Truepic Inc.

Android SDK API

v2.0.6

Welcome to the API documentation for the Truepic Android SDK. The Truepic SDK is a customizable camera library that can easily integrate into any existing Android project. It uses a variety of security and verification measures to guarantee authenticity for your captured media, which is then processed by us and made available to you via a simple API.

Technical Overview

Implemented Feature Set

- Camera UX with authenticated image / video capture, tap to focus, pinch to zoom, front and rear camera support
- Save media to local gallery and/or post to Truepic servers for remote verification & authenticity checks
- Ability to upload additional parameters or metadata with each media item
- Configurable real-time verification state checks based on smartphone sensor and environmental data:
    - Geolocation accuracy
    - Network connectivity
    - Rootkit / jailbroken devices
    - Developer mode
- Gallery API for querying and managing previously captured media

Device Requirements and Supported Android versions

Currently the SDK supports any device running Lollipop (Android API 21) and newer. On any devices running a version of Android lower than 21 (as low as KitKat), the camera behavior will default to opening your default camera application, to prevent disrupting any existing photo capture pipelines, but will not support Truepic verification, nor additional camera input settings.

Importing the SDK into your project

AAR

If you are importing the Truepic SDK .AAR, then you can use the following steps in Android Studio to import the SDK.

- Open module settings (for the module that you want to depend on the SDK)
- Click 'New Module' or the (+) icon
- Select 'Import .JAR/.AAR package option
- Browse to the local location of the Truepic SDK JAR/AAR and select it, click OK to finish
- Your dependency list in your module's build.gradle should be updated to include something like this (needs to match your actual .AAR file name)
    - `dependencies  { compile project(":truepic-sdk") }`

Support library dependencies

Currently, the Truepic Android SDK depends upon the appcompat-v7, v13 support library and the new EXIF support library for backwards compatibility. In addition we rely on the open source library RootBeer which provides comprehensive root detection on Android. Below shows how to add them to your applications build.gradle dependencies list. **Note:** To avoid conflicts we recommend using the latest versions whenever possible (which Android Studio will highlight for you to upgrade).

Add the following to your build.gradle file to the application module of your project:

```
dependencies {
      implementation 'com.android.support:appcompat-v7:27.1.1'
      implementation 'com.android.support:support-v13:27.1.1'
      implementation 'com.scottyab:rootbeer-lib:0.0.6'
      implementation 'com.android.support:exifinterface:27.1.1'
}
```

Required Permissions

The Truepic SDK declares multiple permissions in its Manifest, but doesn’t require any action for integration, but they are required for the SDK to properly operate. These will merge into the final host applications Manifest when building your project.

- `android.permission.CAMERA`
- `android.permission.WRITE_EXTERNAL_STORAGE`
- `android.permission.READ_EXTERNAL_STORAGE`
- `android.permission.RECORD_AUDIO`
- `android.permission.ACCESS_LOCATION`
- `android.permission.ACCESS_COARSE_LOCATION`
- `android.permission.ACCESS_FINE_LOCATION`
- `android.permission.INTERNET`
- `android.permission.ACCESS_NETWORK_STATE`

Media Capture

Starting the Camera

To open the Truepic camera SDK from your project, declare an Intent that launches the Truepic SDK. The only required parameters are your API key and Application ID values, which should be passed in as extras and provided to you by Truepic. If you are not interested in the local media path data after capture, you can simply use startActivity(). Note: If you pass in an invalid value for either your API key or application ID, then any attempted uploads will fail, but it will not prevent the camera from functioning locally and saving media to your camera roll.

After the camera capture session is complete, developers can opt-in to receive the local file paths of all captured media) or utilize the gallery API. If you opt for remote verification, the SDK gallery API interface can return the verification codes for each of your verified media captures stored on our servers. See below to find out more about how to customize settings.

Sample Code

This example assumes this code is called from an Activity where ‘this’ would be the activity reference. The return code is a custom value set by your application, which can be used to process data returned in onActivityResult().

```
Intent truepicSdkIntent = new Intent(this, TruepicSdk.class);
truepicSdkIntent.putExtra(TruepicSdk.EXTRA_API_KEY, <your_api_key_string>);
truepicSdkIntent.putExtra(TruepicSdk.EXTRA_APP_ID, <your_app_id_string>);

// Start the SDK using transitions to animate enter and exit
// (or any startActivity API)
ActivityCompat.startActivityForResult(
      this,
      truepicSdkIntent,
      <YOUR_REQUEST_CODE>,
      ActivityOptions.makeSceneTransitionAnimation(this).toBundle());
```

Configuring the Camera based on EXTRA settings

Here is a list of the different settings that can be passed into the SDK upon opening it:

EXTRA_ACCURACY_THRESHOLD_METERS: An optional integer parameter that can be used to set the accuracy threshold in meters for when Truepic verification is enabled/disabled. The default value is 300 meters.

EXTRA_USER_ID: An optional user ID parameter that can be provided to group uploads to a specific feed

EXTRA_USERNAME: An optional username parameter that will be used on our back-end services to tag your media uploads with a username

EXTRA_SQUARE_ASPECT: Forces both a square preview as well as the output media to be square dimensions

EXTRA_DISABLE_CAM_SELECT: Removes the UI that allows swapping between front-facing and rear facing camera (i.e. disable front facing)

EXTRA_FRONT_DEFAULT: The default camera or video will open in front facing mode (if there is a front facing camera)

EXTRA_VIDEO_DEFAULT: Whether or not the SDK should start in video mode

EXTRA_DISABLE_MEDIA_SELECTION: Disable changing between the camera and video camera

EXTRA_SAVE_TO_GALLERY: When set to true, captured output media will be stored to the device's PICTURES directory and made available to the media gallery by scanning it. When set to false, we will still save the image to the device, but it will not be made available to the gallery.

EXTRA_SAVE_FLASH: When set to true the flash setting will be remembered between SDK sessions, false by default (starts in auto-flash mode each time)

EXTRA_VIDEO_LENGTH_MAX: Set the maximum length of video (in seconds) before the video will automatically stop recording

Sample Code

When starting the SDK, you can pass in parameters to the Activity as EXTRAS to configure the default functionality. This is done by using any of the aforementioned extras / SDK constants. All of the following extras are currently booleans.

```
Intent truepicActivity = new Intent(this, TruepicSdk.class);

// attach the username and pass as extras
// also attach api-key and app-id
truepicActivity.putExtra(TruepicSdk.EXTRA_USERNAME, username);
truepicActivity.putExtra(TruepicSdk.EXTRA_USER_ID, userId);
truepicActivity.putExtra(TruepicSdk.EXTRA_API_KEY, API_KEY);
truepicActivity.putExtra(TruepicSdk.EXTRA_APP_ID, APP_ID);

// Declare an optional preference array if we wanted to
// set additional preferences or configurations
if (mPreferences != null && mPreferences.length > 0) {
    truepicActivity
           .putExtra(TruepicSdk.EXTRA_SQUARE_ASPECT, mPreferences[0])
           .putExtra(TruepicSdk.EXTRA_DISABLE_CAM_SELECT, mPreferences[1])
           .putExtra(TruepicSdk.EXTRA_FRONT_DEFAULT, mPreferences[2])
           .putExtra(TruepicSdk.EXTRA_VIDEO_DEFAULT, mPreferences[3])
           .putExtra(TruepicSdk.EXTRA_DISABLE_MEDIA_SELECTION,mPreferences[4])
           .putExtra(TruepicSdk.EXTRA_SAVE_TO_GALLERY, mPreferences[5])
           .putExtra(TruepicSdk.EXTRA_SAVE_FLASH, mPreferences[6]);
}
```

Passing additional, optional upload parameters

We allow the caller of the SDK to pass additional String parameters to be stored remotely along with the media captured through the static API 'setParams(HashMap<String, String>). They will be available when querying server-side media later. These are optional String key and value pairs. Parameters must be passed in using a HashMap<String, String>. Here is an example of how to add 3 additional extra parameters (you can provide as many or as little as you would like). Pictures or videos captured after setParams() is called will have them attached.

Sample Code

```
HashMap < String,String > params = new HashMap < >();
params.put("Truepic-param-1", "123456");
params.put("Truepic-param-2", "TestParam123");
TruepicSdk.setParams(params);
```

Gesture Support

We support a few types of built-in gesture support to manipulate capture functionality at runtime (in both camera and video modes):

- Tap-to-focus
- Pinch-to-zoom

Querying Captured Media

There are three types of information you might be interested in polling once a user has completed a capture session with the SDK:

- The absolute file paths of locally captured media
- The verification codes and status of media remotely uploaded to truepic servers
- The internal SDK database record for media (via the Gallery API)

Locally Captured Media

To receive a list of local, absolute file paths to any photo or video captured during a Truepic SDK session, you must do two things. First, you have to start the Truepic SDK using **startActivityForResult**, such as:

```
startActivityForResult(this, truepicSdkIntent, <YOUR_REQUEST_CODE>,
       ActivityOptions.makeSceneTransitionAnimation(this).toBundle());
```

`YOUR_REQUEST_CODE` can be any identifying integer you'd like it to be. You must also override onActivityResult from the calling Activity with the same code, which will be called after the SDK has shutdown. Here's an example of how to override the method and parse (if any) captured media:

```
@Override protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
 if (requestCode == REQUEST_CODE) {  // REQUEST_CODE same for startActivity
      if (resultCode == RESULT_OK) {
            ArrayList<String> mediaPaths =
                  data.getExtras().getStringArrayList(TruepicSdk.RESULT_EXTRA_
                  CAPTURED_MEDIA);
  } }
```

Uploaded Media

If you are interested in receiving the verified codes for each uploaded media, you must declare a broadcast receiver and register it using the LocalBroadcastManager, as well as unregister it in the enclosing Activity. Here's an example of declaring a receiver and registering it locally. Every media captured and uploaded will send a broadcast containing the status (true or false) and a message.

```
// Register a handler for to receive Truepic Intents.
private BroadcastReceiver mVerifiedReceiver = new BroadcastReceiver()  {
      @Override
      public void onReceive(Context context, Intent intent) {
      // Get extra data included in the Intent
      if (intent.getAction().equals(TruepicSdk.ACTION_VERIFIED_TRUEPIC)) {
            boolean verifySuccess =
                  intent.getBooleanExtra(TruepicSdk.EXTRA_VERIFIED_STATUS,
                  false);

            String verifyMsg =
                  intent.getStringExtra(TruepicSdk.EXTRA_VERIFIED_MSG);
      }
};

// register custom receiver for verified events from Truepic in your oncreate
@Override
protected void onCreate(Bundle savedInstanceState) {
      LocalBroadcastManager.getInstance(this).registerReceiver(mVerifiedReceiv
      er, new IntentFilter(TruepicSdk.ACTION_VERIFIED_TRUEPIC));
}

// unregister the same receiver before you application closes
// example location could be onBackPressed (or any place before the app exits)
LocalBroadcastManager.getInstance(this).unregisterReceiver(mVerifiedReceiver);
```

Gallery API

The Truepic SDK maintains an internal database record of media captured during sessions. There are two main asynchronous APIs to interact with the results DB captured by the SDK:

```
TruepicSdk.getGallery(Context applicationContext, GalleryMediaType mediaType,
      GalleryUploadStatusType uploadStatusType, GalleryOrderType orderType,
      boolean rescanGallery, TruepicGalleryListener listener)

TruepicSdk.deleteGalleryItems(Context applicationContext,
      GalleryUploadStatusType  filterStatus, String filterVerifCode,
      TruepicGalleryListener listener)
```

Both parameters are required and the first is a valid application context. The second parameter is an interface available in the Truepic SDK upon import. Below is the API for the TruepicGalleryListener API and other optional parameters and enums.

SDK Interfaces and Parameters for the Gallery

```
// All interfaces available in the Truepic. namespace

public enum GalleryMediaType {
      Image,
      Video
}

public enum GalleryUploadStatusType {
      Pending,
      Complete,
      Failed
}

public enum GalleryOrderType {
      NewestFirst,
      OldestFirst
}

public interface TruepicGalleryListener {
      void onGetCompleted(ArrayList<GalleryItem> output);
      void onDeleteCompleted(int count);
}
```

The **`TruepicGalleryListener`** interface is used for both getGallery() and deleteGallery(). You will receive the appropriate asynchronous callback when calling each respective API. In onGetCompleted() it will return a list of GalleryItems, each of which contains information about each upload (or empty if none have been taken yet). The onDeleteCompleted(int count) API returns the number of items deleted from the gallery, or 0 if none were, or an error occurred.

GalleryItem API (onGetCompleted())

When calling the **`getGallery()`** API with a valid **`TruepicGalleryListener`**, the onGetCompleted() interface will be called back with a list of **`GalleryItem.`** Below are the various methods and APIs that can be called on an individual **`GalleryItem.`**

`public String`	**`getFilename()`**	Returns the absolute path of the gallery item's media on device.
`public String`	**`getTimestamp()`**	Returns the time since epoch in milliseconds (long, converted to String).
`public String`	**`getStatus()`**	Returns a **`GalleryUploadStatusType,`** which is either 'Pending', 'Complete' or 'Failed' depending on the upload status, as a String.
`public String`	**`getExtraMessage()`**	Returns a description (if available) regarding the current status with as much detail as possible.
`public String`	**`getVerificationCode()`**	Returns the Truepic verification code after it has been inspected by our services. This value will be empty until the status has completed Pending. This value can then be tied to the remote server media.

Sample Code (getGallery() and deleteGallery())

```
// test the current gallery API
// gets ALL gallery items without filtering (by passing null, instead of enum)
TruepicSdk.getGallery(getApplicationContext(), null, null,null, false,
    new TruepicSdk.TruepicGalleryListener() {
        @Override
        public void onGetCompleted(ArrayList<GalleryItem> galleryItems) {
            for (int i = 0; i < galleryItems.size(); i++) {
                final GalleryItem curGalleryItem = galleryItems.get(i);
                Log.e(TAG," filename: " + curGalleryItem.getFilename()
                        + " timestamp: " + curGalleryItem.getTimestamp()
                        + " status: " + curGalleryItem.getStatus()
                        + " code: " + curGalleryItem.getVerificationCode()
                        + " msg: " + curGalleryItem.getExtraMessage()
                        + " media-type: " +
                          curGalleryItem.getMediaType().toString());
            }
        }

        @Override
        public void onDeleteCompleted(int i) {
            // not called when using getGallery()
        }
});

// perform a filtered, or unfiltered (by passing null), deletion against
// the SDK media records
TruepicSdk.deleteGalleryItems(
    getApplicationContext(),
    null,
    null,
    new TruepicSdk.TruepicGalleryListener() {
        @Override
        public void onGetCompleted(ArrayList<GalleryItem> arrayList) {
            // not called/empty since we're doing a delete
        }

        @Override
        public void onDeleteCompleted(int deleteCount) {
            Log.e(TAG, "delete completed, delete count: " + deleteCount);
        }
});
```

Updating the SDK

If you are importing the latest Truepic SDK AAR, but you currently have an older version configured in your project-module, then you can follow the steps below to update the latest build into your project and replace the older version.

- In Android Studio, open the (Module Settings) for the module in your project that depends on the Truepic SDK
- Remove the 'truepic-sdk' module and click OK (this will not remove anything on disk, it only removes the dependency in Android Studio
- Open the root of your Android project in your file explorer and delete the truepic-sdk folder on disk (you can't re-import the new one without doing this)

Now you can re-import the SDK just as if you were performing a fresh install. Please see the steps above regarding how to import the SDK to your project and modules

Features Coming Soon

- (End of 2018) Remote distribution via product page
- (End of 2018) Differentiated capture experiences based on subject media

Contact & Support

matt@truepic.com