RELEASE NOTES IN-APP MESSAGING SDK VERSION 2.8 - iOS

These are the main feature releases available in the **In-App Messaging SDK version 2.8 for iOS**.

Version 2.8 planned roll-out: September 27th 2017

New functionalities

- Structured content enablement (GA in SDK)
- Automatic messages for messaging
- Unread messages badge
- Set icon for “Send” button
- iOS11, XCode 9 and Swift 4/Swift 3.2 Certification, and removal of iOS8 from supported list
- Customizable bubble corners
- Reconnect attempt termination callback
- Logout - add completion closure/failure
- New event - conversation interactions

New properties

- Structured content
- Bubble corners
- Unread messages badge
- Scroll to bottom button
- Send button
- Controller bubble

New APIs

- Conversation interactions API
- Unread messages badge API
- Logout API

New Callbacks

- Reconnect attempt termination callback

New functionalities

Structured content enablement (GA in SDK)

Type: Feature

Available to all customers? No - early adopters only

The beta version was released in v2.7 (for a full description, refer to the v2.7 release notes). The SDK delivers structured content enablement only; the feature will be made fully productive in October. In v2.8 the feature is enabled by default in the SDK.

The dictionary of template elements can be found here.

What does enablement mean?
Until rollout is complete, the structured content capability in SDK v2.7 was flagged as a Beta feature. The feature has an enablement toggle in the SDK which was disabled by default.
In SDK v2.8, it is enabled by default.

The toggle may be switched on or off as part of the SDK implementation within the host app, however it is highly recommended not to release the SDK in the host app with structured content enabled until end to end flow has been fully tested on the brand's account.

Inapp Messaging SDK toggle -

- iOS - enableStructuredContent

Related properties: Structured content

The following additional conditions and configurations are required:*

Backend update	Backend enablement	Backend configuration	SDK enablement	SDK configuration
Yes	Yes	Yes	Yes	Yes

Automatic messages for messaging

Type: Feature

Available to all customers? No - early adopters only

Description
Automatic Messages (AKA System Messages) are predefined messages about events that occur in the conversation and are sent to the consumer as the events occur. Their purpose is to gain the consumer's trust in the messaging channel, by setting expectations and giving the consumer visibility over the agent's availability.

Auto messages are triggered upon specific events that are detected by the system (e.g. the consumer opens a new conversation, the conversation is transferred to another agent, the time to respond is updated, etc.). When the auto messages are sent, they are displayed to the consumer and the agent within the conversation transcript, and they also appear in the conversation's history both on the consumer's side and in LE.

Notes:

- Messages are supported in all LiveEngage languages.
- The content of each message can be edited by the brand.
- Skill variation is supported, including enabling/disabling the messages for each skill.
- Certain messages can have different parameters, such as the time the conversation is in the queue before the message is sent.
- Dynamic text can be added to the messages, which will be replaced with a runtime value, such as agent name.
- Auto messages do not affect whom the conversation is pending, nor the time to respond.
- They are filtered out of the reports by default (unless manually included).

The following auto messages are supported:

- New conversations
  - A consumer opens a conversation during working hours
  - A consumer opens a conversation for the first time ever, during working hours
- Off hours
  - A consumer opens a conversation during off hours
  - A consumer opens a conversation for the first time ever, during off hours
  - The consumer sends the first message during off hours in an open conversation
- Time to respond
  - The response time is updated manually by the agent
  - The consumer marks the conversation as urgent

  - The consumer dismisses the conversation urgent state
- Consumer/Agent non-responsive
  - The consumer has not responded for X seconds/minutes/hours
  - The agent did not respond for X seconds/minutes/hours
  - Conversation is in queue for X mins/hours
- Transfers and connection to agents
  - The conversation is transferred to a different skill
  - The agent returns the conversation to the queue
  - The consumer is connected to an agent
- Conversation participants
  - Agent manager joins the conversation
  - The joined agent manager leaves the conversation

How to enable auto messages
Auto messages will be enabled for early adopters upon release. Please contact your account manager for more information.

IMPORTANT NOTES:
When auto messages are enabled, they are all enabled by default and all have the default text. It is advised to review them immediately and modify them to suit the brand's needs.
Once auto messages are enabled, the SDK does not show toast messages which were presented in the past.
The following messages remain in the SDK :

- Introduction message from the consumer's first ever conversation. Make sure you do not have a collision between that message and auto messages.
- Conversation resolved message

The following additional conditions and configurations are required*:

Backend update	Backend enablement	Backend configuration	SDK enablement	SDK configuration
Yes	Yes	Yes	Yes	Yes

Unread messages badge

When there are unread messages waiting for the consumer within the brand app, this information can be pushed to display in the app's notification badge. Within the app, brands can develop their own visualization of a badge, such as a number, icon or other marker to show unread messages.

The unread messages number is passed to the SDK through LP Push service with every push.

IMPORTANT NOTES :
A push is sent to the last device which was registered to the LP push service, meaning that the unread messages indication can be fetched by only one device.

- If the user is using two devices in parallel, the device that does not receive push events will receive updates of the unread message indicator only once that a message has been sent from that device and the push arrives to it.
- In addition, if a conversation is ongoing in web messaging, then the push will not arrive to the device, since the web-socket is already open.

How to enable the unread messages counter
There are two options to set up this counter:

1. If the time condition is met, a REST request is performed to get the counter from the pusher
2. Return the cached number on the app

Parameters:

- conversationQuery: conversationQuery: used to identify the related brand
- completion: called once the operation ends successfully
- failure: called once the operation failed

Related properties: Unread messages badge
Related API: Unread messages badge API

The following additional conditions and configurations are required*:

Backend update	**Backend enablement**	**Backend configuration**	**SDK enablement**	**SDK configuration**
Yes	No	No	No	Yes

Set icon for “Send” button

Brands are now able to choose and configure their own icon for the ‘Send’ button for in-app messaging conversations.

How to configure the send button icon:

1. Ensure the SDK is set to use an icon and not text for the Send button (**isSendMessageButtonInTextMode = false**)
2. Set the icon - sendButtonImage: UIImage

IMPORTANT NOTES:
The chosen icon must adhere to Apple’s guidelines for custom icons.

Related properties: Send button

The following additional conditions and configurations are required*:

Backend update	**Backend enablement**	**Backend configuration**	**SDK enablement**	**SDK configuration**
N/A	N/A	N/A	N/A	Yes

iOS11, XCode 9 and Swift 4/Swift 3.2 Certification, and removal of iOS8 from supported list

The In-app Messaging SDK v.28 was built and certified with XCode 9 in Swift 4/ Swift 3.2 for iOS11.

The following devices were certified with iOS11:

- iPhone 6
- iPhone 7

With this new certification, support for iOS8 was removed.

The following additional conditions and configurations are required*:

Backend update	Backend enablement	Backend configuration	SDK enablement	SDK configuration
N/A	N/A	N/A	N/A	Yes

Customizable bubble corners

The radius of the corners of message bubbles can now be configured by the brand. There are 8 different parameters to be configured: the 4 corners of the agent message bubble and the 4 corners of the consumer message bubble.

The corners of the scroll button may also be set.

Related properties: Bubble corners

The following additional conditions and configurations are required*:

Backend update	Backend enablement	Backend configuration	SDK enablement	SDK configuration
N/A	N/A	N/A	N/A	Yes

Reconnect attempt termination callback

The new callback will be invoked when all connection retries have failed:

Related callback: Reconnect attempt termination callback

The following additional conditions and configurations are required*:

Backend update	Backend enablement	Backend configuration	SDK enablement	SDK configuration
N/A	N/A	N/A	N/A	Yes

Logout - add completion closure/failure

This method is a destructive method that is typically used to clean a user's data before a second user logs into the same device or simply to log the current user out.

The method carried out the following steps:
- Unregisters from the push notification service.
- Clears all SDK persistent data.
- Cleans running operations (see [destruct](consumer-experience-ios-sdk-destruct.html){:target="_blank"}).
- Invokes destruct() method

Parameters:
- completion: a completion block for successful logout. The completion block will be invoked only if all logout steps succeeded.
- failure: a failure block with a specified error for logout failure. The failure block will be invoked if at least one of the logout steps has failed.

Related API: Logout API

The following additional conditions and configurations are required*:

Backend update	Backend enablement	Backend configuration	SDK enablement	SDK configuration
N/A	N/A	N/A	N/A	Yes

New event - conversation interactions

The conversation interactions event communicates the Inactive time interval within seconds of the user last touching the screen. This interval applies to the scroll/messaging/action menus and any other general action on the conversation screen.

If the screen is not active or the application is running in the background, the API will return -1.

Reset events:
- send message
- text edit change
- scroll
- viewDidAppear
- appWillEnterForeground

- tap
- long press
- WindowContainerViewController -
- menuButtonClicked
- WindowContainerViewController -
- customButtonClicked

Clear events:

- viewDidDisappear
- appWillEnterBackground

Related API: Conversation interactions API

The following additional conditions and configurations are required*:

Backend update	Backend enablement	Backend configuration	SDK enablement	SDK configuration
N/A	N/A	N/A	N/A	Yes

** Key for items as follows:*
Backend update: This feature requires an update to the backend.
Backend enablement: This feature requires items to be toggled on in the backend.
Backend configuration: This feature requires configuration in the backend.
SDK enablement: This feature requires items to be toggled on in the SDK.
SDK configuration: This features requires items to be configured in the SDK.

New properties

Structured content

The following properties for structured content can now be configured:

Name	Description	Default
`enableStrucutredContent: Bool`	Enable or Disable toggle for Structured Content feature in conversations.	true

Bubble corners

The following properties for customizable bubble corners can now be configured:

Name	Description	Default
`userBubbleTopLeftCornerRadius:Float = 8`	Top left radius corner on the user bubble **Note (applies to all bubble corner properties:** Setting the radius to a value greater than 0.0 causes the bubble's layer to begin drawing rounded corners on its background. This attribute affects the bubble's masking and it is recommended to use a corner radius which at maximum equals to half of the bubble's height. Setting a corner radius larger than half of the bubble's height will cause the text to cut visually.	8
`userBubbleTopRightCornerRadius:Float = 8`	Top right radius corner on the user bubble	8
`userBubbleBottomLeftCornerRadius:Float = 8`	Bottom left radius corner on the user bubble	8
`userBubbleBottomRightCornerRadius:Float = 0`	Bottom right radius corner on the user bubble	0
`remoteUserBubbleTopLeftCornerRadius:Float = 8`	Top left radius corner on the remote bubble	8
`remoteUserBubbleTopRightCornerRadius:Float = 8`	Top right radius corner on the remote bubble	8
`remoteUserBubbleBottomLeftCornerRadius:Float = 0`	Bottom left radius corner on the remote bubble	0
`remoteUserBubbleBottomRightCornerRadius:Float = 8`	Bottom right radius corner on the remote bubble	8

Unread messages badge

The following properties for the unread messages badge can be configured:

Name	Description	Default
`unreadMessagesCornersRadius :Float = 8`	Sets the corner radius of the unread messages cell	8

Scroll to bottom button

The following properties for the scroll to bottom button can now be configured:

Name	Description	Default
`scrollToBottomButtonCornerRadius:Float = 20`	Sets the top left and bottom left radius of the scroll to bottom button	20
`scrollToBottomButtonBadgeCornerRadius:Float = 12`	Sets the radius of the scroll to bottom badge corners	12

Send button

The following properties for the send button can now be configured:

Name	Description	Default
`sendButtonImage: UIImage`	Custom send button image in the conversation screen	Arrow icon

Controller bubble

The following properties for the controller bubble can now be configured:

Name	Description	Default
`controllerBubbleTextColor : UIColor`	Color code for the text of the controller bubble	UI color

New APIs

Conversation interactions API

```
func getInactiveUserInteractionTimeInterval(_conversationQuery:
ConversationParamProtocol) -> TimeInterval
```

- Returns: Inactive TimeInterval (Double)

Unread messages badge API

```
func getUnreadMessagesCount(_conversationQuery: ConversationParamProtocol, completion:
@escaping (_ badgeCounter: Int)->(), failure: @escaping (_ error:NSError)→())
```

Logout API

```
func logout(completion: @escaping ()->(), failure: @escaping (_ error: Error)→())
```

New Callbacks

Reconnect attempt termination callback

```
func LPMessagingSDKConnectionRetriesFailed(_error: NSError)
```