Message UI Framework Reference

User Experience

2010-04-30

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

Simultaneously published in the United States and Canada.

Contents

Figures

Introduction

Framework	/System/Library/Frameworks/MessageUI.framework
Header file directories	/System/Library/Frameworks/MessageUI.framework/Headers
Companion guide	Device Features Programming Guide
Declared in	MFMailComposeViewController.h MFMessageComposeViewController.h

The Message UI framework contains view controllers for presenting standard composition interfaces for things like email and SMS messages. You can use these interfaces from your own applications to incorporate the corresponding message delivery capabilities without requiring the user to leave your application.

The composition interface classes are standard view controllers. To display one of the interfaces, you present the corresponding view controller modally from your application. Once presented, the user has the option of customizing the email contents before sending or canceling the email. Your custom delegate object then handles the dismissal of the view controller at the appropriate time. For information about how to present and dismiss view controllers, see *View Controller Programming Guide for iOS*.

Important: If an iOS-based device is not configured to send a given type of message, you should avoid displaying the corresponding composition interface. The view controllers in this framework provide methods for determining if support is available for a given message type.

Introduction

Classes

PART I

Classes

MFMailComposeViewController Class Reference

Inherits from	UINavigationController : UIViewController : UIResponder : NSObject
Conforms to	NSCoding (UIViewController) NSObject (NSObject)
Availability	Available in iOS 3.0 and later.
Declared in	MessageUI/MFMailComposeViewController.h

Overview

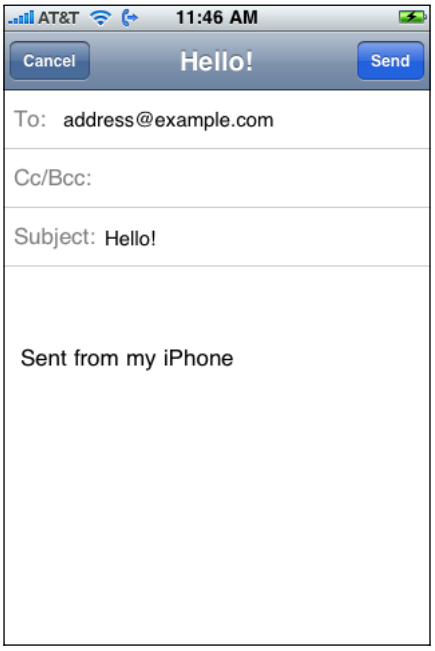
The `MFMailComposeViewController` class provides a standard interface that manages the editing and sending an email message. You can use this view controller to display a standard email view inside your application and populate the fields of that view with initial values, such as the subject, email recipients, body text, and attachments. The user can edit the initial contents you specify and choose to send the email or cancel the operation.

Using this interface does not guarantee immediate delivery of the corresponding email message. The user may cancel the creation of the message, and if the user does choose to send the message, the message is only queued in the Mail application outbox. This allows you to generate emails even in situations where the user does not have network access, such as in airplane mode. This interface does not provide a way for you to verify whether emails were actually sent.

Before using this class, you must always check to see if the current device is configured to send email at all using the `canSendMail` (page 13) method. If the user's device is not set up for the delivery of email, you can notify the user or simply disable the email dispatch features in your application. You should not attempt to use this interface if the `canSendMail` (page 13) method returns `NO`.

To display the view managed by this view controller, you can use any of the standard techniques for displaying view controllers. However, the most common way to present this interface is do so modally using the `presentModalViewController:animated:` method. Figure 1-1 shows the view that is displayed when you present the mail composition interface, with some of the fields already filled in. For more information on displaying the views associated with view controllers, see *View Controller Programming Guide for iOS*.

Figure 1-1 The mail composition interface



Important: The mail composition interface itself is not customizable and must not be modified by your application. In addition, after presenting the interface, your application is not allowed to make further changes to the email content. The user may still edit the content using the interface, but programmatic changes are ignored. Thus, you must set the values of content fields before presenting the interface.

Tasks

Determining Mail Availability

`+ canSendMail` (page 13)
Returns a Boolean indicating whether the current device is able to send email.

Setting Mail Fields Programmatically

`- setSubject:` (page 16)
Sets the initial text for the subject line of the email.

`- setToRecipients:` (page 16)
Sets the initial recipients to include in the email's "To" field.

`- setCcRecipients:` (page 15)
Sets the initial recipients to include in the email's "Cc" field.

`- setBccRecipients:` (page 14)
Sets the initial recipients to include in the email's "Bcc" field.

- `setMessageBody:isHTML:` (page 15)
  Sets the initial body text to include in the email.
- `addAttachmentData:mimeType:fileName:` (page 14)
  Adds the specified data as an attachment to the message.

Accessing the Delegate

`mailComposeDelegate` (page 13) *property*
The mail composition view controller's delegate.

Properties

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

mailComposeDelegate

The mail composition view controller's delegate.

```
@property(nonatomic,assign) id<MFMailComposeViewControllerDelegate>
    mailComposeDelegate;
```

Discussion
The delegate object is responsible for dismissing the view presented by this view controller at the appropriate time. Therefore, you should always provide a delegate and that object should implement the methods of the `MFMailComposeViewControllerDelegate` protocol.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

Class Methods

canSendMail

Returns a Boolean indicating whether the current device is able to send email.

```
+ (BOOL)canSendMail
```

Return Value
`YES` if the device is configured for sending email or `NO` if it is not.

Discussion
You should call this method before attempting to display the mail composition interface. If it returns `NO`, you must not display the mail composition interface.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

Instance Methods

addAttachmentData:mimeType:fileName:

Adds the specified data as an attachment to the message.

```
- (void)addAttachmentData:(NSData*)attachment  mimeType:(NSString*)mimeType
    fileName:(NSString*)filename
```

Parameters

attachment

The data to attach. Typically, this is the contents of a file that you want to include. This parameter must not be `nil`.

mimeType

The MIME type of the specified data. (For example, the MIME type for a JPEG image is `image/jpeg`.) For a list of valid MIME types, see http://www.iana.org/assignments/media-types/. This parameter must not be `nil`.

filename

The preferred filename to associate with the data. This is the default name applied to the file when it is transferred to its destination. Any path separator (/) characters in the filename are converted to underscore (_) characters prior to transmission. This parameter must not be `nil`.

Discussion
This method attaches the specified data after the message body but before the user's signature. You may attach multiple files (using different file names) but must do so prior to displaying the mail composition interface. Do not call this method after presenting the interface to the user.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

setBccRecipients:

Sets the initial recipients to include in the email's "Bcc" field.

```
- (void)setBccRecipients:(NSArray*)bccRecipients
```

Parameters

bccRecipients

An array of `NSString` objects, each of which contains the email address of a single recipient.

Discussion
This method replaces the previous blind carbon-copy recipients with the new ones listed in the *bccRecipients* parameter. This method does not filter out duplicate email addresses, so if duplicates are present, multiple copies of the email message may be sent to the same address.

You should call this method before you display the mail composition interface only. Do not call it after presenting the interface to the user.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

setCcRecipients:

Sets the initial recipients to include in the email's "Cc" field.

```
- (void)setCcRecipients:(NSArray*)ccRecipients
```

Parameters

ccRecipients

An array of `NSString` objects, each of which contains the email address of a single recipient.

Discussion
This method replaces the previous carbon-copy recipients with the new ones listed in the *ccRecipients* parameter. This method does not filter out duplicate email addresses, so if duplicates are present, multiple copies of the email message may be sent to the same address.

You should call this method before you display the mail composition interface only. Do not call it after presenting the interface to the user.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

setMessageBody:isHTML:

Sets the initial body text to include in the email.

```
- (void)setMessageBody:(NSString*)body  isHTML:(BOOL)isHTML
```

Parameters

body

The initial body text of the message. The text is interpreted as either plain text or HTML depending on the value of the *isHTML* parameter.

isHTML

Specify `YES` if the body parameter contains HTML content or specify `NO` if it contains plain text.

Discussion
This method replaces the previous body content with the new content. If the user has a signature file, the body content is inserted immediately before the signature. If you want to include images with your content, you must attach the images separately using the `addAttachmentData:mimeType:fileName:` method.

You should call this method before you display the mail composition interface only. Do not call it after presenting the interface to the user.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

setSubject:

Sets the initial text for the subject line of the email.

```
- (void)setSubject:(NSString*)subject
```

Parameters
subject
The text to display in the subject line.

Discussion
This method replaces the previous subject text with the new text. You should call this method before you display the mail composition interface only. Do not call it after presenting the interface to the user.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

setToRecipients:

Sets the initial recipients to include in the email's "To" field.

```
- (void)setToRecipients:(NSArray*)toRecipients
```

Parameters
toRecipients
An array of `NSString` objects, each of which contains the email address of a single recipient.

Discussion
This method replaces the previous recipients with the new ones listed in the *toRecipients* parameter. This method does not filter out duplicate email addresses, so if duplicates are present, multiple copies of the email message may be sent to the same address.

You should call this method before you display the mail composition interface only. Do not call it after presenting the interface to the user.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

Constants

MFMailComposeResult

Result codes returned when the mail composition interface is dismissed.

```
enum MFMailComposeResult {
    MFMailComposeResultCancelled,
    MFMailComposeResultSaved,
    MFMailComposeResultSent,
    MFMailComposeResultFailed
};
typedef enum MFMailComposeResult MFMailComposeResult;
```

Constants

`MFMailComposeResultCancelled`
The user cancelled the operation. No email message was queued.
Available in iOS 3.0 and later.
Declared in `MFMailComposeViewController.h`.

`MFMailComposeResultSaved`
The email message was saved in the user's Drafts folder.
Available in iOS 3.0 and later.
Declared in `MFMailComposeViewController.h`.

`MFMailComposeResultSent`
The email message was queued in the user's outbox. It is ready to send the next time the user connects to email.
Available in iOS 3.0 and later.
Declared in `MFMailComposeViewController.h`.

`MFMailComposeResultFailed`
The email message was not saved or queued, possibly due to an error.
Available in iOS 3.0 and later.
Declared in `MFMailComposeViewController.h`.

Mail Message Error Domain

The domain used for `NSError` objects associated with the mail composition interface.

```
NSString *const MFMailComposeErrorDomain;
```

Constants

`MFMailComposeErrorDomain`

The error domain associated with `NSError` objects.

Available in iOS 3.0 and later.

Declared in `MFMailComposeViewController.h`.

MFMailComposeErrorCode

Error codes for `NSError` objects associated with the mail composition interface.

```
enum MFMailComposeErrorCode {
    MFMailComposeErrorCodeSaveFailed,
    MFMailComposeErrorCodeSendFailed
};
typedef enum MFMailComposeErrorCode MFMailComposeErrorCode;
```

Constants

`MFMailComposeErrorCodeSaveFailed`

An error occurred trying to save the email message to the Drafts folder.

Available in iOS 3.0 and later.

Declared in `MFMailComposeViewController.h`.

`MFMailComposeErrorCodeSendFailed`

An error occurred while trying to queue or send the email message.

Available in iOS 3.0 and later.

Declared in `MFMailComposeViewController.h`.

MFMessageComposeViewController Class Reference

Inherits from	UINavigationController : UIViewController : UIResponder : NSObject
Conforms to	NSCoding (UIViewController) NSObject (NSObject)
Framework	/System/Library/Frameworks/MessageUI.framework
Availability	Available in iOS 4.0 and later.
Declared in	MFMessageComposeViewController.h

Overview

The `MFMessageComposeViewController` class presents a standard system interface for composing SMS text messages. You use this class to configure the initial recipients and body of the message and to configure a delegate to respond to the final result. After configuring the initial values, you present the view controller modally using the `presentModalViewController:animated:` method and dismiss it using the `dismissModalViewControllerAnimated:` method.

Before using this class, you must always check to see if the current device is configured to send SMS messages by calling the `canSendText` (page 21) class method. If the user's device is not set up for the delivery of SMS messages, you can notify the user or simply disable the SMS features in your application. You should not attempt to use this interface if the `canSendText` method returns `NO`.

Your delegate is responsible for dismissing the message compose view controller in its `messageComposeViewController:didFinishWithResult:` (page 27) method. For more information about implementing this method in your delegate object, see *MFMessageComposeViewControllerDelegate Protocol Reference*.

Important: The message composition interface itself is not customizable and must not be modified by your application. In addition, after presenting the interface, your application is not allowed to make further changes to the SMS content. The user may still edit the content using the interface, but programmatic changes are ignored. Thus, you must set the values of content fields before presenting the interface

Tasks

Determining If Message Composition Is Available

`+ canSendText` (page 21)
Returns a Boolean value indicating whether the current device is capable of sending text messages.

Accessing the Delegate

`messageComposeDelegate` (page 21) *property*
The delegate to which message-related notifications should be sent.

Setting the Initial Message Information

`recipients` (page 21) *property*
An array of strings containing the initial recipients of the message.

`body` (page 20) *property*
The initial content of the message.

Properties

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

body

The initial content of the message.

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

Availability
Available in iOS 4.0 and later.

Declared In
`MFMessageComposeViewController.h`

messageComposeDelegate

The delegate to which message-related notifications should be sent.

```
@property(nonatomic,assign) id<MFMessageComposeViewControllerDelegate>
    messageComposeDelegate
```

Discussion
When the user taps a button to send or cancel the message, your delegate is notified and should respond by dismissing the message composition interface. For more information about implementing the methods of your delegate object, see *MFMessageComposeViewControllerDelegate Protocol Reference*.

Availability
Available in iOS 4.0 and later.

Declared In
`MFMessageComposeViewController.h`

recipients

An array of strings containing the initial recipients of the message.

```
@property(nonatomic,copy) NSArray *recipients
```

Discussion
Each string should in the array should contain the phone number of the intended recipient.

Availability
Available in iOS 4.0 and later.

Declared In
`MFMessageComposeViewController.h`

Class Methods

canSendText

Returns a Boolean value indicating whether the current device is capable of sending text messages.

```
+ (BOOL)canSendText
```

Return Value
`YES` if the device can send text messages or `NO` if it cannot.

Discussion
You should always call this method before attempting to present the message compose view controller. A device may be unable to send messages if it does not support text message or if it is not currently configured to send messages. This method applies only to the ability to send text messages. Sending multimedia messages with this class is not supported.

Availability
Available in iOS 4.0 and later.

Declared In
`MFMessageComposeViewController.h`

Constants

MessageComposeResult

These constants describe the result of the message composition interface.

```
enum MessageComposeResult {
    MessageComposeResultCancelled,
    MessageComposeResultSent,
    MessageComposeResultFailed
};
typedef enum MessageComposeResult MessageComposeResult;
```

Constants

`MessageComposeResultCancelled`
The user canceled the composition.
Available in iOS 4.0 and later.
Declared in `MFMessageComposeViewController.h`.

`MessageComposeResultSent`
The user successfully queued or sent the message.
Available in iOS 4.0 and later.
Declared in `MFMessageComposeViewController.h`.

`MessageComposeResultFailed`
The user's attempt to save or send the message was unsuccessful.
Available in iOS 4.0 and later.
Declared in `MFMessageComposeViewController.h`.

Availability
Available in iOS 4.0 and later.

Declared In
`MFMessageComposeViewController.h`

Protocols

MFMailComposeViewControllerDelegate Protocol Reference

Conforms to	NSObject
Availability	Available in iOS 3.0 and later.
Declared in	MessageUI/MFMailComposeViewController.h

Overview

The `MFMailComposeViewControllerDelegate` protocol defines the method that your delegate must implement to manage the mail composition interface. The method of this protocol notifies your delegate object when the user has finished with the interface and is ready to dismiss it.

Your delegate object is responsible for dismissing the picker when the operation completes. You do this using the `dismissModalViewControllerAnimated:` method of the parent view controller responsible for displaying the `MFMailComposeViewController` object's interface.

Tasks

Responding to Email Completion

`- mailComposeController:didFinishWithResult:error:` (page 25)
Tells the delegate that the user wants to dismiss the mail composition view.

Instance Methods

mailComposeController:didFinishWithResult:error:

Tells the delegate that the user wants to dismiss the mail composition view.

```
- (void)mailComposeController:(MFMailComposeViewController*)controller
    didFinishWithResult:(MFMailComposeResult)result error:(NSError*)error
```

Parameters

controller
The view controller object managing the mail composition view.

result
The result of the user's action.

error
If an error occurred, this parameter contains an error object with information about the type of failure.

Discussion

Your implementation of this method should dismiss the mail composition view. Implementation of this method is optional but expected.

If the user has opted to send the email created by this interface, that email should be queued in the user's Mail program by the time this method is called. If an error occurred while queueing the email message, the *error* parameter contains an error object indicating the type of failure that occurred.

Availability
Available in iOS 3.0 and later.

Declared In
`MFMailComposeViewController.h`

MFMessageComposeViewControllerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/MessageUI.framework
Availability	Available in iOS 4.0 and later.
Declared in	MFMessageComposeViewController.h

Overview

The `MFMessageComposeViewControllerDelegate` protocol defines a single method that custom objects can implement to respond to updates from a `MFMessageComposeViewController` class. You use the method of this protocol to respond to the end of the user composing an SMS message. The method includes information about whether the user chose to send or cancel the message or whether the attempt to send it failed.

Tasks

Responding to the Message Completion

`- messageComposeViewController:didFinishWithResult:` (page 27) *required method*
Tells the delegate that the user finished composing the message. (required)

Instance Methods

messageComposeViewController:didFinishWithResult:

Tells the delegate that the user finished composing the message. (required)

```
- (void)messageComposeViewController:(MFMessageComposeViewController *)controller
    didFinishWithResult:(MessageComposeResult)result
```

Parameters

controller
The message composition view controller that is returning the result.

result

A result code indicating how the user chose to complete the composition.

Discussion

This method is called when the user taps one of the buttons to dismiss the message composition interface. Your implementation of this method should dismiss the view controller and perform any additional actions needed to process the sending of the message. The result parameter lets you know whether the user chose to cancel or send the message or whether sending the message failed.

Implementation of this method is required.

Availability

Available in iOS 4.0 and later.

Declared In

`MFMessageComposeViewController.h`

Document Revision History

This table describes the changes to *Message UI Framework Reference*.

Date	Notes
2010-04-30	Added the MFMessageComposeViewController class and corresponding delegate protocol.
2009-07-15	Added an introduction to the framework collection.
2009-02-22	New document describing the classes of the Message UI framework.

Document Revision History