What’s New in iMessage Apps

Session 234

Eugene Bistolas, Messages Engineer
Jay Chae, Messages Engineer
Stephen Lottermoser, Messages Engineer
What’s New in Messages
Direct Send
Live Message Layouts
Best Practices
What’s New in Messages

Direct Send
Live Message Layouts
Best Practices
What’s New in Messages

Direct Send

Live Message Layouts

Best Practices
What’s New in Messages
Direct Send
Live Message Layouts
Best Practices
What’s New in Messages
Direct Send
Live Message Layouts
Best Practices
iMessage App Fundamentals
iMessage App Fundamentals

iMessage Apps and Stickers Part 2

WWDC 2016
What’s New in Messages
Add your face to GIFs, eCards and Music Videos.

Sign in with Facebook

Sign up with Email

Already have an account? Sign In
Direct Send
// iMessage App Insert Draft API

open func insert(_ message: MSMessage, completionHandler: ((Error?) -> Void)? = nil)

open func insert(_ sticker: MSSticker, completionHandler: ((Error?) -> Void)? = nil)

open func insertText(_ text: String, completionHandler: ((Error?) -> Void)? = nil)

open func insertAttachment(_ URL: URL, withAlternateFilename filename: String?, completionHandler: ((Error?) -> Void)? = nil)
// iMessage App Direct Send API

open func send(_ message: MSMessage, completionHandler: ((Error?) -> Void)? = nil)

open func send(_ sticker: MSSticker, completionHandler: ((Error?) -> Void)? = nil)

open func sendText(_ text: String, completionHandler: ((Error?) -> Void)? = nil)

open func sendAttachment(_ URL: URL, withAlternateFilename filename: String?, completionHandler: ((Error?) -> Void)? = nil)
Choosing a Send API

Insert draft API

Insert is recommended for most iMessage apps

Best for rich message composition

• Append comments
• Send with effects, full screen moments

Allow user confirmation for sent content
Choosing a Send API

Direct send API

Provides quick fire and forget user experience

Great for flows where inserting message, then sending, adds extra step

Maintain trust in your app

• Clearly indicate to user what content will be sent

• Clearly denote UI elements that trigger a send
iMessage App Direct Send
API requirements

Messages enforces requirements for API use

• App must be visible
• App can send one message per user interaction
• No further messages can be sent until next interaction
• New error codes
iMessage App Direct Send

API requirements

Messages enforces requirements for API use

• App must be visible
• App can send one message per user interaction
• No further messages can be sent until next interaction
• New error codes

// MSMessage Error Codes

```swift
public enum MSMessageErrorCode: Int {
    // New iOS 11 Error Codes
    @available(iOS 11.0, *)
    case sendWithoutRecentInteraction

    @available(iOS 11.0, *)
    case sendWhileNotVisible
}
```
Direct Send

New fast and easy send experience

For some apps, direct send provides excellent alternative to staging drafts

Consider which API is best for your app
Live Message Layouts

Stephen Lottermoser, Messages Engineer
Template Image
Template message
A simple template message
Read 4:47 PM

Your App
Your App!

Your App
Your App!
Your App!
Your App!
Your App!
Jay

Today 1:25 AM

iMessage

Your App!

Your App!

Someone Else's App

Read 1:26 AM
Your App!

Your App!

Your App!

Read 1:26 AM
Demo
Live Message Layouts
How do they work?

iMessage app

MSMessagesAppViewController .compact

Your App
Live Message Layouts
How do they work?

iMessage app

MSMessagesAppViewController
.compact

MSMessagesAppViewController
.transcript
Live Message Layouts
How do they work?

iMessage app
Live Message Layouts
How do they work?

iMessage app

MSMessagesAppViewController
.transcript

Your App!
Live Message Layouts
How do they work?

iMessage app

MSMessagesAppViewController
.transcript

MSMessagesAppViewController
.transcript
Live Message Layouts
How do they work?

iMessage app

MSMessagesAppViewController
  .transcript

MSMessagesAppViewController
  .transcript

MSMessagesAppViewController
  .transcript

Your App!

Your App!
Live Message Layouts
Sending and receiving messages
// MSMessageLiveLayout

@available(iOS 11.0, *)
open class MSMessageLiveLayout : MSMessageLayout {

    public init(alternateLayout: MSMessageTemplateLayout)

    open var alternateLayout: MSMessageTemplateLayout { get }
}
MSMessageLiveLayout
Alternate layout

Used on:
• Devices without your iMessage app installed
• Devices running iOS 10
• Devices running watchOS 3 or macOS Sierra and later
iMessage app installed

MSMessageLiveLayout
Alternate layout

iMessage app not installed

You are going to The Bash!

<table>
<thead>
<tr>
<th>02</th>
<th>14</th>
<th>05</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Days</td>
<td>Hours</td>
<td>Minutes</td>
<td>Seconds</td>
</tr>
</tbody>
</table>

Invitation to The Bash!
June 8, 2017 at 5:00 PM
let alternateLayout = MSMessageTemplateLayout()
alternateLayout.caption = summaryText
alternateLayout.subcaption = readableDateFormatter.string(from: event.date)
alternateLayout.image = UIImage(named: event.fallbackImageName)

let layout = MSMessageLiveLayout(alternateLayout: alternateLayout)
let message = MSMessage()
message.layout = layout

conversation.send(message, completionHandler: nil)
let alternateLayout = MSMassageTemplateLayout()
alternateLayout.caption = summaryText
alternateLayout.subcaption = readableDateFormatter.string(from: event.date)
alternateLayout.image = UIImage(named: event.fallbackImageName)

let layout = MSMassageLiveLayout(alternateLayout: alternateLayout)
let message = MSMassage()
message.layout = layout

conversation.send(message, completionHandler: nil)
let alternateLayout = MSMessageTemplateLayout()
alternateLayout.caption = summaryText
alternateLayout.subcaption = readableDateFormatter.string(from: event.date)
alternateLayout.image = UIImage(named: event.fallbackImageName)

let layout = MSMessageLiveLayout(alternateLayout: alternateLayout)
let message = MSMessage()
message.layout = layout

conversation.send(message, completionHandler: nil)
let alternateLayout = MSMMessageTemplateLayout()
alternateLayout.caption = summaryText
alternateLayout.subcaption = readableDateFormatter.string(from: event.date)
alternateLayout.image = UIImage(named: event.fallbackImageName)

let layout = MSMMessageLiveLayout(alternateLayout: alternateLayout)
let message = MSMMessage()
message.layout = layout

conversation.send(message, completionHandler: nil)
MSMessagesAppViewController
MSMessagesAppViewController
MSMessagesAppViewController
public enum MSMessagesAppPresentationStyle : UInt {
    case compact
    case expanded
    @available(iOS 11.0, *)
    case transcript
}
// Configuring a MSMessagesAppViewController subclass for display

public enum MSMessagesAppPresentationStyle: UInt {
    case compact
    case expanded
    @available(iOS 11.0, *)
    case transcript
}

MSMessagesAppViewController

Configuring for display

When `willBecomeActive(with:)` is called, you have enough information to configure your view controller.

- `presentationStyle`
- `activeConversation?.selectedMessage`
When `willBecomeActive(with:)` is called, you have enough information to configure your view controller.

- `presentationStyle`
- `activeConversation?.selectedMessage`
MSMessagesAppViewController
Configuring for display

When `willBecomeActive(with:)` is called, you have enough information to configure your view controller.

- `presentationStyle`
- `activeConversation?.selectedMessage`

**Transcript Child View Controller**
`transcript`

**Compact Child View Controller**
`compact`

(selectedMessage)
override func willBecomeActive(with conversation: MSConversation) {
    super.willBecomeActive(with: conversation)
    let message = activeConversation?.selectedMessage

    switch presentationStyle {
    case .compact:
        presentSummaryViewController(for: message)
    case .expanded:
        presentDetailViewController(for: message)
    }
}
override func willBecomeActive(with conversation: MSConversation) {
    super.willBecomeActive(with: conversation)
    let message = activeConversation?.selectedMessage

    switch presentationStyle {
    case .compact:
        presentSummaryViewController(for: message)
    case .expanded:
        presentDetailViewController(for: message)
    case .transcript:
        presentTranscriptViewController(for: message)
    }
}
override func willBecomeActive(with conversation: MSConversation) {
    super.willBecomeActive(with: conversation)
    let message = activeConversation?.selectedMessage

    switch presentationStyle {
    case .compact:
        presentSummaryViewController(for: message)
    case .expanded:
        presentDetailViewController(for: message)
    case .transcript:
        presentTranscriptViewController(for: message)
    }
}
// Configuring a MSMessagesAppViewController subclass for display

public enum MSMessagesAppPresentationStyle : UInt {  
    case compact  
    case expanded  
    @available(iOS 11.0, *)  
    case transcript  
}
// Configuring a MSMessagesAppViewController subclass for display

public enum MSMessagesAppPresentationStyle : UInt {
    case compact
    case expanded
    @available(iOS 11.0, *)
    case transcript
}

@available(iOS 11.0, *)
public protocol MSMessagesAppTranscriptPresentation {
    public func contentSizeThatFits(_ size: CGSize) -> CGSize
}

override func contentSizeThatFits(_ size: CGSize) -> CGSize {
    let contentHeight: CGFloat = 217.0
    let titleFont = EventCountdownTranscriptView.titleFont
    let titleHeight = titleFont.lineHeight
    let totalHeight = contentHeight + titleHeight
    return CGSize(width: size.width, height: totalHeight)
}
override func contentSizeThatFits(_ size: CGSize) -> CGSize {
    let contentHeight: CGFloat = 217.0
    let titleFont = EventCountdownTranscriptView.titleFont
    let titleHeight = titleFont.lineHeight

    let totalHeight = contentHeight + titleHeight
    return CGSize(width: size.width, height: totalHeight)
}
override func contentSizeThatFits(_ size: CGSize) -> CGSize {
    let contentHeight: CGFloat = 217.0
    let titleFont = EventCountdownTranscriptView.titleFont
    let titleHeight = titleFont.lineHeight
    let totalHeight = contentHeight + titleHeight
    return CGSize(width: size.width, height: totalHeight)
}
override func contentSizeThatFits(_ size: CGSize) -> CGSize {
    let contentHeight: CGFloat = 217.0
    let titleFont = EventCountdownTranscriptView.titleFont
    let titleHeight = titleFont.lineHeight
    let totalHeight = contentHeight + titleHeight
    return CGSize(width: size.width, height: totalHeight)
}
Code Demo
Live Message Layouts

Interaction and more

Stephen Lottermoser, Messages Engineer
Demo
Adding interaction
Interaction

Keep views lightweight
• Simple buttons or tap gestures
• Limit use of scroll views
• No keyboard input
Interaction

Use other presentation styles for complex interactions

requestPresentationStyle(.expanded)
Lifecycle
In iOS 10, these methods referred to the extension’s lifecycle

class MSMessagesAppViewController {
    open func willBecomeActive(with conversation: MSConversation)
    open func didResignActive(with conversation: MSConversation)
}
In iOS 10, these methods referred to the extension’s lifecycle

```swift
class MSMessagesAppViewController {
    open func willBecomeActive(with conversation: MSConversation)
    open func didResignActive(with conversation: MSConversation)
}
```
In iOS 11, these methods referred to the view controller’s lifecycle

```swift
class MSMessagesAppViewController {

    open func willBecomeActive(with conversation: MSConversation)
    open func didResignActive(with conversation: MSConversation)

}
```
Lifecycle
Extension callbacks

Observe extension notifications for extension lifecycle events

public static let NSExtensionHostWillEnterForeground: NSNotification.Name
public static let NSExtensionHostDidEnterBackground: NSNotification.Name
public static let NSExtensionHostWillResignActive: NSNotification.Name
public static let NSExtensionHostDidBecomeActive: NSNotification.Name
View Controller Lifecycle
Live message layout

When presenting a transcript presentation style view controller, methods will be called in this order

```swift
class MSMessagesAppViewController {
}
```
When presenting a transcript presentation style view controller, methods will be called in this order

```swift
class MSMessagesAppViewController {
    public init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {
    }
    open func viewDidLoad() {
    }
    open func viewDidLoad() {
    }
}
```
View Controller Lifecycle

Live message layout

When presenting a transcript presentation style view controller, methods will be called in this order

```swift
class MSMessagesAppViewController {

    public init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {
        // Initialization code
    }

    open func viewDidLoad() {
        // View loaded code
    }

    open func willBecomeActive(with conversation: MSConversation) {
        // Will become active code
    }

    open func didBecomeActive(with conversation: MSConversation) {
        // Did become active code
    }

}
```
When presenting a transcript presentation style view controller, methods will be called in this order

```swift
class MSMessagesAppViewController {
    public init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?)
    open func viewDidLoad()
    open func willBecomeActive(with conversation: MSConversation)
    open func didBecomeActive(with conversation: MSConversation)
    open func viewWillAppear(_ animated: Bool)
    open func viewDidAppear(_ animated: Bool)
    open func viewDidLoad() {
    }
}
```
View Controller Lifecycle
Live message layout

When presenting a transcript presentation style view controller, methods will be called in this order

class MSMessagesAppViewController {

    public init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?)
    open func viewDidLoad()

    open func willBecomeActive(with conversation: MSConversation)
    open func didBecomeActive(with conversation: MSConversation)

    open func viewWillAppear(_ animated: Bool)
    open func viewDidAppear(_ animated: Bool)

    open func contentSizeThatFits(_ size: CGSize) -> CGSize
}


When `willBecomeActive` is called, you have enough information to configure your view controller.

```swift
class MSMessagesAppViewController {
    public init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {
        // Initialization code...
    }

    open func viewDidLoad() {
        // View did load code...
    }

    open func willBecomeActive(with conversation: MSConversation) {
        // Will become active code...
    }

    open func didBecomeActive(with conversation: MSConversation) {
        // Did become active code...
    }

    open func viewWillAppear(_ animated: Bool) {
        // View will appear code...
    }

    open func viewDidAppear(_ animated: Bool) {
        // View did appear code...
    }

    open func contentSizeThatFits(_ size: CGSize) -> CGSize {
        // Content size that fits code...
    }
}
```
View Controller Lifecycle
Live message layout

`contentSizeThatFits()` is called last

It may be called multiple times in response to events in Messages

```swift
open func contentSizeThatFits(_ size: CGSize) -> CGSize
```
Pending Messages
Jay

Your App!

Delivered

Today 1:07 PM
open class MSMessage: NSObject, NSCopying, NSSecureCoding {
    @available(iOS 11.0, *)
    open var isPending: Bool { get }
}
Live Message Layouts

Interactive iMessage apps in the transcript!

Same `MSMessagesAppViewController` base class

New layout: `MSMessageLiveLayout`

New presentation style: `.transcript`
Best Practices

Eugene Bistolas, Messages Engineer
iMessage App Area Insets

Updated insets for Messages app area

Insets can be determined via

- `[UIView safeAreaInsets]`
- `UIViewController top/bottomLayoutGuide`
- `Autolayout`

Legacy apps get old insets
iMessage App Area Insets

Updated insets for Messages app area

Insets can be determined via

- `[UIView safeAreaInsets]`
- `UIViewController top/bottomLayoutGuide`
- Autolayout

Legacy apps get old insets
Message Summary Text

Succinct message summary
Provide summary for each message
Visible in notifications and conversation list

// MSMessage
open var summaryText: String?
Stickers

Preferred format: PNG / APNG

Optimizations
- Sticker pixel size
- Frame rate
- File size

Adhering to guidelines ensures fast launch, quick sends

For more info, reference Messages Human Interface Guidelines
MFMessageComposeViewController

MSMessage API

Match user experience between iMessage app and parent app

Template and Live layout support

Received messages can open in iMessage app or parent app

// MFMessageComposeViewController

/// This property sets the initial interactive message.
@available(iOS 10.0, *)
@NSCopying open var message: MSMessage?
Best Practices

Use layout margins

Set summaryText on all MSMessages

Consider sticker size, frame rate, for faster stickers

Send app balloons from MFMessageComposeViewController
Summary

What’s New in Messages

Direct Send

Live Message Layouts

Best Practices
More Information

<table>
<thead>
<tr>
<th>Session</th>
<th>Venue</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing Business Chat</td>
<td>Hall 3</td>
<td>Friday 10:00AM</td>
</tr>
<tr>
<td>Design Studio Shorts 3</td>
<td>Executive Ballroom</td>
<td>Friday 11:00AM</td>
</tr>
<tr>
<td>Labs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>iMessage Apps and Business Chat Lab</strong></td>
<td>Technology Lab B</td>
<td>Fri 11:00AM–2:00PM</td>
</tr>
</tbody>
</table>