSObject
Conforms to	NSCoding (UIViewController) NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide
Related sample code	AddMusic

Overview

An `MPMediaPickerController` object, or media item picker, is a specialized view controller that you employ to provide a graphical interface for selecting media items. To display a media item picker, present it modally on an existing view controller. Media items are described in *MPMediaItem Class Reference*.

To respond to user selections and to dismiss a media item picker, use the `MPMediaPickerControllerDelegate` protocol as described in *MPMediaPickerControllerDelegate Protocol Reference*.

Notes: The `MPMediaPickerController` class supports portrait mode only. This class does support subclassing. The view hierarchy for this class is private; do not modify the view hierarchy.

Tasks

Initializing a Media Item Picker

- `init` (page 41)
  Initializes a media item picker for all media types.
- `initWithMediaTypes:` (page 42)
  Initializes a media item picker for specified media types.

Using a Media Item Picker

`allowsPickingMultipleItems` (page 40) *property*
A Boolean value specifying multiple (`YES`) or single (`NO`) selection behavior for a media item picker.

`delegate` (page 40) *property*
The delegate for a media item picker.

`mediaTypes` (page 41) *property*
The media types that media item picker presents.

`prompt` (page 41) *property*
A prompt, for the user, that appears above the navigation bar buttons.

Properties

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

allowsPickingMultipleItems

A Boolean value specifying multiple (`YES`) or single (`NO`) selection behavior for a media item picker.

```
@property (nonatomic) BOOL allowsPickingMultipleItems;
```

Discussion
The default behavior for a media item picker is `NO`, which means that the picker allows selection of only a single media item. In this instance, the button for dismissing the picker is labeled "Cancel."

When using the multiple-selection version, the button for dismissing the picker is labeled "Done."

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
`MPMediaPickerController.h`

delegate

The delegate for a media item picker.

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

Discussion
Typically, you set the delegate to be the same object that initializes and displays the media item picker. The delegate protocol is described in *MPMediaPickerControllerDelegate Protocol Reference*.

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
`MPMediaPickerController.h`

mediaTypes

The media types that media item picker presents.

```
@property (nonatomic, readonly) MPMediaType mediaTypes;
```

Discussion
The available media types are listed in the `Media Item Type Flags` (page 19) enumeration.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaPickerController.h`

prompt

A prompt, for the user, that appears above the navigation bar buttons.

```
@property (nonatomic, copy) NSString *prompt;
```

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
`MPMediaPickerController.h`

Instance Methods

init

Initializes a media item picker for all media types.

```
- (id) init;
```

Discussion
The default media type for a media item picker is `MPMediaTypeAny` (page 20).

Availability
Available in iOS 3.0 and later.

See Also
`- initWithMediaTypes:` (page 42)

Declared In
`MPMediaPickerController.h`

initWithMediaTypes:

Initializes a media item picker for specified media types.

```
- (id) initWithMediaTypes: (MPMediaType) mediaTypes;
```

Parameters

mediaTypes

An integer representing the media types for the media item picker. See the `Media Item Type Flags` (page 19) enumeration.

Availability
Available in iOS 3.0 and later.

See Also
`- init` (page 41)

Related Sample Code
AddMusic

Declared In
`MPMediaPickerController.h`

MPMediaPlaylist Class Reference

Inherits from	MPMediaItemCollection : MPMediaEntity : NSObject
Conforms to	NSCoding (MPMediaEntity) NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide

Overview

A media playlist is a playable collection of related media items. (Media items are described in *MPMediaItem Class Reference*.) Each playlist has a name, a set of attributes, and a unique identifier that persists across application launches.

Users configure playlists using iTunes on the desktop or by creating an on-the-go playlist on the device. To your iOS application, playlists are read-only. To obtain playlists, configure a media query that is grouped by playlist. Each returned media item collection is a media playlist. The following code snippet illustrates this by logging playlist and song names to the Xcode debugger console:

```
MPMediaQuery *myPlaylistsQuery = [MPMediaQuery playlistsQuery];
NSArray *playlists = [myPlaylistsQuery collections];

for (MPMediaPlaylist *playlist in playlists) {
    NSLog (@"%@", [playlist valueForProperty: MPMediaPlaylistPropertyName]);

    NSArray *songs = [playlist items];
    for (MPMediaItem *song in songs) {
        NSString *songTitle =
            [song valueForProperty: MPMediaItemPropertyTitle];
        NSLog (@"\t\t%@", songTitle);
    }
}
```

The API for building a media query is described in *MPMediaPropertyPredicate Class Reference* and *MPMediaQuery Class Reference*. The methods for querying media playlist property values are described in *MPMediaEntity Class Reference*.

Constants

Playlist Attribute Flags

Playlist attributes, used as possible values for the `MPMediaPlaylistPropertyPlaylistAttributes` (page 45) property.

```
enum {
    MPMediaPlaylistAttributeNone    = 0,
    MPMediaPlaylistAttributeOnTheGo = (1 << 0),
    MPMediaPlaylistAttributeSmart   = (1 << 1),
    MPMediaPlaylistAttributeGenius  = (1 << 2)
};
typedef NSInteger MPMediaPlaylistAttribute;
```

Constants

`MPMediaPlaylistAttributeNone`

If set, the playlist has no attributes.

Available in iOS 3.0 and later.

Declared in `MPMediaPlaylist.h`.

`MPMediaPlaylistAttributeOnTheGo`

If set, the playlist was created on a device rather than synced from iTunes.

Available in iOS 3.0 and later.

Declared in `MPMediaPlaylist.h`.

`MPMediaPlaylistAttributeSmart`

If set, the playlist is a “smart” playlist, whose members are determined by user-specified rules.

Available in iOS 3.0 and later.

Declared in `MPMediaPlaylist.h`.

`MPMediaPlaylistAttributeGenius`

If set, the playlist is a Genius playlist.

Available in iOS 3.0 and later.

Declared in `MPMediaPlaylist.h`.

Playlist Property Keys

Use these keys with the `canFilterByProperty:` (page 14) and `valueForProperty:` (page 15) methods to obtain information about a playlist. Properties described as “filterable” can be used to build media property predicates (see *MPMediaPropertyPredicate Class Reference*).

```
NSString *const MPMediaPlaylistPropertyPersistentID;        // filterable
NSString *const MPMediaPlaylistPropertyName;                // filterable
NSString *const MPMediaPlaylistPropertyPlaylistAttributes;  // filterable
NSString *const MPMediaPlaylistPropertySeedItems;
```

Constants

`MPMediaPlaylistPropertyPersistentID`

The persistent identifier for the playlist. Value is an `NSNumber` object containing a `UInt64_t` (unsigned long long).

Can be used to build a media property predicate as described in *MPMediaQuery Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaPlaylist.h`.

`MPMediaPlaylistPropertyName`

The name of the playlist. Value is an `NSString` object.

Can be used to build a media property predicate as described in *MPMediaQuery Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaPlaylist.h`.

`MPMediaPlaylistPropertyPlaylistAttributes`

The attributes associated with the playlist. Value is an `NSNumber` object containing an `NSInteger` data type. Fields in the `NSInteger` identify the attributes of the playlist. A playlist may have any combination of attributes described in "Playlist Attribute Flags" (page 44).

Can be used to build a media property predicate as described in *MPMediaQuery Class Reference*.

Available in iOS 3.0 and later.

Declared in `MPMediaPlaylist.h`.

`MPMediaPlaylistPropertySeedItems`

The items seeded to generate the playlist; applies only to Genius playlists. Value is an `NSArray` object containing one or more `MPMediaItem` objects.

Value is `nil` for playlists that do not have the `MPMediaPlaylistAttributeGenius` (page 44) flag set.

Available in iOS 3.0 and later.

Declared in `MPMediaPlaylist.h`.

MPMediaPredicate Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide

Overview

Use this class's concrete subclass, described in *MPMediaPropertyPredicate Class Reference*, to define the filter in a media query to retrieve a subset of media items from the iPod library. Media queries are described in *MPMediaQuery Class Reference*.

In iPod library queries, a **predicate** is a statement of a logical condition that you want to test each media item against. Those media items that satisfy the condition are retrieved in the query result.

MPMediaPropertyPredicate Class Reference

Inherits from	MPMediaPredicate : NSObject
Conforms to	NSCoding (MPMediaPredicate) NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide

Overview

Use one or more `MPMediaPropertyPredicate` objects, or media property predicates, to define the filter in a media query to retrieve a subset of media items from the iPod library. A **predicate** in this context is a statement of a logical condition that you want to test each media item against. Those items that satisfy the condition are retrieved in the query result.

You define iPod library queries, and retrieve query results, using the `MPMediaQuery` class, described in *MPMediaQuery Class Reference*. The media items and media item collections that you retrieve with a query are described in *MPMediaItem Class Reference* and *MPMediaItemCollection Class Reference*.

Tasks

Creating Media Property Predicates

`+ predicateWithValue:forProperty:` (page 51)
Creates a media property predicate with the default comparison type.

`+ predicateWithValue:forProperty:comparisonType:` (page 51)
Creates a media property predicate with a specified comparison type.

Examining Media Property Predicates

`property` (page 50) *property*
The property that the media property predicate uses when you invoke a query.

`value` (page 50) *property*
The value that the media property predicate matches against when you invoke a query.

`comparisonType` (page 50) *property*
The type of matching comparison that the media property predicate performs when you invoke a query.

Properties

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

comparisonType

The type of matching comparison that the media property predicate performs when you invoke a query.

```
@property (nonatomic, readonly) MPMediaPredicateComparison comparisonType;
```

Discussion
For comparison types, see "Media Property Predicate Comparison Types" (page 52).

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

property

The property that the media property predicate uses when you invoke a query.

```
@property (nonatomic, readonly, copy) NSString *property;
```

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

value

The value that the media property predicate matches against when you invoke a query.

```
@property (nonatomic, readonly) id value;
```

Availability
Available in iOS 3.0 and later.

Declared In
MPMediaQuery.h

Class Methods

predicateWithValue:forProperty:

Creates a media property predicate with the default comparison type.

```
+ (MPMediaPropertyPredicate *)predicateWithValue:(id)value
    forProperty:(NSString *)property
```

Parameters

predicateWithValue:

The property value that you want to match when you query the iPod library. For example, if you specify the MPMediaItemPropertyArtist constant in the *forProperty* parameter, in this parameter you supply a string containing the artist name.

forProperty:

A so-called *filterable* property—one that can be used to build a media property predicate. See General Media Item Property Keys (page 20) and Podcast Item Property Keys (page 26) in *MPMediaItem Class Reference*.

Return Value
A media property predicate.

Discussion
This is a convenience method that uses the default logical comparison type of MPMediaPredicateComparisonEqualTo (page 52).

Availability
Available in iOS 3.0 and later.

See Also
+ predicateWithValue:forProperty:comparisonType: (page 51)

Declared In
MPMediaQuery.h

predicateWithValue:forProperty:comparisonType:

Creates a media property predicate with a specified comparison type.

```
+ (MPMediaPropertyPredicate *)predicateWithValue:(id)value
    forProperty:(NSString *)property
    comparisonType:(MPMediaPredicateComparison)comparisonType
```

Parameters

predicateWithValue:

The property value that you want to match when you query the iPod library. For example, if you specify the `MPMediaItemPropertyArtist` constant in the *forProperty* parameter, in this parameter you supply a string containing the artist name.

forProperty:

A so-called *filterable* property—one that can be used to build a media property predicate. See `General Media Item Property Keys` (page 20) and `Podcast Item Property Keys` (page 26) in *MPMediaItem Class Reference*.

comparisonType:

The logical comparison type for the predicate. See "Media Property Predicate Comparison Types" (page 52).

Return Value

A media property predicate.

Availability

Available in iOS 3.0 and later.

See Also

`+ predicateWithValue:forProperty:` (page 51)

Declared In

`MPMediaQuery.h`

Constants

Media Property Predicate Comparison Types

Logical comparison types for media queries.

```
enum {
    MPMediaPredicateComparisonEqualTo,
    MPMediaPredicateComparisonContains
};
typedef NSInteger MPMediaPredicateComparison;
```

Constants

`MPMediaPredicateComparisonEqualTo`

Matches when a media item's value for a given property is equal to the value in the media property predicate.

Available in iOS 3.0 and later.

Declared in `MPMediaQuery.h`.

`MPMediaPredicateComparisonContains`

Matches when a media item's value for a given property is contained in the value of the media property predicate.

Available in iOS 3.0 and later.

Declared in `MPMediaQuery.h`.

MPMediaQuery Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide

Overview

A media query specifies a set of media items (instances of `MPMediaItem`) from the iPod library by way of a filter and a grouping type. Filter and grouping type are both optional; an unqualified query matches the entire library.

A query has at most one grouping type. A query's filter can consist of any number of media property predicates. You build filters using methods described in *MPMediaPropertyPredicate Class Reference*, based on property keys described in *MPMediaItem Class Reference*.

After creating and configuring a query, you use it to retrieve media items or media item collections. Collections are described in *MPMediaItemCollection Class Reference*. You can also use a query to retrieve an array of `MPMediaQuerySection` instances, useful for displaying the results of a query in the user interface of your app. See the `itemSections` (page 58) and `collectionSections` (page 56) properties.

This class includes a number of convenience constructors that each apply a grouping type and, in most cases, match a subset of the iPod library. Table 10-1 summarizes the features of these constructors. See *MPMediaItem Class Reference* for descriptions of the entries in the Filter column. See "Media Item Collection Grouping Keys" (page 63) for descriptions of the entries in the Grouping type column.

Table 10-1 Convenience constructors from the MPMediaQuery class

Constructor name	Matches entire iPod library	Filter	Grouping type
`albumsQuery`	-	`MPMediaTypeMusic`	`MPMediaGroupingAlbum`
`artistsQuery`	-	`MPMediaTypeMusic`	`MPMediaGroupingArtist`

Constructor name	Matches entire iPod library	Filter	Grouping type
audiobooksQuery	-	MPMediaTypeAudioBook	MPMediaGroupingTitle
compilationsQuery	-	MPMediaTypeAny with MPMediaItemProperty-IsCompilation	MPMediaGroupingAlbum
composersQuery	Yes	MPMediaTypeAny	MPMediaGroupingComposer
genresQuery	Yes	MPMediaTypeAny	MPMediaGroupingGenre
playlistsQuery	Yes	MPMediaTypeAny	MPMediaGroupingPlaylist
podcastsQuery	-	MPMediaTypePodcast	MPMediaGrouping-PodcastTitle
songsQuery	-	MPMediaTypeMusic	MPMediaGroupingTitle

Tasks

Creating Media Queries

The class methods in this section create queries which you can use directly or modify as described in "Configuring Media Queries" (page 55). For each class method, the query's `groupingType` (page 57) property is set automatically according to the name of the method. For example, the `albumsQuery` method assigns a grouping type of `MPMediaGroupingAlbum`. The grouping type specifies the nature of the media item collections you can then retrieve from the query. Some class methods match the entire iPod library while others match a subset, as described in the Discussion sections for each method.

`+ albumsQuery` (page 58)
: Creates a media query that matches music items and that groups and sorts collections by album name.

`+ artistsQuery` (page 58)
: Creates a media query that matches music items and that groups and sorts collections by artist name.

`+ songsQuery` (page 61)
: Creates a media query that matches music items and that groups and sorts collections by song name.

`+ playlistsQuery` (page 60)
: Creates a media query that matches the entire iPod library and that groups and sorts collections by playlist name.

`+ podcastsQuery` (page 60)
: Creates a media query that matches podcast items and that groups and sorts collections by podcast name.

+ `audiobooksQuery` (page 59)
  Creates a media query that matches audio book items and that groups and sorts collections by audio book name.

+ `compilationsQuery` (page 59)
  Creates a media query that matches compilation items and that groups and sorts collections by album name.

+ `composersQuery` (page 60)
  Creates a media query that matches all media items and that groups and sorts collections by composer name.

+ `genresQuery` (page 60)
  Creates a media query that matches all media items and that groups and sorts collections by genre name.

- `init` (page 62)
  Initializes a generic media query.

- `initWithFilterPredicates:` (page 62)
  Initializes a media query with a set of media property predicates.

Configuring Media Queries

`filterPredicates` (page 57) *property*
  The media property predicates of the media query.

`groupingType` (page 57) *property*
  The grouping for collections retrieved with the media query.

`itemSections` (page 58) *property*
  An array of `MPMediaQuerySection` instances representing the section grouping of the query's specified media items.

`collectionSections` (page 56) *property*
  An array of `MPMediaQuerySection` instances representing the section grouping of the query's specified media item collections.

- `addFilterPredicate:` (page 61)
  Adds a media property predicate to a query.

- `removeFilterPredicate:` (page 62)
  Removes a filter predicate from a query.

Performing Media Queries

You obtain a specified array of media items or media item collections from the iPod library by calling the `items` (page 57) or `collections` (page 56) accessor methods.

`items` (page 57) *property*
  An array of media items that match the media query's predicate.

`collections` (page 56) *property*
An array of media item collections whose contained items match the query's media property predicate.

Properties

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

collections

An array of media item collections whose contained items match the query's media property predicate.

```
@property (nonatomic, readonly) NSArray *collections;
```

Discussion
The returned array of collections are grouped and sorted by the `groupingType` of the media query. The following code snippet illustrates how to use this property:

```
// Specify a media query; this one matches the entire iPod library because it
// does not contain a media property predicate
MPMediaQuery *everything = [[MPMediaQuery alloc] init];

// Configure the media query to group its media items; here, grouped by artist
[everything setGroupingType: MPMediaGroupingArtist];

// Obtain the media item collections from the query
NSArray *collections = [everything collections];
```

Each element of the `collections` array now contains a media item collection. Each collection contains the media items from the iPod library by a particular artist. The elements of the array are sorted by artist name.

For the available grouping types, see "Media Item Collection Grouping Keys" (page 63).

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

collectionSections

An array of `MPMediaQuerySection` instances representing the section grouping of the query's specified media item collections.

```
@property (nonatomic, readonly) NSArray *collectionSections;
```

Discussion
The value of this property may be `nil` if there is no appropriate section grouping of the media item collections.

Availability
Available in iOS 4.2 and later.

Declared In
`MPMediaQuery.h`

filterPredicates

The media property predicates of the media query.

```
@property (nonatomic, retain) NSSet *filterPredicates;
```

Discussion
The `General Media Item Property Keys` (page 20) and `Podcast Item Property Keys` (page 26) enumerations in *MPMediaItem Class Reference* contain the keys you can use to construct predicates.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

groupingType

The grouping for collections retrieved with the media query.

```
@property (nonatomic) MPMediaGrouping groupingType;
```

Discussion
The default grouping type is `MPMediaGroupingTitle` (page 63). See "Media Item Collection Grouping Keys" (page 63) for the list of available grouping types.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

items

An array of media items that match the media query's predicate.

```
@property (nonatomic, readonly) NSArray *items;
```

Discussion
If no items match the predicate, this method returns an empty array. On error, returns `nil`.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

itemSections

An array of `MPMediaQuerySection` instances representing the section grouping of the query's specified media items.

```
@property (nonatomic, readonly) NSArray *itemSections;
```

Discussion
The value of this property may be `nil` if there is no appropriate section grouping of the media items.

Availability
Available in iOS 4.2 and later.

Declared In
`MPMediaQuery.h`

Class Methods

albumsQuery

Creates a media query that matches music items and that groups and sorts collections by album name.

```
+ (MPMediaQuery *)albumsQuery
```

Return Value
A media query that matches media items of type `MPMediaTypeMusic` (page 19) and has a grouping type of `MPMediaGroupingAlbum` (page 63).

Discussion
A media item can have more than one media type; for example, an item could be of types “music” and “podcast.” An `albumsQuery` query matches all `MPMediaTypeMusic` items, whether or not they are also of other media types.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

artistsQuery

Creates a media query that matches music items and that groups and sorts collections by artist name.

```
+ (MPMediaQuery *)artistsQuery
```

Return Value
A media query that matches media items of type `MPMediaTypeMusic` (page 19) and has a grouping type of `MPMediaGroupingArtist` (page 63).

Discussion
A media item can have more than one media type; for example, an item could be of types “music” and “podcast.” An `artistsQuery` query matches all `MPMediaTypeMusic` items, whether or not they are also of other media types.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

audiobooksQuery

Creates a media query that matches audio book items and that groups and sorts collections by audio book name.

```
+ (MPMediaQuery *)audiobooksQuery
```

Return Value
A media query that matches media items of type `MPMediaTypeAudioBook` (page 20) and that uses the default grouping type of `MPMediaGroupingTitle` (page 63).

Discussion
A media item can have more than one media type; for example, an item could be of types “music” and “podcast.” An `audiobooksQuery` query matches all `MPMediaTypeAudioBook` (page 20) items, whether or not they are also of other media types.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

compilationsQuery

Creates a media query that matches compilation items and that groups and sorts collections by album name.

```
+ (MPMediaQuery *)compilationsQuery
```

Return Value
A media query that matches any media item that has the `MPMediaItemPropertyIsCompilation` (page 25) property; the returned query has a grouping type of `MPMediaGroupingAlbum` (page 63).

Discussion
A media item can have more than one media type; for example, an item could be of types “music” and “podcast.” A `compilationsQuery` query matches all media items that have the `MPMediaItemPropertyIsCompilation` property, irrespective of their media types.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

composersQuery

Creates a media query that matches all media items and that groups and sorts collections by composer name.

```
+ (MPMediaQuery *)composersQuery
```

Return Value
A media query that matches all media items and that has a grouping type of `MPMediaGroupingComposer` (page 63).

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

genresQuery

Creates a media query that matches all media items and that groups and sorts collections by genre name.

```
+ (MPMediaQuery *)genresQuery
```

Return Value
A media query that matches all media items and that has a grouping type of `MPMediaGroupingGenre` (page 64).

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

playlistsQuery

Creates a media query that matches the entire iPod library and that groups and sorts collections by playlist name.

```
+ (MPMediaQuery *)playlistsQuery
```

Return Value
A media query that matches all media items and that has a grouping type of `MPMediaGroupingPlaylist` (page 64).

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

podcastsQuery

Creates a media query that matches podcast items and that groups and sorts collections by podcast name.

```
+ (MPMediaQuery *)podcastsQuery
```

Return Value
A media query that matches media items of type `MPMediaTypePodcast` (page 19) and that has a grouping type of `MPMediaGroupingPodcastTitle` (page 64).

Discussion
A media item can have more than one media type; for example, an item could be of types "music" and "podcast." A `podcastsQuery` query matches all `MPMediaTypePodcast` (page 19) items, whether or not they are also of other media types.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

songsQuery

Creates a media query that matches music items and that groups and sorts collections by song name.

```
+ (MPMediaQuery *)songsQuery
```

Return Value
A media query that matches media items of type `MPMediaTypeMusic` (page 19) and has a grouping type of `MPMediaGroupingTitle` (page 63).

Discussion
A media item can have more than one media type; for example, an item could be of types "music" and "podcast." A `songsQuery` query matches all `MPMediaTypeMusic` items, whether or not they are also of other media types.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

Instance Methods

addFilterPredicate:

Adds a media property predicate to a query.

```
- (void)addFilterPredicate:(MPMediaPredicate *)predicate
```

Parameters
predicate
The media predicate to add to the set of predicates for the query.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

init

Initializes a generic media query.

```
- (id)init
```

Discussion
A generic media query has no filter predicates and no grouping configuration. It matches everything in the iPod library and provides no grouping or sorting.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

initWithFilterPredicates:

Initializes a media query with a set of media property predicates.

```
- (id)initWithFilterPredicates:(NSSet *)filterPredicates
```

Parameters
filterPredicates
The set of media property predicates to use as a filter on the iPod library.

Return Value
An initialized media query.

Discussion
MPMediaPropertyPredicate Class Reference describes how to create media property predicates. The `General Media Item Property Keys` (page 20) and `Podcast Item Property Keys` (page 26) enumerations in *MPMediaItem Class Reference* contain the keys you can use to construct predicates.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

removeFilterPredicate:

Removes a filter predicate from a query.

```
- (void)removeFilterPredicate:(MPMediaPredicate *)predicate
```

Parameters
predicate
The media predicate to remove from the set of predicates for the query.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMediaQuery.h`

Constants

Media Item Collection Grouping Keys

Keys used with the `groupingType` property to configure a media query.

```
enum {
    MPMediaGroupingTitle,
    MPMediaGroupingAlbum,
    MPMediaGroupingArtist,
    MPMediaGroupingAlbumArtist,
    MPMediaGroupingComposer,
    MPMediaGroupingGenre,
    MPMediaGroupingPlaylist,
    MPMediaGroupingPodcastTitle,
};
typedef NSInteger MPMediaGrouping;
```

Constants

`MPMediaGroupingTitle`
Groups and sorts media item collections by title. For songs, for example, the title is the song name. This is the default grouping key.
Available in iOS 3.0 and later.
Declared in `MPMediaQuery.h`.

`MPMediaGroupingAlbum`
Groups and sorts media item collections by album, and sorts songs within an album by track order.
Available in iOS 3.0 and later.
Declared in `MPMediaQuery.h`.

`MPMediaGroupingArtist`
Groups and sorts media item collections by performing artist.
Available in iOS 3.0 and later.
Declared in `MPMediaQuery.h`.

`MPMediaGroupingAlbumArtist`
Groups and sorts media item collections by album artist (the primary performing artist for an album as a whole).
Available in iOS 3.0 and later.
Declared in `MPMediaQuery.h`.

`MPMediaGroupingComposer`
Groups and sorts media item collections by composer.
Available in iOS 3.0 and later.
Declared in `MPMediaQuery.h`.

`MPMediaGroupingGenre`
: Groups and sorts media item collections by musical or film genre.

  Available in iOS 3.0 and later.

  Declared in `MPMediaQuery.h`.

`MPMediaGroupingPlaylist`
: Groups and sorts media item collections by playlist.

  Available in iOS 3.0 and later.

  Declared in `MPMediaQuery.h`.

`MPMediaGroupingPodcastTitle`
: Groups and sorts media item collections by podcast title.

  Available in iOS 3.0 and later.

  Declared in `MPMediaQuery.h`.

Discussion
The following code snippet shows how to apply a grouping key:

```
MPMediaQuery *everything = [[MPMediaQuery alloc] init];
[everything setGroupingType: MPMediaGroupingAlbum];
NSArray *collections = [everything collections];
```

After running these code lines, the `collections` array contains all the matched media items grouped and sorted according to album name.

To obtain a sorted list of songs, configure a media query with the `MPMediaGroupingTitle` key, or take advantage of the title key being the default for a media query. In either case, each obtained media item is, in effect, its own collection.

Collections sort according to the same rules used by iTunes on the desktop. This includes respecting the primary system language chosen by the user. Leading articles, including “A,” “An,” and “The” when using English, or “L’,” “La,” and “Le” when using French, are ignored during sorting. If you need precise control over sorting, implement it in your application.

MPMediaQuerySection Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 4.2 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide

Overview

A media query section represents a range of media items or media item collections from within an iPod library media query. You can use sections when displaying a query's items or collections in your app's user interface.

You obtain an array of media query sections by using the `itemSections` or `collectionSections` properties of a media query (an instance of the `MPMediaQuery` class). The property values of a media query section are read-only.

Tasks

Working with Media Query Sections

`title` (page 66) *property*
The localized title of the media query section.

`range` (page 66) *property*
The range in the media query's items or collections array that is represented by the media query section.

Properties

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

range

The range in the media query's items or collections array that is represented by the media query section.

```
@property (nonatomic, assign, readonly) NSRange range
```

Availability
Available in iOS 4.2 and later.

Declared In
`MPMediaQuerySection.h`

title

The localized title of the media query section.

```
@property (nonatomic, copy, readonly) NSString *title
```

Availability
Available in iOS 4.2 and later.

Declared In
`MPMediaQuerySection.h`

MPMovieAccessLog Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 4.3 and later.
Declared in	

Overview

A movie access log accumulates key metrics about network playback for an associated movie player that is playing streamed content. The log presents these metrics as a collection of `MPMovieAccessLogEvent` instances and also makes it available in a textual format. A movie access log describes one uninterrupted period of playback. A movie player (an instance of the `MPMoviePlayerController` class) can access this log from its `setAccessLog:` property.

All movie access log properties are read-only.

Tasks

Movie Access Log Properties

`extendedLogData` (page 68) *property*
A textual version of the web server access log for the associated movie player.

`extendedLogDataStringEncoding` (page 68) *property*
The string encoding for the `extendedLogData` (page 68) property.

`events` (page 68) *property*
The events in the movie access log.

Properties

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

events

The events in the movie access log.

```
@property (nonatomic, readonly) NSArray *events;
```

Discussion
An ordered collection of `MPMovieAccessLogEvent` instances that represent the chronological sequence of events in the movie access log.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

extendedLogData

A textual version of the web server access log for the associated movie player.

```
@property (nonatomic, readonly) NSData *extendedLogData;
```

Discussion
The web server access log in a textual format that conforms to the W3C Extended Log File Format for web server log files. For more information, see http://www.w3.org/pub/WWW/TR/WD-logfile.html.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

extendedLogDataStringEncoding

The string encoding for the `extendedLogData` (page 68) property.

```
@property (nonatomic, readonly) NSStringEncoding extendedLogDataStringEncoding;
```

Discussion
For possible string encodings, see `String_Encodings`.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

MPMovieAccessLogEvent Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 4.3 and later.
Declared in	

Overview

A movie access log event represents a single piece of information for a movie access log. For a description of movie access logs, see *MPMovieAccessLog Class Reference*.

Tasks

Movie Access Log Event Properties

`numberOfSegmentsDownloaded` (page 71) *property*
A count of media segments downloaded from the web server to your app.

`playbackStartDate` (page 72) *property*
The timestamp for when playback began for the movie log access event.

`URI` (page 73) *property*
The URI of the playback item.

`serverAddress` (page 73) *property*
The IP address of the web server that was the source of the last delivered media segment. Can be either an IPv4 or an IPv6 address.

`numberOfServerAddressChanges` (page 71) *property*
A count of changes to the `serverAddress` (page 73) property over the last uninterrupted period of playback.

`playbackSessionID` (page 72) *property*
A GUID that identifies the playback session. This value is used in HTTP requests.

`playbackStartOffset` (page 73) *property*
An offset into the playlist where the last uninterrupted period of playback began, in seconds.

`segmentsDownloadedDuration` (page 73) *property*
The accumulated duration of the media downloaded, in seconds.

`durationWatched` (page 70) *property*
The accumulated duration of the media played, in seconds.

`numberOfStalls` (page 72) *property*
The total number of playback stalls encountered.

`numberOfBytesTransferred` (page 71) *property*
The accumulated number of bytes transferred.

`observedBitrate` (page 72) *property*
The empirical throughput across all media downloaded for the movie player, in bits per second.

`indicatedBitrate` (page 70) *property*
The throughput required to play the stream, as advertised by the web server, in bits per second.

`numberOfDroppedVideoFrames` (page 71) *property*
The total number of dropped video frames.

Properties

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

durationWatched

The accumulated duration of the media played, in seconds.

```
@property (nonatomic, readonly) NSTimeInterval durationWatched;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

indicatedBitrate

The throughput required to play the stream, as advertised by the web server, in bits per second.

```
@property (nonatomic, readonly) double indicatedBitrate;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

numberOfBytesTransferred

The accumulated number of bytes transferred.

```
@property (nonatomic, readonly) int64_t numberOfBytesTransferred;
```

Discussion
If the number of bytes transferred is unknown, this property's value is negative.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

numberOfDroppedVideoFrames

The total number of dropped video frames.

```
@property (nonatomic, readonly) NSInteger numberOfDroppedVideoFrames;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

numberOfSegmentsDownloaded

A count of media segments downloaded from the web server to your app.

```
@property (nonatomic, readonly) NSUInteger numberOfSegmentsDownloaded;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

numberOfServerAddressChanges

A count of changes to the `serverAddress` (page 73) property over the last uninterrupted period of playback.

```
@property (nonatomic, readonly) NSUInteger numberOfServerAddressChanges;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

numberOfStalls

The total number of playback stalls encountered.

```
@property (nonatomic, readonly) NSInteger numberOfStalls;
```

Discussion
If the number of playback stalls is unknown, this property's value is negative.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

observedBitrate

The empirical throughput across all media downloaded for the movie player, in bits per second.

```
@property (nonatomic, readonly) double observedBitrate;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

playbackSessionID

A GUID that identifies the playback session. This value is used in HTTP requests.

```
@property (nonatomic, readonly) NSString *playbackSessionID;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

playbackStartDate

The timestamp for when playback began for the movie log access event.

```
@property (nonatomic, readonly) NSDate *playbackStartDate;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

playbackStartOffset

An offset into the playlist where the last uninterrupted period of playback began, in seconds.

```
@property (nonatomic, readonly) NSTimeInterval playbackStartOffset;
```

Discussion
If the playback start offset is unknown, this property's value is negative.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

segmentsDownloadedDuration

The accumulated duration of the media downloaded, in seconds.

```
@property (nonatomic, readonly) NSTimeInterval segmentsDownloadedDuration;
```

Discussion
If the accumulated media duration is unknown, this property's value is negative.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

serverAddress

The IP address of the web server that was the source of the last delivered media segment. Can be either an IPv4 or an IPv6 address.

```
@property (nonatomic, readonly) NSString *serverAddress;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

URI

The URI of the playback item.

```
@property (nonatomic, readonly) NSString *URI;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

MPMovieErrorLog Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 4.3 and later.
Declared in	

Overview

A movie error log contains data describing network resource playback failures for the associated movie player. The data includes timestamps indicating when each failure occurred.

All movie error log properties are read-only.

Tasks

Movie Error Log Properties

`extendedLogData` (page 76) *property*
A textual version of the web server error log.

`extendedLogDataStringEncoding` (page 76) *property*
The string encoding for the `extendedLogData` (page 76) property.

`events` (page 76) *property*
The events in the movie error log.

Properties

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

events

The events in the movie error log.

```
@property (nonatomic, readonly) NSArray *events;
```

Discussion
An ordered collection of `MPMovieErrorLogEvent` instances that represent the chronological sequence of events in the movie error log.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

extendedLogData

A textual version of the web server error log.

```
@property (nonatomic, readonly) NSData *extendedLogData;
```

Discussion
The web server error log in a textual format that conforms to the W3C Extended Log File Format for web server log files. For more information, see http://www.w3.org/pub/WWW/TR/WD-logfile.html.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

extendedLogDataStringEncoding

The string encoding for the `extendedLogData` (page 76) property.

```
@property (nonatomic, readonly) NSStringEncoding extendedLogDataStringEncoding;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

MPMovieErrorLogEvent Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 4.3 and later.
Declared in	

Overview

A movie error log event represents a single piece of information for a movie error log. For a description of movie error logs, see *MPMovieErrorLog Class Reference*.

All movie error log event properties are read-only.

Tasks

Movie Error Log Event Properties

`date` (page 78) *property*
The date and time when the error occurred.

`URI` (page 79) *property*
The URI of the item being played when the error occurred.

`serverAddress` (page 79) *property*
The IP address of the web server that was the source of the error.

`playbackSessionID` (page 79) *property*
A GUID (globally unique identifier) for the playback session.

`errorStatusCode` (page 78) *property*
A unique error code identifier.

`errorDomain` (page 78) *property*
The network domain of the error.

`errorComment` (page 78) *property*
A description of the error.

Properties

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

date

The date and time when the error occurred.

```
@property (nonatomic, readonly) NSDate *date;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

errorComment

A description of the error.

```
@property (nonatomic, readonly) NSString *errorComment;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

errorDomain

The network domain of the error.

```
@property (nonatomic, readonly) NSString *errorDomain;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

errorStatusCode

A unique error code identifier.

```
@property (nonatomic, readonly) NSInteger errorStatusCode;
```

Discussion
If the error is unknown, the value of this property is negative.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

playbackSessionID

A GUID (globally unique identifier) for the playback session.

```
@property (nonatomic, readonly) NSString *playbackSessionID;
```

Discussion
The GUID is used in HTTP requests.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

serverAddress

The IP address of the web server that was the source of the error.

```
@property (nonatomic, readonly) NSString *serverAddress;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

URI

The URI of the item being played when the error occurred.

```
@property (nonatomic, readonly) NSString *URI;
```

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerController Class Reference

Inherits from	NSObject
Conforms to	MPMediaPlayback NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 2.0 and later.
Declared in	
Companion guide	Multimedia Programming Guide
Related sample code	MoviePlayer

Overview

A movie player (of type `MPMoviePlayerController`) manages the playback of a movie from a file or a network stream. Playback occurs in a view owned by the movie player and takes place either fullscreen or inline. You can incorporate a movie player's view into a view hierarchy owned by your app, or use an `MPMoviePlayerViewController` object to manage the presentation for you.

Starting in iOS 4.3, movie players support wireless movie playback to AirPlay-enabled hardware such as Apple TV. To enable AirPlay playback, set the `allowsAirPlay` (page 86) property to `YES`. The movie player then presents a control that allows the user to choose AirPlay-enabled hardware for playback when such hardware is in range.

When you add a movie player's view to your app's view hierarchy, be sure to size the frame correctly, as shown here:

```
MPMoviePlayerController *player =
        [[MPMoviePlayerController alloc] initWithContentURL: myURL];
[player.view setFrame: myView.bounds];  // player's frame must match parent's
[myView addSubview: player.view];
// ...
[player play];
```

Consider a movie player view to be an opaque structure. You can add your own custom subviews to layer content on top of the movie but you must never modify any of its existing subviews.

In addition to layering content on top of a movie, you can provide custom background content by adding subviews to the view in the `backgroundView` (page 87) property. Custom subviews are supported in both inline and fullscreen playback modes but you must adjust the positions of your views when entering or

exiting fullscreen mode. Use the `MPMoviePlayerWillEnterFullscreenNotification` and `MPMoviePlayerWillExitFullscreenNotification` notifications to detect changes to and from fullscreen mode.

This class supports programmatic control of movie playback, and user-based control via buttons supplied by the movie player. You can control most aspects of playback programmatically using the methods and properties of the `MPMediaPlayback` protocol, to which this class conforms. The methods and properties of that protocol let you start and stop playback, seek forward and backward through the movie's content, and even change the playback rate. In addition, the `controlStyle` (page 88) property of this class lets you display a set of standard system controls that allow the user to manipulate playback. You can also set the `shouldAutoplay` (page 93) property for network-based content to start automatically.

You typically specify the movie you want to play when you create a new `MPMoviePlayerController` object. However, you can also change the currently playing movie by changing the value in the `contentURL` (page ?) property. Changing this property lets you reuse the same movie player controller object in multiple places. For performance reasons you may want to play movies as local files. Do this by first downloading them to a local directory.

Note: Although you can create multiple `MPMoviePlayerController` objects and present their views in your interface, only one movie player at a time can play its movie.

To facilitate the creation of video bookmarks or chapter links for a long movie, the `MPMoviePlayerController` class defines methods for generating thumbnail images at specific times within a movie. You can request a single thumbnail image using the `thumbnailImageAtTime:timeOption:` (page 97) method or request multiple thumbnail images using the `requestThumbnailImagesAtTimes:timeOption:` (page 96) method.

To play a network stream whose URL requires access credentials, first create an appropriate `NSURLCredential` object. Do this by calling, for example, the `initWithUser:password:persistence:` method, as shown here:

```
NSURLCredential *credential = [[NSURLCredential alloc]
                        initWithUser: @"userName"
                            password: @"password"
                         persistence: NSURLCredentialPersistenceForSession];

self.credential = credential;
[credential release];
```

In addition, create an appropriate `NSURLProtectionSpace` object, as shown here. Make appropriate modifications for the realm you are accessing:

```
NSURLProtectionSpace *protectionSpace = [[NSURLProtectionSpace alloc]
                             initWithHost: "@streams.mydomain.com"
                                     port: 80
                                 protocol: @"http"
                                    realm: @"mydomain.com"
                       authenticationMethod: NSURLAuthenticationMethodDefault];

self.protectionSpace = protectionSpace;
[protectionSpace release];
```

Add the URL credential and the protection space to the Singleton `NSURLCredentialStorage` object. Do this by calling, for example, the `setCredential:forProtectionSpace:` method, as shown here:

```
[[NSURLCredentialStorage sharedCredentialStorage]
                    setDefaultCredential: credential
                      forProtectionSpace: protectionSpace];
```

With the credential and protection space information in place, you can then play the protected stream.

Movie Player Notifications

The `MPMoviePlayerController` class generates numerous notifications to keep your application informed about the state of movie playback. In addition to being notified when playback finishes, interested clients can be notified in the following situations:

- When the movie player begins playing, is paused, or begins seeking forward or backward
- When the scaling mode of the movie changes
- When the movie enters or exits fullscreen mode
- When the load state for network-based movies changes
- When meta information about the movie itself becomes available

For more information, see the Notifications section in this document.

Supported Formats

This class plays any movie or audio file supported in iOS. This includes both streamed content and fixed-length files. For movie files, this typically means files with the extensions `.mov`, `.mp4`, `.mpv`, and `.3gp` and using one of the following compression standards:

- H.264 Baseline Profile Level 3.0 video, up to 640 x 480 at 30 fps. (The Baseline profile does not support B frames.)
- MPEG-4 Part 2 video (Simple Profile)

If you use this class to play audio files, it displays a white screen with a QuickTime logo while the audio plays. For audio files, this class supports AAC-LC audio at up to 48 kHz, and MP3 (MPEG-1 Audio Layer 3) up to 48 kHz, stereo audio.

Behavior in iOS 3.1 and Earlier

In iOS 3.1 and earlier, this class implemented a full-screen movie player only. After creating the movie player and initializing it with a single movie file, you called the `play` method to present the movie. (The definition of the `play` method has since moved out of this class and into the `MPMediaPlayback` protocol.) The movie player object itself handled the actual presentation of the movie content.

Tasks

Creating and Initializing the Object

- `initWithContentURL:` (page 95)
  Returns a `MPMoviePlayerController` object initialized with the movie at the specified URL.

Accessing Movie Properties

`contentURL` (page 88) *property*
The URL that points to the movie file.

`movieSourceType` (page 91) *property*
The playback type of the movie.

`movieMediaTypes` (page 91) *property*
The types of media available in the movie. (read-only)

`naturalSize` (page 91) *property*
The width and height of the movie frame. (read-only)

`fullscreen` (page 89) *property*
A Boolean that indicates whether the movie player is in full-screen mode.

- `setFullscreen:animated:` (page 96)
  Causes the movie player to enter or exit full-screen mode.

`scalingMode` (page 93) *property*
The scaling mode to use when displaying the movie.

`controlStyle` (page 88) *property*
The style of the playback controls.

`useApplicationAudioSession` (page 94) *property*
A Boolean value that indicates whether the movie player should use the application's audio session.

Accessing the Movie Duration

`duration` (page 88) *property*
The duration of the movie, measured in seconds. (read-only)

`playableDuration` (page 92) *property*
The amount of currently playable content. (read-only)

Accessing the View

`view` (page 94) *property*
The view containing the movie content and controls. (read-only)

`backgroundView` (page 87) *property*
A customizable view that is displayed behind the movie content. (read-only)

Controlling and Monitoring Playback

See also the methods of the `MPMediaPlayback` protocol.

`loadState` (page 90) *property*
The network load state of the movie player. (read-only)

`playbackState` (page 92) *property*
The current playback state of the movie player. (read-only)

`initialPlaybackTime` (page 90) *property*
The time, specified in seconds within the video timeline, when playback should start.

`endPlaybackTime` (page 89) *property*
The end time (measured in seconds) for playback of the movie.

`shouldAutoplay` (page 93) *property*
A Boolean that indicates whether a movie should begin playback automatically.

`repeatMode` (page 93) *property*
Determines how the movie player repeats the playback of the movie.

- `timedMetadata` (page 97)
Obtains the most recent time-based metadata provided by the streamed movie.

`allowsAirPlay` (page 86) *property*
Specifies whether the movie player allows AirPlay movie playback.

Generating Thumbnail Images

- `thumbnailImageAtTime:timeOption:` (page 97)
Captures and returns a thumbnail image from the current movie.

- `requestThumbnailImagesAtTimes:timeOption:` (page 96)
Captures one or more thumbnail images asynchronously from the current movie.

- `cancelAllThumbnailImageRequests` (page 95)
Cancels all pending asynchronous thumbnail image requests.

Retrieving Movie Logs

`accessLog` (page 86) *property*
A snapshot of the network playback log for the movie player if it is playing a network stream.

`errorLog` (page 89) *property*
A snapshot of the playback failure error log for the movie player if it is playing a network stream.

Deprecated Methods and Properties

The following methods and properties are no longer available in iOS 3.2 and must not be used.

`backgroundColor` (page 87) *property*
The color of the background area behind the movie. (Deprecated. Get the view from the `backgroundView` (page 87) property and set its color directly.)

`movieControlMode` (page 90) *property* Available in iOS 2.0 through iOS 3.1
The user controls to display. (Deprecated. Use the "`Accessing Movie Properties`" (page 84) property instead.)

Properties

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

accessLog

A snapshot of the network playback log for the movie player if it is playing a network stream.

```
@property (nonatomic, readonly) MPMovieAccessLog *accessLog;
```

Discussion
Can be `nil`. For information about movie access logs, refer to *MPMovieAccessLog Class Reference*.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

allowsAirPlay

Specifies whether the movie player allows AirPlay movie playback.

```
@property (nonatomic) BOOL allowsAirPlay;
```

Discussion
A movie player supports wireless movie playback to AirPlay-enabled hardware. By default, this property's value is `NO`.

To enable AirPlay movie playback, set this property's value to YES. The movie player then presents a control that allows the user to choose AirPlay-enabled hardware for playback when such hardware is in range.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

backgroundColor

The color of the background area behind the movie. (Available in iOS 2.0 through iOS 3.1. Get the view from the `backgroundView` (page 87) property and set its color directly.)

```
@property (nonatomic, retain) UIColor *backgroundColor
```

Discussion
You should avoid using this property. It is available only when you use the `initWithContentURL:` (page 95) method to initialize the movie player controller object.

The receiver fades to and from the background color when transitioning to and from playback. Whenever the movie does not fill the screen exactly, this color is used to fill the area between the movie's frame and the edges of the screen.

The default color for this property is black. You can change this to other colors (including clear) to provide a more appropriate transition from your application's content to the movie content.

Availability
Available in iOS 2.0 through iOS 3.1.

Related Sample Code
MoviePlayer

Declared In
`MPMoviePlayerController.h`

backgroundView

A customizable view that is displayed behind the movie content. (read-only)

```
@property (nonatomic, readonly) UIView *backgroundView
```

Discussion
This view provides the backing content, on top of which the movie content is displayed. You can add subviews to the background view if you want to display custom background content.

This view is part of the view hierarchy returned by the `view` (page 94) property.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

contentURL

The URL that points to the movie file.

```
@property (nonatomic, copy) NSURL *contentURL
```

Discussion
If you set this property while a movie is playing, that movie pauses and the new movie begins loading. The new movie starts playing at the beginning.

Availability
Available in iOS 2.0 and later.

Declared In
`MPMoviePlayerController.h`

controlStyle

The style of the playback controls.

```
@property (nonatomic) MPMovieControlStyle controlStyle
```

Discussion
The default value of this property is `MPMovieControlStyleDefault` (page 99). You can change the value of this property to change the style of the controls or to hide the controls altogether. For a list of available control styles, see "MPMovieControlStyle" (page 98).

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

duration

The duration of the movie, measured in seconds. (read-only)

```
@property (nonatomic, readonly) NSTimeInterval duration
```

Discussion
If the duration of the movie is not known, the value in this property is `0.0`. If the duration is subsequently determined, this property is updated and a `MPMovieDurationAvailableNotification` (page 105) notification is posted.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

endPlaybackTime

The end time (measured in seconds) for playback of the movie.

```
@property (nonatomic) NSTimeInterval endPlaybackTime
```

Discussion
The default value of this property is -1, which indicates the natural end time of the movie. This property is not applicable for streamed content.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

errorLog

A snapshot of the playback failure error log for the movie player if it is playing a network stream.

```
@property (nonatomic, readonly) MPMovieErrorLog *errorLog;
```

Discussion
Can be `nil`. For information about movie error logs, refer to *MPMovieErrorLog Class Reference*.

Availability
Available in iOS 4.3 and later.

Declared In
`MPMoviePlayerController.h`

fullscreen

A Boolean that indicates whether the movie player is in full-screen mode.

```
@property (nonatomic, getter=isFullscreen) BOOL fullscreen
```

Discussion
The default value of this property is `NO`. Changing the value of this property causes the movie player to enter or exit full-screen mode immediately. If you want to animate the transition to full-screen mode, use the `setFullscreen:animated:` method instead.

Whenever the movie player enters or exits full-screen mode, it posts appropriate notifications to reflect the change. For example, upon entering full-screen mode, it posts `MPMoviePlayerWillEnterFullscreenNotification` (page 109) and `MPMoviePlayerDidEnterFullscreenNotification` (page 106) notifications. Upon exiting from full-screen mode, it posts `MPMoviePlayerWillExitFullscreenNotification` (page 109) and `MPMoviePlayerDidExitFullscreenNotification` (page 107) notifications.

The value of this property may also change as a result of the user interacting with the movie player controls.

Availability
Available in iOS 3.2 and later.

See Also
- setFullscreen:animated: (page 96)

Declared In
MPMoviePlayerController.h

initialPlaybackTime

The time, specified in seconds within the video timeline, when playback should start.

```
@property (nonatomic) NSTimeInterval initialPlaybackTime
```

Discussion
For progressively downloaded content, playback starts at the closest key frame prior to the provided time. For video-on-demand content, playback starts at the nearest segment boundary to the provided time. For live video streams, the playback start time is measured from the start of the current playlist and is rounded to the nearest segment boundary.

The default value of this property is -1, which indicates the natural start time of the movie.

Availability
Available in iOS 3.0 and later.

Declared In
MPMoviePlayerController.h

loadState

The network load state of the movie player. (read-only)

```
@property (nonatomic, readonly) MPMovieLoadState loadState
```

Availability
Available in iOS 3.2 and later.

Declared In
MPMoviePlayerController.h

movieControlMode

The user controls to display. (Available in iOS 2.0 through iOS 3.1. Use the "Accessing Movie Properties" (page 84) property instead.)

```
@property (nonatomic) MPMovieControlMode movieControlMode
```

Discussion
You should avoid using this property. It is available only when you use the initWithContentURL: (page 95) method to initialize the movie player controller object.

Determines the control (if any) the user has over movie playback. Different modes give the user access to different sets of playback controls, some of which allow the user to pause and resume playback and some of which do not.

This property is set to `MPMovieControlModeDefault` by default. See the "MPMovieControlMode" (page 105) enumeration for the available control modes.

Availability
Available in iOS 2.0 through iOS 3.1.

Related Sample Code
MoviePlayer

Declared In
`MPMoviePlayerController.h`

movieMediaTypes

The types of media available in the movie. (read-only)

```
@property (nonatomic, readonly) MPMovieMediaTypeMask movieMediaTypes
```

Discussion
Movies can contain a combination of audio, video, or a combination of the two. The default value of this property is `MPMovieMediaTypeMaskNone` (page 103).

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

movieSourceType

The playback type of the movie.

```
@property (nonatomic) MPMovieSourceType movieSourceType
```

Discussion
The default value of this property is `MPMovieSourceTypeUnknown` (page 103). This property provides a clue to the playback system as to how it should download and buffer the movie content. If you know the source type of the movie, setting the value of this property before playback begins can improve the load times for the movie content. If you do not set the source type explicitly before playback, the movie player controller must gather this information, which might delay playback.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

naturalSize

The width and height of the movie frame. (read-only)

```
@property (nonatomic, readonly) CGSize naturalSize
```

Discussion
This property reports the clean aperture of the video in square pixels. Thus, the reported dimensions take into account anamorphic content and aperture modes.

It is possible for the natural size of a movie to change during playback. This typically happens when the bit-rate of streaming content changes or when playback toggles between audio-only and a combination of audio and video.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

playableDuration

The amount of currently playable content. (read-only)

```
@property (nonatomic, readonly) NSTimeInterval playableDuration
```

Discussion
For progressively downloaded network content, this property reflects the amount of content that can be played now.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

playbackState

The current playback state of the movie player. (read-only)

```
@property (nonatomic, readonly) MPMoviePlaybackState playbackState
```

Discussion
The playback state is affected by programmatic calls to play, pause, or stop the movie player. It can also be affected by user interactions or by the network, in cases where streaming content cannot be buffered fast enough.

For a list of valid values for this property, see "`MPMoviePlaybackState`" (page 100).

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

repeatMode

Determines how the movie player repeats the playback of the movie.

```
@property (nonatomic) MPMovieRepeatMode repeatMode
```

Discussion
The default value of this property is `MPMovieRepeatModeNone` (page 101). For a list of available repeat modes, see "MPMovieRepeatMode" (page 101).

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

scalingMode

The scaling mode to use when displaying the movie.

```
@property (nonatomic) MPMovieScalingMode scalingMode
```

Discussion
Changing this property while the movie player is visible causes the current movie to animate to the new scaling mode.

The default value of this property is `MPMovieScalingModeAspectFit`. For a list of available scaling modes, see "MPMovieScalingMode" (page 101).

Availability
Available in iOS 2.0 and later.

Related Sample Code
MoviePlayer

Declared In
`MPMoviePlayerController.h`

shouldAutoplay

A Boolean that indicates whether a movie should begin playback automatically.

```
@property (nonatomic) BOOL shouldAutoplay
```

Discussion
The default value of this property is `YES`. This property determines whether the playback of network-based content begins automatically when there is enough buffered data to ensure uninterrupted playback.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

useApplicationAudioSession

A Boolean value that indicates whether the movie player should use the application's audio session.

```
@property (nonatomic) BOOL useApplicationAudioSession
```

Discussion
The default value of this property is `YES`. Setting this property to `NO` causes the movie player to use a system-supplied audio session with a nonmixable playback category.

> **Important:** In iOS 3.1 and earlier, a movie player always uses a system-supplied audio session. To obtain that same behavior in iOS 3.2 and newer, you must set this property's value to `NO`.

When this property is `YES`, the movie player shares the application's audio session. This give you control over how the movie player content interacts with your audio and with audio from other applications, such as the iPod. For important guidance on using this feature, see "Working with Movies and iPod Music" in *Audio Session Programming Guide*.

Changing the value of this property does not affect the currently playing movie. For the new setting to take effect, you must stop playback and then start it again.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

view

The view containing the movie content and controls. (read-only)

```
@property (nonatomic, readonly) UIView *view
```

Discussion
This property contains the view used for presenting the video content. This view incorporates all the background, content, and controls needed to display movies. You can incorporate this view into your own view hierarchies or present it by itself using a view controller.

To embed the view into your own view hierarchies, add it as a subview to one of your existing views. A good place to do this is in the `loadView` or `viewDidLoad` method of the custom view controller that presents your view hierarchy. You are free to change the view's `frame` rectangle to accommodate the space available in your view hierarchy. The movie player uses the value in the `scalingMode` (page 93) property to scale the movie content to match the frame you specify.

If you want to present the view by itself—that is, without embedding it in an existing view hierarchy—you can use an instance of the `MPMoviePlayerViewController` class to manage the presentation of the view. That class works directly with the movie player controller to present the view by itself.

You can add subviews to the view in this property. You might do this in cases where you want to display custom playback controls or add other custom content that is relevant to your application.

Availability
Available in iOS 3.2 and later.

See Also
`@property backgroundView` (page 87)

Declared In
`MPMoviePlayerController.h`

Instance Methods

cancelAllThumbnailImageRequests

Cancels all pending asynchronous thumbnail image requests.

```
- (void)cancelAllThumbnailImageRequests
```

Discussion
This method cancels only requests made using the `requestThumbnailImagesAtTimes:timeOption:` (page 96) method. It does not cancel requests made synchronously using the `thumbnailImageAtTime:timeOption:` (page 97) method.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

initWithContentURL:

Returns a `MPMoviePlayerController` object initialized with the movie at the specified URL.

```
- (id)initWithContentURL:(NSURL *)url
```

Parameters

url

The location of the movie file. This file must be located either in your application directory or on a remote server.

Return Value
The movie player object.

Discussion
This method initializes the movie player in full-screen mode.

If you provide a `nil` value in the *url* parameter, or call the `init` method directly, the system throws an exception.

To check for errors in URL loading, register for the `MPMoviePlayerContentPreloadDidFinishNotification` (page 106) or `MPMoviePlayerPlaybackDidFinishNotification` (page 107) notifications. On error, these notifications contain an `NSError` object available using the `@"error"` key in the notification's `userInfo` dictionary.

Availability
Available in iOS 2.0 and later.

Related Sample Code
MoviePlayer

Declared In
`MPMoviePlayerController.h`

requestThumbnailImagesAtTimes:timeOption:

Captures one or more thumbnail images asynchronously from the current movie.

```
- (void)requestThumbnailImagesAtTimes:(NSArray *)playbackTimes
    timeOption:(MPMovieTimeOption)option
```

Parameters

playbackTimes

An array of `NSNumber` objects containing the times at which to capture the thumbnail images. Each time value represents the number of seconds from the beginning of the current movie.

option

The option to use when determining which specific frame to use for each thumbnail image. For a list of possible values, see "MPMovieTimeOption" (page 102).

Discussion
This method processes each thumbnail request separately and asynchronously. When the results for a single image arrive, the movie player posts a `MPMoviePlayerThumbnailImageRequestDidFinishNotification` (page 108) notification with the results for that image. Notifications are posted regardless of whether the image capture was successful or failed. You should register for this notification prior to calling this method.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

setFullscreen:animated:

Causes the movie player to enter or exit full-screen mode.

```
- (void)setFullscreen:(BOOL)fullscreen animated:(BOOL)animated
```

Parameters

fullscreen

Specify `YES` to enter full-screen mode or `NO` to exit full-screen mode.

animated

Specify `YES` to animate the transition between modes or `NO` to switch immediately to the new mode.

Availability
Available in iOS 3.2 and later.

See Also
`@property fullscreen` (page 89)

Declared In
`MPMoviePlayerController.h`

thumbnailImageAtTime:timeOption:

Captures and returns a thumbnail image from the current movie.

```
- (UIImage *)thumbnailImageAtTime:(NSTimeInterval)playbackTime
    timeOption:(MPMovieTimeOption)option
```

Parameters

playbackTime

The time at which to capture the thumbnail image. The time value represents the number of seconds from the beginning of the current movie.

option

The option to use when determining which specific frame to use for the thumbnail image. For a list of possible values, see "MPMovieTimeOption" (page 102).

Return Value
An image object containing the image from the movie or `nil` if the thumbnail could not be captured.

Discussion
This method captures the thumbnail image synchronously from the current movie (which is accessible from the `MPMovieSourceTypeUnknown` (page 103) property).

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

timedMetadata

Obtains the most recent time-based metadata provided by the streamed movie.

```
- (NSArray *) timedMetadata;
```

Return Value
An array of the most recent `MPTimedMetadata` objects provided by the streamed movie.

Availability
Available in iOS 4.0 and later.

Declared In
`MPMoviePlayerController.h`

Constants

MPMovieLoadState

Constants describing the network load state of the movie player.

```
enum {
    MPMovieLoadStateUnknown        = 0,
    MPMovieLoadStatePlayable       = 1 << 0,
    MPMovieLoadStatePlaythroughOK  = 1 << 1,
    MPMovieLoadStateStalled        = 1 << 2,
};
typedef NSInteger MPMovieLoadState;
```

Constants

`MPMovieLoadStateUnknown`

The load state is not known at this time.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieLoadStatePlayable`

The buffer has enough data that playback can begin, but it may run out of data before playback finishes.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieLoadStatePlaythroughOK`

Enough data has been buffered for playback to continue uninterrupted.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieLoadStateStalled`

The buffering of data has stalled. If started now, playback may pause automatically if the player runs out of buffered data.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieControlStyle

Constants describing the style of the playback controls.

```
enum {
    MPMovieControlStyleNone,
    MPMovieControlStyleEmbedded,
    MPMovieControlStyleFullscreen,
    MPMovieControlStyleDefault = MPMovieControlStyleFullscreen
};
typedef NSInteger MPMovieControlStyle;
```

Constants

`MPMovieControlStyleNone`

No controls are displayed.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlStyleEmbedded`

Controls for an embedded view are displayed. The controls include a start/pause button, a scrubber bar, and a button for toggling between fullscreen and embedded display modes.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlStyleFullscreen`

Controls for fullscreen playback are displayed. The controls include a start/pause button, a scrubber bar, forward and reverse seeking buttons, a button for toggling between fullscreen and embedded display modes, a button for toggling the aspect fill mode, and a Done button. Tapping the done button pauses the video and exits fullscreen mode.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlStyleDefault`

Fullscreen controls are displayed by default.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieFinishReason

Constants describing the reason that playback ended.

```
enum {
    MPMovieFinishReasonPlaybackEnded,
    MPMovieFinishReasonPlaybackError,
    MPMovieFinishReasonUserExited
};
typedef NSInteger MPMovieFinishReason;
```

Constants

`MPMovieFinishReasonPlaybackEnded`

The end of the movie was reached.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieFinishReasonPlaybackError`
There was an error during playback.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieFinishReasonUserExited`
The user stopped playback.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMoviePlaybackState

Constants describing the current playback state of the movie player.

```
enum {
    MPMoviePlaybackStateStopped,
    MPMoviePlaybackStatePlaying,
    MPMoviePlaybackStatePaused,
    MPMoviePlaybackStateInterrupted,
    MPMoviePlaybackStateSeekingForward,
    MPMoviePlaybackStateSeekingBackward
};
typedef NSInteger MPMoviePlaybackState;
```

Constants

`MPMoviePlaybackStateStopped`
Playback is currently stopped. Playback will commence from the beginning of the movie.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStatePlaying`
Playback is currently under way.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStatePaused`
Playback is currently paused. Playback will resume from the point where it was paused.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStateInterrupted`
Playback is temporarily interrupted, perhaps because the buffer ran out of content.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStateSeekingForward`
The movie player is currently seeking towards the end of the movie.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStateSeekingBackward`
The movie player is currently seeking towards the beginning of the movie.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieRepeatMode

Constants describing how the movie player repeats content at the end of playback.

```
enum {
    MPMovieRepeatModeNone,
    MPMovieRepeatModeOne
};
typedef NSInteger MPMovieRepeatMode;
```

Constants

`MPMovieRepeatModeNone`
Content is not repeated when playback finishes

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieRepeatModeOne`
The current movie is repeated when it finishes.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieScalingMode

Constants describing how the movie content is scaled to fit the frame of its view.

```
typedef enum {
    MPMovieScalingModeNone,
    MPMovieScalingModeAspectFit,
    MPMovieScalingModeAspectFill,
    MPMovieScalingModeFill
} MPMovieScalingMode;
```

Constants

`MPMovieScalingModeNone`
Do not scale the movie.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieScalingModeAspectFit`
Scale the movie uniformly until one dimension fits the visible bounds of the view exactly. In the other dimension, the region between the edge of the movie and the edge of the view is filled with a black bar. The aspect ratio of the movie is preserved.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieScalingModeAspectFill`
: Scale the movie uniformly until the movie fills the visible bounds of the view. Content at the edges of the larger of the two dimensions is clipped so that the other dimension fits the view exactly. The aspect ratio of the movie is preserved.

  Available in iOS 2.0 and later.

  Declared in `MPMoviePlayerController.h`.

`MPMovieScalingModeFill`
: Scale the movie until both dimensions fit the visible bounds of the view exactly. The aspect ratio of the movie is not preserved.

  Available in iOS 2.0 and later.

  Declared in `MPMoviePlayerController.h`.

MPMovieTimeOption

Constants describing which frame to use when generating thumbnail images.

```
enum {
    MPMovieTimeOptionNearestKeyFrame,
    MPMovieTimeOptionExact
};
typedef NSInteger MPMovieTimeOption;
```

Constants

`MPMovieTimeOptionNearestKeyFrame`
: Generate a thumbnail image using the nearest key frame. This frame could be several frames away from the current frame. This option generally offers better performance than trying to find the exact frame.

  Available in iOS 3.2 and later.

  Declared in `MPMoviePlayerController.h`.

`MPMovieTimeOptionExact`
: Use the exact current frame.

  Available in iOS 3.2 and later.

  Declared in `MPMoviePlayerController.h`.

MPMovieMediaTypeMask

Specifies the types of content available in the movie file.

```
enum {
    MPMovieMediaTypeMaskNone  = 0,
    MPMovieMediaTypeMaskVideo = 1 << 0,
    MPMovieMediaTypeMaskAudio = 1 << 1
};
typedef NSInteger MPMovieMediaTypeMask;
```

Constants

`MPMovieMediaTypeMaskNone`

The types of media available in the media are not yet known.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieMediaTypeMaskVideo`

The movie file contains video media.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieMediaTypeMaskAudio`

The movie file contains audio media.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

Discussion

You can OR the specified constants together to specify a movie

MPMovieSourceType

Specifies the type of the movie file.

```
enum {
    MPMovieSourceTypeUnknown,
    MPMovieSourceTypeFile,
    MPMovieSourceTypeStreaming
};
typedef NSInteger MPMovieSourceType;
```

Constants

`MPMovieSourceTypeUnknown`

The movie type is not yet known.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieSourceTypeFile`

The movie is a local file or is a file that can be downloaded from the network.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieSourceTypeStreaming`

The movie is a live or on-demand stream.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

Thumbnail Notification User Info Keys

The following keys may be found in the *userInfo* dictionary of a MPMoviePlayerThumbnailImageRequestDidFinishNotification (page 108) notification.

```
NSString *const MPMoviePlayerThumbnailImageKey;
NSString *const MPMoviePlayerThumbnailTimeKey;
NSString *const MPMoviePlayerThumbnailErrorKey;
```

Constants

`MPMoviePlayerThumbnailImageKey`

The value of this key is a `UIImage` object containing the image that was obtained for the desired frame.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerThumbnailTimeKey`

The value of this key is a `NSNumber` object containing a double value. This value represents the actual time (measured in seconds) from the beginning of the movie at which the image was captured.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerThumbnailErrorKey`

The value of this key is an `NSError` object identifying the error that occurred, if any.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

Fullscreen Notification Keys

The following keys may be found in the *userInfo* dictionary of notifications for transitioning in or out of full-screen mode.

```
NSString *const MPMoviePlayerFullscreenAnimationDurationUserInfoKey;
NSString *const MPMoviePlayerFullscreenAnimationCurveUserInfoKey;
```

Constants

`MPMoviePlayerFullscreenAnimationDurationUserInfoKey`

The value of this key is an `NSNumber` containing a double value. This value represents the duration (measured in seconds) of the animation used to transition in or out of full-screen mode.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerFullscreenAnimationCurveUserInfoKey`

The value of this key is an `NSNumber` containing an integer value that represents one of the `UIViewAnimationCurve` constants.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

Playback Finished Notification Key

The following key may be found in the userInfo dictionary of a `MPMoviePlayerPlaybackDidFinishNotification` (page 107) notification.

```
NSString *const MPMoviePlayerPlaybackDidFinishReasonUserInfoKey;
```

Constants

`MPMoviePlayerPlaybackDidFinishReasonUserInfoKey`

The value of this key is an `NSNumber` containing an integer value that represents one of the `"MPMovieFinishReason"` (page 99) constants.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieControlMode

Options for displaying movie playback controls. (Deprecated. Use the `"MPMovieControlStyle"` (page 98) constants in conjunction with the `controlStyle` (page 88) property instead.)

```
typedef enum {
    MPMovieControlModeDefault,
    MPMovieControlModeVolumeOnly,
    MPMovieControlModeHidden
} MPMovieControlMode;
```

Constants

`MPMovieControlModeDefault`

Display the standard controls for controlling playback. This includes play/pause controls, a volume slider, and a timeline control.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlModeVolumeOnly`

Display volume controls only.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlModeHidden`

Do not display any controls. This mode prevents the user from controlling playback.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

Notifications

MPMovieDurationAvailableNotification

This notification is posted when the duration of a movie object is determined. The object of the notification is the `MPMoviePlayerController` object itself. There is no *`userInfo`* dictionary. The duration value is reflected in the `duration` (page 88) property of the movie player controller.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMovieMediaTypesAvailableNotification

This notification is posted when the media types of a movie object are determined. The object of the notification is the `MPMoviePlayerController` object itself. There is no *`userInfo`* dictionary. The supported media types are reflected in the `movieMediaTypes` (page 91) property of the movie player controller.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMovieNaturalSizeAvailableNotification

This notification is posted when the frame size of a movie object is first determined or subsequently changes. The object of the notification is the `MPMoviePlayerController` object itself. There is no *`userInfo`* dictionary. The frame size value is reflected in the `naturalSize` (page 91) property of the movie player controller.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerContentPreloadDidFinishNotification

Notifies observers that the movie is now in memory and ready to play. The affected movie player is stored in the `object` parameter of the notification. If an error occurred during loading, the `userInfo` dictionary of this notification contains a key with the name “error” whose value is the `NSError` object describing the problem. (#Deprecated. Use the `MPMoviePlayerLoadStateDidChangeNotification` (page 107) notification to determine the readiness of the player.)

Availability
Available in iOS 2.0 and later.
Deprecated in iOS 3.2.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerDidEnterFullscreenNotification

Notifies observers that the movie player entered into full-screen mode. The affected movie player is stored in the `object` parameter of the notification. There is no *`userInfo`* dictionary.

User actions may also cause the media player to send this notification.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerDidExitFullscreenNotification

Notifies observers that the movie player exited full-screen mode. The affected movie player is stored in the `object` parameter of the notification. There is no *`userInfo`* dictionary.

User actions may also cause the media player to send this notification.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerLoadStateDidChangeNotification

Notifies observers that the network buffering state changed. The affected movie player is stored in the `object` parameter of the notification. There is no `userInfo` dictionary. The current load state can be retrieved from the `loadState` (page 90) property of the movie player controller.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerNowPlayingMovieDidChangeNotification

Notifies observers that the currently playing movie changed. The affected movie player is stored in the `object` parameter of the notification. There is no `userInfo` dictionary. The currently playing movie can be retrieved from the `contentURL` (page ?) property of the movie player controller.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerPlaybackDidFinishNotification

Notifies observers that the movie finished playing. The affected movie player is stored in the `object` parameter of the notification. The `userInfo` dictionary of this notification contains the `MPMoviePlayerPlaybackDidFinishReasonUserInfoKey` (page 105) key, which indicates the reason that playback finished. This notification is also sent when playback fails because of an error.

This notification is not sent in cases where the movie player is displaying in fullscreen mode and the user taps the Done button. In that instance, the Done button causes movie playback to pause while the player transitions out of fullscreen mode. If you want to detect this scenario in your code, you should monitor other notifications such as `MPMoviePlayerDidExitFullscreenNotification` (page 107).

Availability
Available in iOS 2.0 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerPlaybackStateDidChangeNotification

Notifies observers that the playback state changed. The affected movie player is stored in the `object` parameter of the notification. There is no `userInfo` dictionary.

The playback state can change by programmatic means or because of user interactions with the controls. To get the current playback state, get the value of the `playbackState` (page 92) property of the movie player object.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerScalingModeDidChangeNotification

Notifies observers that the scaling mode property of the player changed. The affected movie player is stored in the `object` parameter of the notification. There is no `userInfo` dictionary.

User actions may also cause the media player to send this notification.

Availability
Available in iOS 2.0 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerThumbnailImageRequestDidFinishNotification

Notifies observers that a request to capture a thumbnail from the movie is now complete. The affected movie player is stored in the `object` parameter of the notification. The *`userInfo`* dictionary of this notification contains one or more keys with information about the thumbnail image.

A separate notification is sent for each thumbnail that is captured. Upon successful capture of a given image, the *`userInfo`* dictionary contains the `MPMoviePlayerThumbnailImageKey` (page 104) and `MPMoviePlayerThumbnailTimeKey` (page 104) keys. If an error occurs, the notification contains the `MPMoviePlayerThumbnailErrorKey` (page 104) and `MPMoviePlayerThumbnailTimeKey` keys.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerWillEnterFullscreenNotification

Notifies observers that the movie player is about to enter full-screen mode. The affected movie player is stored in the `object` parameter of the notification. The *`userInfo`* dictionary of this notification contains keys describing the transition animation used to enter full-screen mode. These keys are described in "Fullscreen Notification Keys" (page 104).

User actions may also cause the media player to send this notification.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerWillExitFullscreenNotification

Notifies observers that the movie player is about to exit full-screen mode. The affected movie player is stored in the `object` parameter of the notification. The *`userInfo`* dictionary of this notification contains keys describing the transition animation used to exit full-screen mode. These keys are described in "Fullscreen Notification Keys" (page 104).

User actions may also cause the media player to send this notification.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMovieSourceTypeAvailableNotification

This notification is posted when the source type of a movie object is unknown initially but is determined later. The object of the notification is the `MPMoviePlayerController` object itself. There is no *`userInfo`* dictionary. The source type is reflected in the `movieSourceType` (page 91) property of the movie player controller.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerViewController Class Reference

Inherits from	UIViewController : UIResponder : NSObject
Conforms to	NSCoding (UIViewController) NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.2 and later.
Declared in	

Overview

The `MPMoviePlayerViewController` class implements a simple view controller for displaying full-screen movies. Unlike using an `MPMoviePlayerController` object on its own to present a movie immediately, you can incorporate a movie player view controller wherever you would normally use a view controller. For example, you can present it using a tab bar or navigation bar-based interface, taking advantage of the transitions offered by those interfaces.

To present a movie player view controller modally, you typically use the `presentMoviePlayerViewControllerAnimated:` (page 140) method. This method is part of a category on the `UIViewController` class and is implemented by the Media Player framework. The `presentMoviePlayerViewControllerAnimated:` method presents a movie player view controller using the standard transition animations for presenting video content. To dismiss a modally presented movie player view controller, call the `dismissMoviePlayerViewControllerAnimated` (page 139) method.

Tasks

New Methods

`- initWithContentURL:` (page 112)
Returns a movie player view controller initialized with the specified movie.

`moviePlayer` (page 112) *property*
The movie player controller object used to present the movie. (read-only)

`- shouldAutorotateToInterfaceOrientation:` (page 112)
Returns a Boolean value indicating whether the view controller supports the specified orientation.

Properties

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

moviePlayer

The movie player controller object used to present the movie. (read-only)

```
@property(nonatomic, readonly) MPMoviePlayerController *moviePlayer
```

Discussion
The `MPMoviePlayerController` object in this property is created automatically by the receiver and cannot be changed. However, you can use the object to manage the presentation and configuration of the movie playback.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerViewController.h`

Instance Methods

initWithContentURL:

Returns a movie player view controller initialized with the specified movie.

```
- (id)initWithContentURL:(NSURL *)contentURL
```

Parameters
`contentURL`
The URL that points to the content to be played.

Return Value
A movie player view controller initialized with the specified URL.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerViewController.h`

shouldAutorotateToInterfaceOrientation:

Returns a Boolean value indicating whether the view controller supports the specified orientation.

```
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
```

Parameters

toInterfaceOrientation

The orientation of the application's user interface after the rotation. The possible values are described in `UIInterfaceOrientation`.

Return Value

`YES` if the view controller supports the specified orientation or `NO` if it does not.

Discussion

This method is an override that replaces the default behavior by returning `YES` for the `UIInterfaceOrientationPortrait`, `UIInterfaceOrientationLandscapeLeft`, and `UIInterfaceOrientationLandscapeRight` orientations.

Availability

Available in iOS 3.2 and later.

Declared In

`MPMoviePlayerViewController.h`

MPMusicPlayerController Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide
Related sample code	AddMusic

Overview

Use an `MPMusicPlayerController` object, or music player, to play media items from the device iPod library. There are two types of music player:

- The *application music player* plays music locally within your application. It is not aware of the iPod app's now-playing item, nor does it affect the iPod state.
- The *iPod music player* employs the built-in iPod app on your behalf. On instantiation, it takes on the current iPod application state, such as the identification of the now-playing item. If a user switches away from your app while music is playing, that music continues to play. The iPod app then has your music player's most recently-set repeat mode, shuffle mode, playback state, and now-playing item.

Important: You must use a music player only on your app's main thread.

Home Sharing and iPod Music Players

Starting in iOS 4, the built-in iPod and Videos apps can play media from shared libraries using Home Sharing. However, third-party apps using the Media Player framework still have access only to the device iPod library. This means that your app cannot display the title of a home-shared song in your user interface. Specifically, if the iPod app is playing a home shared song, and you are using an iPod music player, the value of the `nowPlayingItem` (page 118) property of your music player is `nil`. Other playback information is available, however, when playing shared media. For example, the system does update the values of the `playbackState` (page 119) and `currentPlaybackTime` (page 118) properties when an iPod music player is playing a shared item.

Tasks

Getting a Music Player

`+ applicationMusicPlayer` (page 120)
Returns the application music player.

`+ iPodMusicPlayer` (page 121)
Returns the iPod music player, which controls the iPod application's state.

Setting Up a Playback Queue

Before a music player can produce sound, it needs a playback queue. You provide the music player with its playback queue, which specifies the media items to be played.

`- setQueueWithQuery:` (page 124)
Sets a music player's playback queue based on a media query.

`- setQueueWithItemCollection:` (page 124)
Sets a music player's playback queue using a media item collection.

Managing Playback Mode and State

`currentPlaybackTime` (page 118) *property*
The current playback time, in seconds, measured from the start of the now-playing media item's timeline.

`nowPlayingItem` (page 118) *property*
The currently-playing media item, or the media item, within a queue, that you have designated to begin playback with.

`playbackState` (page 119) *property*
The current playback state of the music player.

`repeatMode` (page 119) *property*
The current repeat mode of the music player.

`shuffleMode` (page 119) *property*
The current shuffle mode of the music player.

`volume` (page 120) *property*
The audio playback volume for the music player, in the range from 0.0 (silent) through 1.0 (maximum volume).

Controlling Playback

- `play` (page 123)
  Plays media items from the current playback queue, resuming paused playback if possible.
- `pause` (page 123)
  Pauses playback if the music player is playing.
- `stop` (page 126)
  Ends playback.
- `beginSeekingForward` (page 122)
  Moves the playback point forward in the media item (for example, toward the end of a song) faster than the normal playback rate.
- `beginSeekingBackward` (page 122)
  Moves the playback point backward in the media item (for example, toward the start of a song).
- `endSeeking` (page 123)
  Stops additional movement of the playback point, returning the playback state to what it was prior to seeking.
- `skipToNextItem` (page 125)
  Starts playback of the next media item in the playback queue; or, the music player is not playing, designates the next media item as the next to be played.
- `skipToPreviousItem` (page 125)
  Starts playback of the previous media item in the playback queue; or, the music player is not playing, designates the previous media item as the next to be played.
- `skipToBeginning` (page 125) Available in iOS 2.0 through iOS 3.1
  Restarts playback at the beginning of the currently playing media item.

Using Music Player Notifications

These methods control the posting of playback notifications. You can nest calls to start or end these notifications.

- `beginGeneratingPlaybackNotifications` (page 121)
  Starts the generation of playback notifications.
- `endGeneratingPlaybackNotifications` (page 122)
  Ends the generation of playback notifications.

Properties

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

currentPlaybackTime

The current playback time, in seconds, measured from the start of the now-playing media item's timeline.

```
@property (nonatomic) NSTimeInterval currentPlaybackTime
```

Discussion
Because this property's access is read/write, you can use it to:

- Acquire data to present current playback time to the user
- Seek to a particular point in a media item

If a music player is configured as an iPod music player and the built-in iPod app is using a home shared library, the value of this property is `nil` and you cannot change it.

Availability
Available in iOS 3.0 and later.

See Also
`@property nowPlayingItem` (page 118)

Related Sample Code
AddMusic

Declared In
`MPMusicPlayerController.h`

nowPlayingItem

The currently-playing media item, or the media item, within a queue, that you have designated to begin playback with.

```
@property (nonatomic, copy) MPMediaItem *nowPlayingItem
```

Discussion
To specify that playback should begin at a particular media item in the playback queue, set this property to that item while the music player is stopped or paused.

If no media item is playing or designated to play, this property's value is `nil`.

If you create an iPod music player and the user plays an item from another library using Home Sharing, the value of this property is `nil`.

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
`MPMusicPlayerController.h`

playbackState

The current playback state of the music player.

```
@property (nonatomic, readonly) MPMusicPlaybackState playbackState
```

Discussion
If you configure a music player as an iPod music player, the playback state matches the playback state of the built-in iPod app. This is true whether the iPod app is using the device iPod library or a home shared library. Note, however, that when the iPod is using a home shared library, the music player's `nowPlayingItem` (page 118) property is `nil`.

For the available playback states, see "Playback States" (page 126).

Availability
Available in iOS 3.0 and later.

See Also
`@property repeatMode` (page 119)
`@property shuffleMode` (page 119)

Related Sample Code
AddMusic

Declared In
`MPMusicPlayerController.h`

repeatMode

The current repeat mode of the music player.

```
@property (nonatomic) MPMusicRepeatMode repeatMode
```

Discussion
For the available repeat modes, see "Repeat Modes" (page 127). If not explicitly set, `repeatMode` defaults to `MPMusicRepeatModeDefault` (page 128).

Availability
Available in iOS 3.0 and later.

See Also
`@property playbackState` (page 119)
`@property shuffleMode` (page 119)

Declared In
`MPMusicPlayerController.h`

shuffleMode

The current shuffle mode of the music player.

```
@property (nonatomic) MPMusicShuffleMode shuffleMode
```

Discussion
For the available shuffle modes, see "Shuffle Modes" (page 128). If not explicitly set, `shuffleMode` defaults to `MPMusicShuffleModeDefault` (page 128).

Availability
Available in iOS 3.0 and later.

See Also
`@property playbackState` (page 119)
`@property repeatMode` (page 119)

Declared In
`MPMusicPlayerController.h`

volume

The audio playback volume for the music player, in the range from 0.0 (silent) through 1.0 (maximum volume).

```
@property (nonatomic) float volume
```

Discussion
A music player's volume has a default value of 1.0.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMusicPlayerController.h`

Class Methods

applicationMusicPlayer

Returns the application music player.

```
+ (MPMusicPlayerController *)applicationMusicPlayer
```

Return Value
The application music player.

Discussion
The application music player plays music locally within your application. It does not affect the iPod state. When your app moves to the background, the music player stops if it was playing.

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
MPMusicPlayerController.h

iPodMusicPlayer

Returns the iPod music player, which controls the iPod application's state.

```
+ (MPMusicPlayerController *)iPodMusicPlayer
```

Return Value
The iPod music player.

Discussion
The iPod music player employs the iPod application on your behalf. On instantiation, it takes on the current iPod application state and controls that state as your application runs. Specifically, the shared state includes the following:

- Repeat mode (see "Repeat Modes" (page 127))
- Shuffle mode (see "Shuffle Modes" (page 128)
- Now-playing item (see `nowPlayingItem` (page 118))
- Playback state (see `playbackState` (page 119))

Other aspects of iPod state, such as the on-the-go playlist, are not shared. Music that is playing continues to play when your app moves to the background.

Availability
Available in iOS 3.0 and later.

Related Sample Code
AddMusic

Declared In
MPMusicPlayerController.h

Instance Methods

beginGeneratingPlaybackNotifications

Starts the generation of playback notifications.

```
- (void)beginGeneratingPlaybackNotifications
```

Availability
Available in iOS 3.0 and later.

See Also
`MPMusicPlayerControllerPlaybackStateDidChangeNotification` (page 129)
`MPMusicPlayerControllerNowPlayingItemDidChangeNotification` (page 129)

Declared In
MPMusicPlayerController.h

beginSeekingBackward

Moves the playback point backward in the media item (for example, toward the start of a song).

```
- (void)beginSeekingBackward
```

Discussion
Seeking rate increases while seeking is active.

Availability
Available in iOS 3.0 and later.

See Also
- beginSeekingForward (page 122)
- endSeeking (page 123)

Declared In
MPMusicPlayerController.h

beginSeekingForward

Moves the playback point forward in the media item (for example, toward the end of a song) faster than the normal playback rate.

```
- (void)beginSeekingForward
```

Discussion
Seeking rate increases while seeking is active.

Availability
Available in iOS 3.0 and later.

See Also
- beginSeekingBackward (page 122)
- endSeeking (page 123)

Declared In
MPMusicPlayerController.h

endGeneratingPlaybackNotifications

Ends the generation of playback notifications.

```
- (void)endGeneratingPlaybackNotifications
```

Availability
Available in iOS 3.0 and later.

See Also
MPMusicPlayerControllerPlaybackStateDidChangeNotification (page 129)
MPMusicPlayerControllerNowPlayingItemDidChangeNotification (page 129)

Declared In
MPMusicPlayerController.h

endSeeking

Stops additional movement of the playback point, returning the playback state to what it was prior to seeking.

```
- (void)endSeeking
```

Availability
Available in iOS 3.0 and later.

See Also
- beginSeekingForward (page 122)
- beginSeekingBackward (page 122)

Declared In
MPMusicPlayerController.h

pause

Pauses playback if the music player is playing.

```
- (void)pause
```

Discussion
Calling play (page 123) again starts playback from the spot where playback was paused.

Availability
Available in iOS 3.0 and later.

See Also
- play (page 123)
- stop (page 126)

Related Sample Code
AddMusic

Declared In
MPMusicPlayerController.h

play

Plays media items from the current playback queue, resuming paused playback if possible.

```
- (void)play
```

Availability
Available in iOS 3.0 and later.

See Also
- `pause` (page 123)
- `stop` (page 126)

Related Sample Code
AddMusic

Declared In
`MPMusicPlayerController.h`

setQueueWithItemCollection:

Sets a music player's playback queue using a media item collection.

```
- (void)setQueueWithItemCollection:(MPMediaItemCollection *)itemCollection
```

Parameters

itemCollection

A media item collection that you want as the playback queue. See *MPMediaItemCollection Class Reference* for a description of media item collections and how to use them.

Discussion
To begin playback after establishing a playback queue, call `play` (page 123).

Availability
Available in iOS 3.0 and later.

See Also
- `setQueueWithQuery:` (page 124)

Declared In
`MPMusicPlayerController.h`

setQueueWithQuery:

Sets a music player's playback queue based on a media query.

```
- (void)setQueueWithQuery:(MPMediaQuery *)query
```

Parameters

query

A media query that specifies the collection of media items that you want as the playback queue. See *MPMediaQuery Class Reference* for a description of query types and how to create them.

Discussion
To begin playback after establishing a playback queue, call `play` (page 123).

Availability
Available in iOS 3.0 and later.

See Also
- `setQueueWithItemCollection:` (page 124)

Declared In
`MPMusicPlayerController.h`

skipToBeginning

Restarts playback at the beginning of the currently playing media item.

```
- (void)skipToBeginning
```

Availability
Available in iOS 3.0 and later.

See Also
- `skipToNextItem` (page 125)
- `skipToPreviousItem` (page 125)

Declared In
`MPMusicPlayerController.h`

skipToNextItem

Starts playback of the next media item in the playback queue; or, the music player is not playing, designates the next media item as the next to be played.

```
- (void)skipToNextItem
```

Discussion
If already at the last item in the playback queue when this method is called, ends playback.

Availability
Available in iOS 3.0 and later.

See Also
- `@property nowPlayingItem` (page 118)
- `skipToBeginning` (page 125)
- `skipToPreviousItem` (page 125)

Declared In
`MPMusicPlayerController.h`

skipToPreviousItem

Starts playback of the previous media item in the playback queue; or, the music player is not playing, designates the previous media item as the next to be played.

```
- (void)skipToPreviousItem
```

Discussion
If already at the first item in the playback queue when this method is called, ends playback.

Availability
Available in iOS 3.0 and later.

See Also
`@property nowPlayingItem` (page 118)
- `skipToBeginning` (page 125)
- `skipToNextItem` (page 125)

Declared In
`MPMusicPlayerController.h`

stop

Ends playback.

```
- (void)stop
```

Discussion
Calling `play` (page 123) again starts playback from the beginning of the queue.

Availability
Available in iOS 3.0 and later.

See Also
- `play` (page 123)
- `stop` (page 126)

Declared In
`MPMusicPlayerController.h`

Constants

Playback States

Values for the `playbackState` (page 119) property.

```
enum {
    MPMusicPlaybackStateStopped,
    MPMusicPlaybackStatePlaying,
    MPMusicPlaybackStatePaused,
    MPMusicPlaybackStateInterrupted,
    MPMusicPlaybackStateSeekingForward,
    MPMusicPlaybackStateSeekingBackward
};
typedef NSInteger MPMusicPlaybackState;
```

Constants

`MPMusicPlaybackStateStopped`
The music player is stopped.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicPlaybackStatePlaying`
The music player is playing.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicPlaybackStatePaused`
The music player is paused.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicPlaybackStateInterrupted`
The music player has been interrupted, such as by an incoming phone call.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicPlaybackStateSeekingForward`
The music player is seeking forward.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicPlaybackStateSeekingBackward`
The music player is seeking backward.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

Discussion

You determine a music player's state by checking the `playbackState` (page 119) property. Depending on the property's value, you can update your application's user interface or take other appropriate action.

Repeat Modes

Values for the `repeatMode` (page 119) property.

```
enum {
    MPMusicRepeatModeDefault,
    MPMusicRepeatModeNone,
    MPMusicRepeatModeOne,
    MPMusicRepeatModeAll
};
typedef NSInteger MPMusicRepeatMode;
```

Constants

`MPMusicRepeatModeDefault`
The user's preferred repeat mode.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicRepeatModeNone`
The music player will not repeat the current song or playlist.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicRepeatModeOne`
The music player will repeat the current song.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicRepeatModeAll`
The music player will repeat the current playlist.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

Shuffle Modes

Values for the `shuffleMode` (page 119) property.

```
enum {
    MPMusicShuffleModeDefault,
    MPMusicShuffleModeOff,
    MPMusicShuffleModeSongs,
    MPMusicShuffleModeAlbums
};
typedef NSInteger MPMusicShuffleMode;
```

Constants

`MPMusicShuffleModeDefault`
The user's preferred shuffle mode.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicShuffleModeOff`
The playlist is not shuffled.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicShuffleModeSongs`
The playlist is shuffled by song.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

`MPMusicShuffleModeAlbums`
The playlist is shuffled by album.
Available in iOS 3.0 and later.
Declared in `MPMusicPlayerController.h`.

Notifications

MPMusicPlayerControllerPlaybackStateDidChangeNotification

Posted when the playback state has been changed programmatically or by user action. The object associated with the notification is the music player whose playback state changed. The notification has no `userInfo` dictionary.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMusicPlayerController.h`

MPMusicPlayerControllerNowPlayingItemDidChangeNotification

Posted when the currently playing media item has changed. The object associated with the notification is the music player whose now-playing item changed. The notification has no `userInfo` dictionary.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMusicPlayerController.h`

MPMusicPlayerControllerVolumeDidChangeNotification

Posted when the audio playback volume for the music player has changed. The object associated with the notification is the music player whose playback volume changed. The notification has no `userInfo` dictionary.

Availability
Available in iOS 3.0 and later.

Declared In
`MPMusicPlayerController.h`

MPTimedMetadata Class Reference

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

Overview

An instance of the `MPTimedMetadata` class, called a *timed metadata object*, carries time-based information within HTTP streamed media. Content providers can embed such objects when creating a stream. The properties and constants in this class let you extract the metadata as you play the stream using an `MPMoviePlayerController` object.

For example, the provider of a live sports video stream could use `MPTimedMetadata` instances to embed game scores, with timestamps, in the stream. On the client side—that is, on the user's device—their application could employ the properties of this class to update their app's user interface in real time during the game.

A Javascript implementation of this class is also available for use by web-based applications.

Tasks

Extracting Timed Metadata from a Stream

`allMetadata` (page 132) *property*
A dictionary containing all the metadata in the object. (read-only)

`key` (page 132) *property*
A key that identifies a piece of timed metadata. (read-only)

`keyspace` (page 132) *property*
The namespace of the identifying key. (read-only)

`timestamp` (page 132) *property*
The timestamp of the metadata, in the timebase of the media stream. (read-only)

`value` (page 133) *property*
The timed metadata. (read-only)

Properties

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

allMetadata

A dictionary containing all the metadata in the object. (read-only)

```
@property (nonatomic, readonly) NSDictionary *allMetadata
```

Discussion
To retrieve metadata from the dictionary, use the keys described in "Timed Metadata Dictionary Keys" (page 133).

Availability
Available in iOS 4.0 and later.

Declared In
`MPMoviePlayerController.h`

key

A key that identifies a piece of timed metadata. (read-only)

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

Availability
Available in iOS 4.0 and later.

Declared In
`MPMoviePlayerController.h`

keyspace

The namespace of the identifying key. (read-only)

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

Availability
Available in iOS 4.0 and later.

Declared In
`MPMoviePlayerController.h`

timestamp

The timestamp of the metadata, in the timebase of the media stream. (read-only)

```
@property (nonatomic, readonly) NSTimeInterval timestamp
```

Availability
Available in iOS 4.0 and later.

Declared In
`MPMoviePlayerController.h`

value

The timed metadata. (read-only)

```
@property (nonatomic, readonly) id value
```

Availability
Available in iOS 4.0 and later.

Declared In
`MPMoviePlayerController.h`

Constants

Timed Metadata Dictionary Keys

Dictionary keys for use with the `allMetadata` (page 132) property. All keys are optional.

```
NSString *const MPMoviePlayerTimedMetadataKeyName;
NSString *const MPMoviePlayerTimedMetadataKeyInfo;
NSString *const MPMoviePlayerTimedMetadataKeyMIMEType;
NSString *const MPMoviePlayerTimedMetadataKeyDataType;
NSString *const MPMoviePlayerTimedMetadataKeyLanguageCode;
```

Constants

`MPMoviePlayerTimedMetadataKeyName`
The name of the timed metadata key.
Available in iOS 4.0 and later.
Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerTimedMetadataKeyInfo`
Arbitrary information about the timed metadata.
Available in iOS 4.0 and later.
Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerTimedMetadataKeyMIMEType`
The MIME type for the timed metadata.
Available in iOS 4.0 and later.
Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerTimedMetadataKeyDataType`
The data type of the timed metadata.

Available in iOS 4.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerTimedMetadataKeyLanguageCode`
The metadata language, expressed using ISO 639-2, in a string object.

Available in iOS 4.0 and later.

Declared in `MPMoviePlayerController.h`.

Notifications

MPMoviePlayerTimedMetadataUpdatedNotification

Posted when new timed metadata arrives.

Availability
Available in iOS 4.0 and later.

Declared In
`MPMoviePlayerController.h`

MPMoviePlayerTimedMetadataUserInfoKey

An NSArray object containing the most recent `MPTimedMetadata` objects.

Availability
Available in iOS 4.0 and later.

Declared In
`MPMoviePlayerController.h`

MPVolumeView Class Reference

Inherits from	UIView : UIResponder : NSObject
Conforms to	NSCoding NSCoding (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 2.0 and later.
Declared in	

Overview

Use a volume view to present the user with a slider control for setting the system audio output volume, and a button for choosing the audio output route. When first displayed, the slider's position reflects the current system audio output volume. As the user drags the slider, the changes update the volume. If the user presses the device volume buttons while sound is playing, the slider moves to reflect the new volume.

If there is an Apple TV or other AirPlay-enabled device in range, the route button allows the user to choose it. If there is only one audio output route available, the route button is not displayed.

Use this class by embedding an instance of it in your view hierarchy. The following code snippet assumes you have placed an instance of the `UIView` class on a view using Interface Builder, sizing and positioning it as desired to contain the volume view. Point to the `UIView` instance with an outlet variable—named, in the case of this example, `mpVolumeViewParentView`. You would typically place code like that shown in Listing 20-1 in your `viewDidLoad` method.

Listing 20-1 Adding a volume view to your view hierarchy

```
mpVolumeViewParentView.backgroundColor = [UIColor clearColor];
MPVolumeView *myVolumeView =
    [[MPVolumeView alloc] initWithFrame: mpVolumeViewParentView.bounds];
[mpVolumeViewParentView addSubview: myVolumeView];
[myVolumeView release];
```

When an audio output route that does not support volume control, such as A2DP, is active, the volume slider is replaced with the route name.

To instead display a volume slider as an alert, use the functions described in *Media Player Functions Reference*.

Tasks

Resizing Subviews

`- sizeThatFits:` (page 137)
Calculates and returns a size that best fits the receiver's subviews.

Managing Visibility of Controls

`showsVolumeSlider` (page 136) *property*
Determines whether or not the volume slider is visible in the volume view.

`showsRouteButton` (page 136) *property*
Determines whether or not the route button is visible in the volume view.

Properties

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

showsRouteButton

Determines whether or not the route button is visible in the volume view.

```
@property (nonatomic) BOOL showsRouteButton;
```

Discussion
The route button is visible by default when there is more than one audio output route available. To hide the route button, set this property's value to `NO`.

Availability
Available in iOS 4.2 and later.

Declared In
`MPVolumeView.h`

showsVolumeSlider

Determines whether or not the volume slider is visible in the volume view.

```
@property (nonatomic) BOOL showsVolumeSlider;
```

Discussion
The volume slider is visible by default. To hide the volume slider, set this property's value to `NO`.

Availability
Available in iOS 4.2 and later.

Declared In
`MPVolumeView.h`

Instance Methods

sizeThatFits:

Calculates and returns a size that best fits the receiver's subviews.

```
- (CGSize)sizeThatFits:(CGSize)size
```

Parameters

size

The preferred size of the receiver.

Return Value
A new size that fits the receiver's subviews.

Discussion
This method overrides the like-named method from the `UIView` class. It returns the preferred size the volume view needs to display the contained slider. You should not need to override this method.

Availability
Available in iOS 2.0 and later.

Declared In
`MPVolumeView.h`

UIViewController MediaPlayer Additions Reference

Inherits from	UIResponder : NSObject
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.2 and later.
Declared in	MPMoviePlayerViewController.h

Overview

This category adds methods to the `UIViewController` class for presenting and dismissing a movie player using a specific set of animations. The transitions used by these methods are the same ones used by the YouTube and iPod applications to display video content.

Tasks

Presenting and Dismissing the Movie Player

`- presentMoviePlayerViewControllerAnimated:` (page 140)
Presents the movie player view controller using the standard movie player transition.

`- dismissMoviePlayerViewControllerAnimated` (page 139)
Dismisses a movie player view controller using the standard movie player transition.

Instance Methods

dismissMoviePlayerViewControllerAnimated

Dismisses a movie player view controller using the standard movie player transition.

```
- (void)dismissMoviePlayerViewControllerAnimated
```

Discussion
If the receiver's `modalViewController` property does not contain a movie player view controller, this method does nothing.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerViewController.h`

presentMoviePlayerViewControllerAnimated:

Presents the movie player view controller using the standard movie player transition.

```
- (void)presentMoviePlayerViewControllerAnimated:(MPMoviePlayerViewController
    *)moviePlayerViewController
```

Parameters

moviePlayerViewController

The movie player view controller to present.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMoviePlayerViewController.h`

Protocols

MPMediaPickerControllerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 3.0 and later.
Declared in	
Companion guide	iPod Library Access Programming Guide
Related sample code	AddMusic

Overview

The delegate for a media item picker can respond to a user making media item selections. The delegate is also responsible for dismissing the media item picker from the parent view controller. The methods in this protocol are optional.

Media items are described in *MPMediaItem Class Reference*. Media item pickers are described in *MPMediaPickerController Class Reference*.

Tasks

Responding to User Actions

`- mediaPicker:didPickMediaItems:` (page 144)
Called when a user has selected a set of media items.

`- mediaPickerDidCancel:` (page 144)
Called when a user dismisses a media item picker by tapping Cancel.

Instance Methods

mediaPicker:didPickMediaItems:

Called when a user has selected a set of media items.

```
- (void)mediaPicker: (MPMediaPickerController *)mediaPicker
    didPickMediaItems:(MPMediaItemCollection *)mediaItemCollection
```

Parameters

mediaPicker

The media item picker to dismiss.

mediaItemCollection

The selected media items.

Availability
Available in iOS 3.0 and later.

Declared In
MPMediaPickerController.h

mediaPickerDidCancel:

Called when a user dismisses a media item picker by tapping Cancel.

```
- (void)mediaPickerDidCancel:(MPMediaPickerController *)mediaPicker
```

Parameters

mediaPicker

The media item picker to dismiss.

Availability
Available in iOS 3.0 and later.

Declared In
MPMediaPickerController.h

MPMediaPlayback Protocol Reference

Framework /System/Library/Frameworks/MediaPlayer.framework

Availability Available in iOS 3.2 and later.

Declared in

Overview

The `MPMediaPlayback` protocol defines the interface adopted by the `MPMoviePlayerController` class for controlling media playback. This protocol supports basic transport operations including start, stop, and pause, and also lets you seek forward and back through a movie or to a specific point in its timeline.

Tasks

Starting and Stopping Playback

- `play` (page 148) *required method*
  Initiates playback of the current item. (required)
- `pause` (page 148) *required method*
  Pauses playback of the current item. (required)
- `stop` (page 149) *required method*
  Ends playback of the current item. (required)
- `prepareToPlay` (page 149) *required method*
  Prepares the current item for playback. (required)

`isPreparedToPlay` (page 147) *required property*
A Boolean value indicating whether the media items are ready to play. (required) (read-only)

Seeking Within Media

- `beginSeekingBackward` (page 147) *required method*
  Begins seeking backward through the media content. (required)

- `beginSeekingForward` (page 147) *required method*
  Begins seeking forward through the media content. (required)
- `endSeeking` (page 147) *required method*
  Ends forward and backward seeking through the media content. (required)

Accessing Playback Attributes

`currentPlaybackRate` (page 146) *required property*
The current playback rate for the player. (required)

`currentPlaybackTime` (page 146) *required property*
The current position of the playhead. (required)

Properties

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

currentPlaybackRate

The current playback rate for the player. (required)

```
@property(nonatomic) float currentPlaybackRate
```

Discussion
This value represents a multiplier for the default playback rate of the current item. A value of 0.0 indicates that playback is stopped while a value of 1.0 indicates that playback is occurring at normal speed. Positive values indicate forward playback while negative values indicate reverse playback.

Setting the value of this property changes the playback rate accordingly.

currentPlaybackTime

The current position of the playhead. (required)

```
@property(nonatomic) NSTimeInterval currentPlaybackTime
```

Discussion
For video-on-demand or progressively downloaded content, this value is measured in seconds from the beginning of the current item. Changing the value of this property moves the playhead to the new location. For content streamed live from a server, this value represents the time from the beginning of the playlist when it was first loaded.

isPreparedToPlay

A Boolean value indicating whether the media items are ready to play. (required) (read-only)

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

See Also
- `prepareToPlay` (page 149)

Instance Methods

beginSeekingBackward

Begins seeking backward through the media content. (required)

```
- (void)beginSeekingBackward
```

Discussion
Use this method to move the current playback position backward in time at an accelerated rate. Seeking begins when you call this method and continues until you call the `endSeeking` (page 147) method.

If the underlying content is streamed, this method has no effect.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMediaPlayback.h`

beginSeekingForward

Begins seeking forward through the media content. (required)

```
- (void)beginSeekingForward
```

Discussion
Use this method to move the current playback position forward in time at an accelerated rate. Seeking begins when you call this method and continues until you call the `endSeeking` (page 147) method.

If the underlying content is streamed, this method has no effect.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMediaPlayback.h`

endSeeking

Ends forward and backward seeking through the media content. (required)

```
- (void)endSeeking
```

Discussion
You must call this method to end a seeking operation begun by calling either the `beginSeekingBackward` (page 147) or `beginSeekingForward` (page 147) method. After calling this method, the player returns to the same state it was in prior to seeking. In other words, if the item was playing before seeking began, it continues playing from the new playhead position after calling this method.

If the underlying content is streamed, this method has no effect.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMediaPlayback.h`

pause

Pauses playback of the current item. (required)

```
- (void)pause
```

Discussion
If playback is not currently underway, this method has no effect. To resume playback of the current item from the pause point, call the `play` (page 148) method.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMediaPlayback.h`

play

Initiates playback of the current item. (required)

```
- (void)play
```

Discussion
If playback was previously paused, this method resumes playback where it left off; otherwise, the method begins playing the first available item.

If the item has not yet been prepared for playback, this method must prepare it before beginning playback. Preparing media in advance (by calling the `prepareToPlay` (page 149) method) can help minimize any delays between the calling of this method and the beginning of playback.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMediaPlayback.h`

prepareToPlay

Prepares the current item for playback. (required)

```
- (void)prepareToPlay
```

Discussion
This method is called automatically when you call the `play` (page 148) method. Calling it before you call `play` gives the receiver a chance to prepare items sooner and may result in decreased latency when starting playback. However, calling this method may also interrupt any active audio sessions.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMediaPlayback.h`

stop

Ends playback of the current item. (required)

```
- (void)stop
```

Discussion
This method stops playback of the current item and resets the playhead to the start of the item. Calling the `play` (page 148) method again initiates playback from the beginning of the item.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMediaPlayback.h`

Notifications

MPMediaPlaybackIsPreparedToPlayDidChangeNotification

Notifies observers of a change in the prepared-to-play state of an object conforming to the `MPMediaPlayback` protocol. The `object` parameter of the notification contains the object whose state changed.

Availability
Available in iOS 3.2 and later.

Declared In
`MPMediaPlayback.h`

Functions

Media Player Functions Reference

Framework: MediaPlayer/MediaPlayer.h
Declared in MPVolumeSettings.h

Overview

The Media Player framework defines several functions for use in displaying and hiding volume controls.

Functions

MPVolumeSettingsAlertHide

Hides the alert panel that controls the system volume.

```
void MPVolumeSettingsAlertHide();
```

Availability
Available in iOS 2.0 and later.

See Also
`MPVolumeSettingsAlertShow` (page 154)

Declared In
`MPVolumeSettings.h`

MPVolumeSettingsAlertIsVisible

Returns a Boolean value indicating whether the volume alert panel is currently visible.

```
BOOL MPVolumeSettingsAlertIsVisible();
```

Return Value
`YES` if the volume alert is visible; otherwise, `NO`.

Availability
Available in iOS 2.0 and later.

See Also
`MPVolumeSettingsAlertShow` (page 154)

Declared In
`MPVolumeSettings.h`

MPVolumeSettingsAlertShow

Displays an alert panel for controlling the system volume.

```
void MPVolumeSettingsAlertShow();
```

Discussion
The alert panel displayed by this function floats above the contents of the current window. It contains a slider for adjusting the system volume setting and a Done button so that the user can dismiss the panel. You can also dismiss the panel programmatically using the `MPVolumeSettingsAlertHide` function.

Availability
Available in iOS 2.0 and later.

See Also
`MPVolumeSettingsAlertHide` (page 153)

Declared In
`MPVolumeSettings.h`

Document Revision History

This table describes the changes to *Media Player Framework Reference*.

Date	Notes
2010-12-09	Added links to *MPMovieAccessLog Class Reference*, *MPMovieAccessLogEvent Class Reference*, *MPMovieErrorLog Class Reference*, and *MPMovieErrorLogEvent Class Reference*.
2010-10-12	Added links to *MPMediaEntity Class Reference* and to *MPMediaQuerySection Class Reference*.
2010-09-01	Added links to *MPMoviePlayerViewController Class Reference*, *MPMediaPlayback Protocol Reference*, and *UIViewController MediaPlayer Additions Reference*.
2010-04-10	Added link to *MPTimedMetadata Class Reference*.
2010-03-02	Added new classes that were introduced in iOS 3.2.
2009-11-24	Minor changes.
2009-04-14	Updated for iPhone OS 3.0.
2008-04-18	New collection that describes the classes and functions that provide basic facilities for playing movie files.

Document Revision History