Tag'by SDK for iOS

Version 1.00

Tag' by

contact@tagby.com – http://www.tagby.com

Table of contents

1. **Revision history** ____ 3

2. **Introduction** ____ 4

3. **System requirements** ____ 5

   3.1. iOS version and hardware ____ 5

   3.2. Software Requirements ____ 5

      3.2.1. Tag'by Launcher ____ 5

      3.2.2. Xcode version ____ 5

4. **Common project settings** ____ 6

   4.1. URL type ____ 6

   4.2. Fonts ____ 6

   4.3. Tag'by SDK files ____ 7

5. **SDK structure** ____ 8

   5.1. TBSocialNetworksPost class ____ 8

   5.2. TBSocialNetworksUser class ____ 8

   5.3. TBSocialNetworksManager class ____ 8

6. **Document/offer structure** ____ 9

   6.1. TBDocument class ____ 9

   6.2. TBSettings class ____ 9

   6.3. Widget and widget controllers ____ 9

1. Revision history

Document	Date	Description
1.00	8th Dec 2014	Initial release

2. Introduction

This document explains how to use the Tag'by SDK on iOS devices.

3. System requirements

3.1. iOS version and hardware

The Tag'by SDK supports iOS devices with the iOS operating system version 7.0 and newer. It has been tested on the following terminals:

- iPhone 5S (**iOS 7.1.1**)
- iPhone 5S (**iOS 8.1.1**)
- iPhone 6 (**iOS 8.1.1**)
- iPad Mini (**iOS 7.1.2**)
- iPad (4th generation) (**iOS 8.1.1**)

Inquiries about compatibility can be sent to Tag'by technical support: contact@tagby.com

3.2. Software Requirements

3.2.1. Tag'by Launcher

The example provided with the Tag'by SDK for iOS will need to use the Tag'by Launcher application to get some information required to communicate with the Tag'by backend services. Please make sure it is installed on your test devices and then on the final ones.

The software is available on App Store under the name of Tag'by Launcher.

3.2.2. Xcode version

The example provided with the Tag'by SDK was created with Xcode v6.1.1.

4. Common project settings

4.1. URL type

All applications using the Tag'by SDK for iOS need to communicate (only once, at launch time) with Tag'by Launcher. In order for Launcher to recognized and go back to the application, a URL type will need to be specified.

This setting is present in the Info.plist file of the project. We used com.example.demoapp as URL identifier (it has to match the application package on the server) and tagbyexample as URL scheme. These two values are used in the application delegate when calling the OpenURL method.

4.2. Fonts

Templates provided to the application with the help of the SDK can include a certain number of fonts that are not directly available in iOS. In order to support them, the application will need to:

- include TagbySDK.bundle as a dependency
- specify fonts in the plist file

There are 19 fonts and we included all of them in the Info.plist file. The final application will need to include the same setting.

4.3. Tag'by SDK files

The Tag'by SDK consists of two files:

- TagBySDK.framework (framework)
- TagbySDK.bundle (resources bundle)

These files will need to be integrated in any Xcode project using the Tag'by SDK for iOS.

5. SDK structure

5.1. TBSocialNetworksPost class

This class includes a social network post contents that will be filled in by the application. All elements are optional, they will be filled by the backend if are missing:

- the `title` property corresponds to the post title
- the `message` property corresponds to the message text
- the `image` property corresponds to the image

Usage example in the test application: `TBFinalShareViewController` (`didClickShareNow` method).

5.2. TBSocialNetworksUser class

This class corresponds to a social network user whose identity will be used to post messages on social networks. Some of its properties can be used on application GUI (like the URL of the user's profile picture).

Usage example in the test application: share view controllers.

5.3. TBSocialNetworksManager class

This class can be used to implement the following operations :

- returning a URL to be used in a web view to register a user manually to a social network
- checking if a user tag is associated to an account existing on the Tag'by server
- posting a message on a social network
- posting Analytics events

All of the above methods are used in the example application with comments.

6. Document/offer structure

Applications using Tag'by SDK for iOS will receive a document per every offer that is available to them. Elements/widgets shown on the GUI when the offer view is active are read from the document.

6.1. TBDocument class

This class contains the contents of an available offer that has been received from the Tag'by backend. On can distinguish the following elements in it:

- background (either a color or a URL to an image)
- settings (described below)
- social networks available for the offer (Facebook, Twitter, Email)
- offer (including its preview image, name and guid)
- an array of widgets (described below)

In the test application the `TBOfferRefresher` to refresh offers available to the application.

6.2. TBSettings class

This class is associated to an instance of the `TBDocument` class. It contains several options related to an offer, but not directly managed by widgets:

- screen orientation supported by the offer (by default, iPad applications work in landscape and iPhone ones in portrait, but it is up to the final application)
- whether the lock screen should be blocked or not when showing the offer
- whether barcodes or NFC tags (not yet supported) can be used to detect user tags
- messages shown when sharing and right afterwards
- five options related only to SlotMachine

6.3. Widget and widget controllers

As indicated in the document class description, an offer includes widgets that will be shown on the main offer screen. There are seven possible widget types:

- image (it includes a URL to a remote image)
- logo image (also an image, but more specific to be able to distinguish logos from images)
- label
- button
- YouTube view
- barcode preview

- camera preview

Every widget is obtained via its controller (there is one controller type per widget). As can be see in the example application, the widgets are laid out in the Z order (the bottomless widgets should be adde first). Widgets are put on the view when their controllers are initialized with the superview.

Every widget's frame is set by the controller and is based on the information received from the Tag'by backend.

a) Image widget

This widget includes a URL on which the destination image is available. It is its only property.

b) Logo image widget

This widget includes a URL on which the destination image is available. It is its only property. This widget type was created to distinguish images from logos.

Both image and logo image widgets are controlled by image controllers. The image controller provides two additional callbacks:

- whenever the remote image has been loaded
- when the image widget has been clicked

c) Label widget

Labels include the following properties:

- text
- font name (please remember to include the Tag'by bundle in the final application)
- font size
- text color
- text alignment

d) Button widget

Buttons include the following properties:

- text
- image URL
- font name (please remember to include the Tag'by bundle in the final application)
- font size
- text color

Button controller provide a callback that is called whenever the related button widget is pressed.

e) Barcode preview widget

This widget is used on share view controllers when user tags are detected. Its view controller allows to set the following properties:

- camera side
- activating/deactivating the camera

The barcode preview controller provides a callback that is called every time a user tag has been detected.

f) Camera preview widget

This widget is used in the Photobooth application and it allows taking photos on the terminal. Its controller allows to set the following properties:

- camera side
- activating/deactivating the camera

The takePicture method can be used to capture photos. The Camera preview widget controller provides an event that is called whenever a photo has been taken.

g) YouTube view widget

This widget can be used to show YouTube videos. Its controller can be used to play, pause and stop the video. The controller also provides an event that is called whenever video information has been received (to be able to read the thumbnail, for example).