Truepic Inc.

iOS SDK API, V2.0 build x80

Welcome to the API documentation for the Truepic iOS SDK. The Truepic SDK is a customizable camera library that you can easily integrate into any existing iOS application. It uses a variety of security and verification measures to guarantee authenticity for captured media, as processed by Truepic with results made available to your application via a simple APIs.

Compatibility

The current version of the Truepic SDK is 2.0, and compatible with iOS versions 10.x and later. Devices compatible with iOS 10+ include

- iPhone 5 and later iPhones
- iPad mini 2 and later
- 5th generation iPads and later
- all iPad Air and iPad Pro devices.

The iOS SDK is implemented as a Framework that exposes public classes and funcs through a Swift API. If you need to call the Truepic API from Objective C, please contact us so we can help ensure the process is as simple and painless as possible.

Integrating the Truepic framework

1. Add the Truepic framework to your Xcode project

Add Truepic.framework to the Embedded Binaries list on the General tab of your app target in Xcode.

2. Add the path to the ModuleMap directory

Note: If you already have CommonCrypto installed (typically in usr/bin from XCode terminal commands installation), your project build may fail due to duplicate instances of CommonCrypto. To fix, simply remove the ModuleMap directory and ignore this section.

Add the path to the ModuleMap directory to the `SWIFT_INCLUDE_PATHS` build setting. For example, if the ModuleMap directory is put in the root of the target's source directory, the value would be: `$(SRCROOT)/$(TARGET_NAME)/ModuleMap` (assuming that directory exists and hasn't been renamed).

3. Add required keys to your app's Info.plist

The Truepic framework requires access to the device's camera, location services, photo library, and microphone to take photos and videos, validate locations and record audio in videos. Your app must provide the corresponding usage descriptions for each of those permissions in its Info.plist. These descriptions are displayed in the alert presented to the user when the Truepic framework requests access to these services.

The Info.plist file within the Truepic.framework contains examples of the required keys, you can simply copy them into your app's Info.plist. The required privacy keys are:

```
NSCameraUsageDescription // Allow camera use

NSLocationWhenInUseUsageDescription // Allow location access when in use

NSMicrophoneUsageDescription // Allow microphone usage for video recording.

NSPhotoLibraryUsageDescription // Access to camera roll to save failed captures
```

4. Initialize SDK at app launch

Import the Truepic framework, and within your app delegate's didFinishLaunchingWithOptions method set its apiKey and appID. You'll also want to call TUploadClient.restartUploads() in two locations.

- The first time is in `didFinishLaunchingWithOptions` (passing nil for the completion routine) to restart any uploads paused if the app was force quit by the user.
- The second time is in UIApplication.handleEventsForBackgroundURLSession(), when iOS passes a completion handler to clean up when uploads are finished.

```
import TruepicSDK

func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
	…
	// Set SDK API key and app ID
	TruepicConfig.setAPIConfig(apiKey: apiKey, appID: appID)

	// Restart pending uploads.
	TPUploadClient.restartUploads(nil)
	return true
}

// Save completion handler for background URL sessions.
func application(_ application: UIApplication,
handleEventsForBackgroundURLSession identifier: String, completionHandler:
@escaping () -> Void) {
	// Pass iOS uploads completion handler to SDK.
	TPUploadClient.restartUploads(completionHandler)
}
```

5. Set the Truepic framework's userID and userName

When a user of your app logs in, pass their userID and userName to the Truepic framework. All photos and videos that the user uploads will be tagged with these values.

```
TruepicConfig.setUserInfo(userID: userID, userName: userName)
```

6. Present the Truepic Camera view

When you are ready to present the view for your users to upload Truepic photos and videos, get an instance of a TruepicViewController and use the present() method from an application view controller to display it:

```
let truepicController = TruepicConfig.truepicViewController()
self.present(truepicController, animated: true, completion: nil)
```

Gallery Items API

The Truepic Gallery API provides application access to display, manage and track progress for photos and videos captured within the Truepic SDK Camera view. Every capture is stored within the Gallery Items database until deleted, and the Gallery Items API allows you to query each captures current status and state values

Gallery Item Class, Types, and States

The gallery item class encapsulates all the status and state values for each gallery item. Review the Capture Lifecycle in the Truepic SDK Overview document for a detailed overview of the meaning of each state.

```
enum TPGalleryItemType: String {
	case photo, video, notSet
}

// The various states a gallery item can take during the verification/upload process
enum TPGalleryItemState: String {
case started,            // Created
validated,               // Metadata validated
failedValidation,        // Server rejected metadata
failedValidationUpload,  // Could not connect to server to validate metadata
uploadInProcess,         // Started/retrying upload
completed,               // Media upload succeeded after metadata validation.
failedUpload             // Failed upload (could not retry?)
}

class TPGalleryItem {
	var created: Date()           // Creation time stamp.
	var type: TPGalleryItemType
	var state: TPGalleryItemState
	var fileName: String
	var verificationCode: String? // Server code for successful pre-verification
	var mediaID: Int64?           // Server identifier for successful uploads
	var error: String?            // Failed verification/upload error.
	var uploadProgress: Double?   // Starts at 0.0, complete at 1.0.
}
```

Gallery Item Functions

Get Gallery Items

```
TPGalleryMgr.shared.gallery(types: [TPGalleryItemType], states: [TPGalleryItemState]?,
lastSessionOnly: Bool, newestFirst: Bool) -> [TPGalleryItem]
```

Return array of gallery items, optionally filtered to match specific characteristics. Default parameters return all gallery items entirely unfiltered, i.e. TPGalleryMgr.shared.gallery(). To filter and sort the returned array, customize the following parameters.

types: array of types to filter for, ie [*.photo]*, [*.video]* or both.

states: nil to ignore states when filtering, or an array of states to filter for. Example: [.completed] will return all successfully uploaded items.

newestFirst: *true* to sort newest items first, *false* to sort oldest items first.

Item URLs

```
TPGalleryMgr.shared.itemURL(item: TPGalleryItem) -> URL?
```

Return URL for specified gallery item (or nil if deleted).

Item Thumbnails

```
func TPGalleryMgr.shared.getItemThumbnail(_ item: TPGalleryItem) -> UIImage?
```

Return thumbnail image for video or photo items, or nil if could not be created.

Deleting Items

```
TPGalleryMgr.shared.deleteItems(types: [TPGalleryItemType] = [.photo, .video], states:
[TPGalleryItemState]? = nil) -> [TPGalleryItem]
```

Deletes items from gallery database and from local directory. Default parameters are set to completely unfiltered. For example, TPGalleryMgr.shared.deleteItems() deletes all local items.

types: array of types to filter for, ie [.*photo]*, [.*video]* or both.

states: nil to ignore states when filtering, or an array of states to filter for. Example: [.completed] will deleted all successfully uploaded items.

Returns array of remaining gallery items to facilitate updating UI.

```
TPGalleryMgr.shared.removeItemBy(verification_code: String, permanent: Bool) ->
NSError?
```

Delete item matching verification_code from gallery database and local directory. Pass permanent if the item is also being removed from server database.

Returns error if it can't be found or deleted.

Fetch Remote Media

A user's authenticated Truepic media can be fetched using the framework. The results of the fetch contain the media data, media type, and date the media was uploaded. The following shows and example of fetching a user’s Truepic media:

```
RemoteTruepicMedia.fetchMediaFor(userID: userID, appID: appID, apiKey: apiKey) {
(result: Result<[TruepicMedia]>) in
        switch result {
        case .success(let media):
                guard let firstMedia = media.first else { return }
                switch firstMedia.receipt.type {
                case .image:
                        let image = UIImage(data: firstMedia.media)
                        let mediaDate: Date = firstMedia.receipt.date
                case .video:
                        break
                }
        case .error(let error):
                break
        }
}
```

Notification Messages

The Truepic SDK sends the following notifications for important events.

`TruepicItemUpdated`

A gallery item has changed state, such as completed initial verification, upload, or failed. If the item is being uploaded the notification is sent whenever it’s progress value has been updated. The changed gallery item is passed in the notification object field.

`TruepicViewClosed`

The SDK camera view has been closed by the user. There is no data associated with the notification.

`TruepicOpenGallery`

The user tapped the gallery button within the camera view. This closes the camera view, and the expectation is the host application will display it's gallery view. There is no data associated with the notification.

Configuration Options

The following options configure the Truepic framework. Setting them is not required to use the framework.

Camera View Options

`TruepicViewController.CameraViewOptions` provides the following options for configuring the presentation of the camera view.

`TruepicViewController.CameraViewOptions.disableCameraSelection`

Disables the front and rear facing camera toggle button.

`TruepicViewController.CameraViewOptions.frontCameraIsDefault`

Set front camera as the default.

`TruepicViewController.CameraViewOptions.videoCaptureIsDefault`

Set the default media capture mode to video (instead of photo).

`TruepicViewController.CameraViewOptions.disableMediaSelection`

Disables the media selection button on the camera view.

Using Camera Options

To change options, set CameraViewOptions to an array of the options you want:

```
TruepicViewController.CameraViewOptions = [.frontCameraIsDefault,
.disableCameraSelection]
```

To test option settings, use contains():

```
TruepicViewController.CameraViewOptions.contains(.disableCameraSelection)
```

Since not all devices support all options (e.g. rear camera, video capture), CameraViewOptions provides deviceAvailableOptions() to return all options available for the current device.

```
TruepicViewController.CameraViewOptions.deviceAvailableOptions()
```

Other Configuration Options

Include additional information with uploads

A dictionary of upload info to accompany uploads can be added. This can be useful for tagging captures to match specific events, such as insurance claims, to make it easier for back-end systems to link to them.

The dictionary type is a `[String: Any]`, and must be convertible to a valid json dictionary or it will not be included with uploads. Example:

```
let uploadInfo: [String : Any] = ["key1": "value", "key2": 0]
let isValidJson = TruepicConfig.setUploadInfo(uploadInfo)
if !isValidJson {
// uploadInfo is not a valid json dictionary
}
```

Save media locally

Media captured by the Truepic framework can be saved locally either to the devices Photo Library, or to the documents directory of the app.

Saving media locally is disabled by default. In general, we recommend leaving it disabled. Due to the size of captured media and the limited amount of disk space on iOS devices, saving media locally can use up disk space quickly. Note: Failed captures are always saved to the devices Photo Library.

To save media locally, set the option to select the save destination:

```
let localSaveDestination: LocalSaveMediaDestination = .photoLibrary
                                  // or .documentsDirectory or nil to disable
TruepicConfig.setSaveMediaLocalDestination(localSaveDestination)
```

The Photos framework posts notifications when changes are made to the user's Photo Library. Implement at `PHPhotoLibraryChangeObserver` to receive the changes.