element

Element Face SDK

The Element Face SDK is an API library to create the biometrics models that can be used to authenticate users. This document contains information to integrate the Element Face SDK into an Android application by using Android Studio.

Version Support

- The Element Face SDK supports Android 5.0+ / API 21+ (Lollipop and up)
- Android Studio 3.1.3 with Gradle 4.4
- Last Target SDK Version: 27
- Last Compiled Support Library Version: 27.1.0
- Last Compiled Google Play Services Version: 15.0.1
- AWS Mobile SDK: 2.6.+
- Google Guava for Android: 23.5

Prerequisites

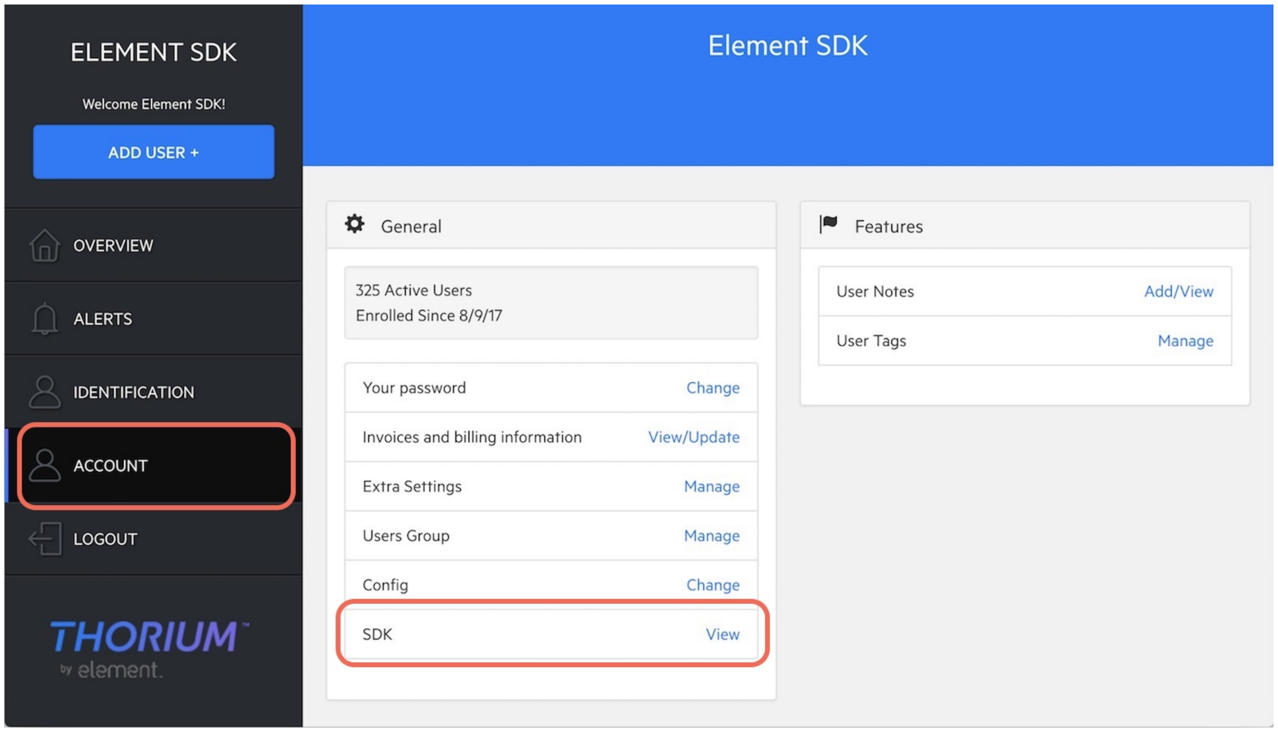
Element Dashboard

The Element Dashboard is the gateway to the assets in order to use the Element Face SDK. The URL of the Element Dashboard varies based on your region. Also an account is required to access the Element Dashboard. Please contact Element for more information.

AAR

The Element Face SDK is in the AAR format. Download the AAR:

1. Log into the Element Dashboard with your account.
2. Select `Account` tab in the left navigation bar.
3. Find the menu item named `SDK`, click the `View` button next to it.

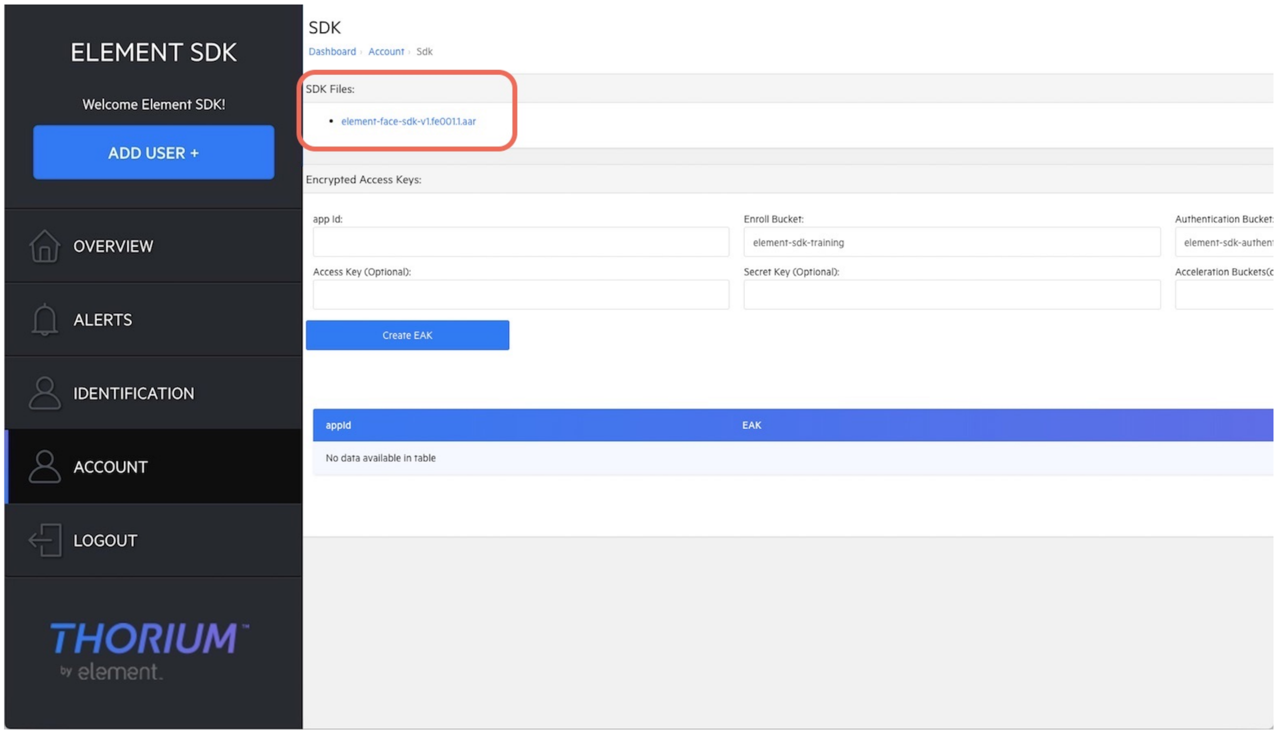


325 Active Users
Enrolled Since 8/9/17

Your password	Change
Invoices and billing information	View/Update
Extra Settings	Manage
Users Group	Manage
Config	Change
SDK	View

User Notes	Add/View
User Tags	Manage

4. Under `SDK Files` section, click the SDK download link and save it to the desktop of your computer.



SDK
Dashboard › Account › Sdk

SDK Files:
- element-face-sdk-v1.fe001.1.aar

Encrypted Access Keys:

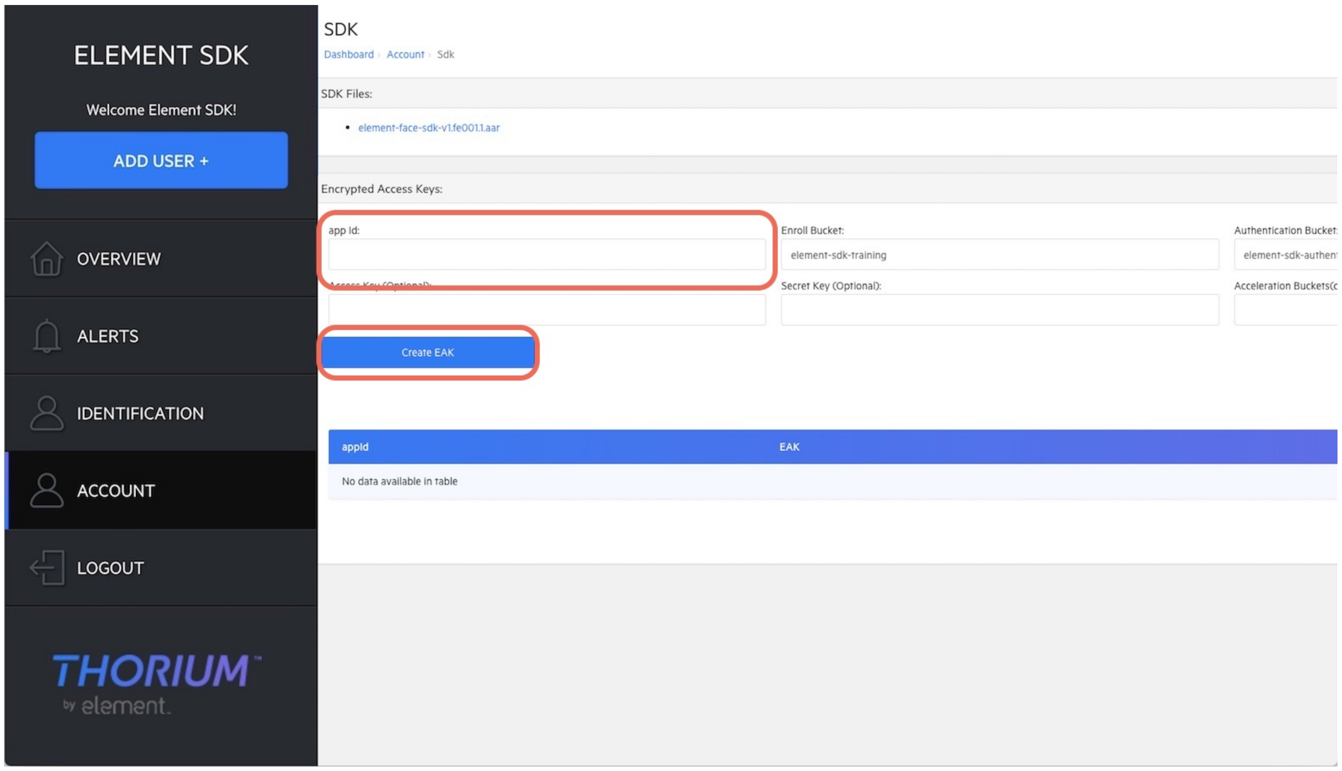
app Id:
Enroll Bucket: element-sdk-training
Authentication Bucket: element-sdk-authen
Access Key (Optional):
Secret Key (Optional):
Acceleration Buckets(c

appId	EAK
No data available in table	

Register the Application Id (App Id) to Element and obtain the Encrypted Access Key (EAK)

The Element Face SDK requires the *Encrypted Access Key* (*EAK*) file. The *EAK* file carries encrypted information including the Application Id (App Id) of your Android app. Your registered *EAK* is available on the Element Dashboard, under `Account -> SDK`. Here is how you get a new *EAK* file:

1. On the same page of where you download the SDK file, fill in the `App Id` field with your `application id`. You can find your `application id` in your module-level `build.gradle` file. Leave other fields unchanged and click `Create EAK`.
2. You new EAK will be listed on the page. Hover your mouse on the EAK you want to download and a little download icon will appear next to your `app id`. Click it, name the file `element.eak` and save it to the desktop of your computer.



SDK Files:

- element-face-sdk-v1.fe001.1.aar

Encrypted Access Keys:

app Id:

Enroll Bucket: element-sdk-training

Authentication Bucket: element-sdk-authen

Access Key (Optional):

Secret Key (Optional):

Acceleration Buckets(

appId	EAK
No data available in table	

Setup with Android Studio

Import the AAR

1. Open your project in Android Studio.
2. On the top menu bar, click `File -> New -> New Module`. In the `Create New Module` window, click `Import .JAR/.AAR Package`, then click `Next`.
3. In the next window, click the `...` next to `File name` field and select the AAR file in your computer's desktop directory. Then type in `element-face-sdk` in the `Subproject Name` field.

4. Click the `Finish` button and wait for Android Studio to finish building the project.

Add element-face-sdk and dependencies

1. Add `element-face-sdk` module to your project by adding the following code to the `settings.gradle` file in the root directory of your project:

   ```
   include ':element-face-sdk'
   ```

2. On the top menu bar, click `File -> Project Structure`.
3. Select your app module under `Modules` on the left pane, click on the `Dependencies` tab, and click on the `+` button at the bottom of the window. In the popup, click `Module Dependency` and select `:element-face-sdk`. Click `Ok`.
4. Add the following dependencies to the module-level `build.gradle`:

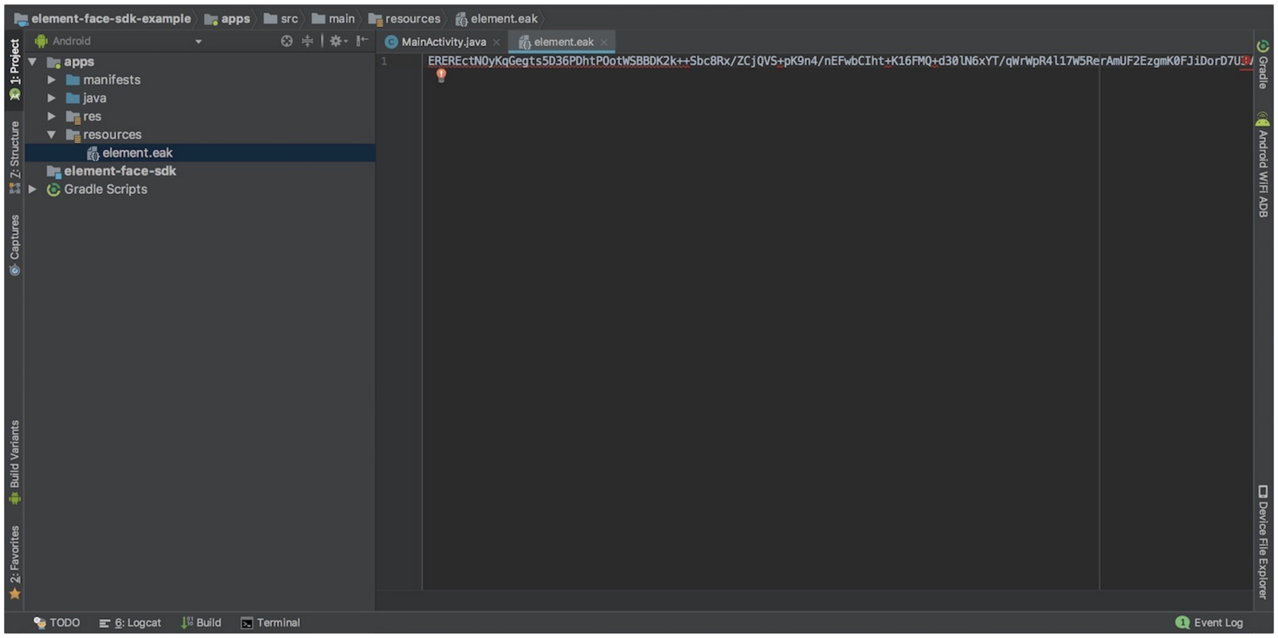
   ```
       dependencies {
         .....
         implementation 'com.android.support:appcompat-v7:27.1.0'
         implementation 'com.amazonaws:aws-android-sdk-core:2.6.+'
         implementation 'com.amazonaws:aws-android-sdk-s3:2.6.+'
         implementation 'com.google.android.gms:play-services-location:+'
         implementation 'com.google.guava:guava:23.5-android'
       }
   ```

   Note that you might have already declared some of these dependencies in your module-level `build.gradle` file, so please make sure you did not declare them twice. And you might also have to tweak a little bit on the versions of the dependencies as well as `compileSdkVersion` and `targetSdkVersion` in the `build.gradle`. Please follow the Android Studio's prompts on this. More information can be found here.
5. Wait for the Android Studio to sync.

Include the EAK in the application

1. Create a resources directory at `[project dir]/app/src/main/resources`.

2. Copy the `element.eak` file into the `resources` directory.



```
ERERECtNOyKqGegts5D36PDhtPOotWSBBDK2k++Sbc8Rx/ZCjQVS+pK9n4/nEFwbCIht+K16FMQ+d30lN6xYT/qWrWpR4l17W5RerAmUF2EzgmK0FJiDorD7U
```

Using the Element Face SDK APIs

Initialize the Element Face SDK

1. Create a class which extends android.app.Application if you haven't, and initialize the Element Face SDK in `onCreate()` method:

```
public class MainApplication extends Application {
  @Override
  public void onCreate() {
    super.onCreate();
    ElementFaceSDK.initSDK(this);
  }
}
```

2. Declare the `MainApplication` class in AndroidManifest.xml:

```
<manifest>
  .....
  <application android:name=".MainApplication">
    .....
  </application>
</manifest>
```

Ask for user permissions

1. The Element Face SDK requires the following permissions:

   - `android.Manifest.permission.CAMERA`
   - `android.Manifest.permission.ACCESS_FINE_LOCATION`
   - `android.Manifest.permission.ACCESS_COARSE_LOCATION` Those permissions are declared in the Element Face SDK AAR, so no need to declare them again in the manifest in your app.
2. The Element Face SDK provides `PermissionUtils.verifyPermissions(Activity activity, String... permissionsToVerified)` for requesting the permissions:

   ```
   PermissionUtils.verifyPermissions(
     MainActivity.this,
     Manifest.permission.CAMERA,
     Manifest.permission.ACCESS_FINE_LOCATION,
     Manifest.permission.ACCESS_COARSE_LOCATION);
   ```

   For the Android Marshmallow 6.0 (API 23) OS and up, make sure the permissions are granted before starting any Activity provided by the Element Face SDK.

User enrollment

The Element Face SDK utilizes the `ElementFaceEnrollActivity` for user enrollment. It's based on the `startActivityForResult`) method.

1. Enroll a user and obtain the `UserInfo` . The `UserInfo` contains a unique `userId` (ElementId). The pair of the userId and the appId (`context.getPackageName()`) is mainly used in the Element Face SDK to inquire the user's information and status. In the `Activity` where you want to start the enrollment process:

   ```
   UserInfo userInfo = UserInfo.enrollNewUser(
     getBaseContext(),
     getPackageName(),
     firstName,
     lastName,
     null);
   ```

2. Declare a request code:

   ```
   public static final int ENROLL_REQ_CODE = 12800;
   ```

3. Start the `ElementFaceEnrollActivity` :

   ```
   Intent intent = new Intent(this, ElementFaceEnrollActivity.class);
   intent.putExtra(ElementFaceEnrollActivity.EXTRA_ELEMENT_USER_ID, userIn
   startActivityForResult(intent, ENROLL_REQ_CODE);
   ```

4. Override the `onActivityResult`) method to receive the enrollment results:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
  if (requestCode == ENROLL_REQ_CODE) {
    if (resultCode == Activity.RESULT_OK) {
      // User enrolled successfully
    } else {
      // Enrollment cancelled
    }
  }
}
```

5. Declare the `ElementFaceEnrollActivity` in AndroidManifest.xml:

```
<manifest>
  .....
  <application android:name=".MainApplication">
    .....
    <activity android:name="com.element.camera.ElementFaceEnrollActivit
      android:clearTaskOnLaunch="true"
      android:hardwareAccelerated="true" />
    .....
  </application>
</manifest>
```

User authentication

User authentication is similar to user enrollment, using `ElementFaceAuthActivity` .

1. Declare the request code:

```
public static final int AUTH_REQ_CODE = 12801;
```

2. Start the `ElementFaceAuthActivity` :

```
Intent intent = new Intent(this, ElementFaceAuthActivity.class);
intent.putExtra(ElementFaceAuthActivity.EXTRA_ELEMENT_USER_ID, userInfo
startActivityForResult(intent, AUTH_REQ_CODE);
```

3. Override the `onActivityResult`) method to receive the authentication results:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
  if (requestCode == AUTH_REQ_CODE) {
    if (resultCode == Activity.RESULT_OK) {
      String results = data.getStringExtra(ElementFaceAuthActivity.EXTR
      if (ElementFaceAuthActivity.USER_VERIFIED.equals(results)) {
          // The user is verified
      } else if (ElementFaceAuthActivity.USER_FAKE.equals(results)) {
          // the user was spoofing
```

```
            } else {
                // The user is not verified
            }
          } else {
            // Verification cancelled
          }
        }
      }
```

4. Declare the `ElementFaceAuthActivity` in AndroidManifest.xml:

```
<manifest>
  .....
  <application android:name=".MainApplication">
    .....
    <activity android:name="com.element.camera.ElementFaceEnrollActivit
      android:clearTaskOnLaunch="true"
      android:hardwareAccelerated="true" />
    <activity android:name="com.element.camera.ElementFaceAuthActivity"
      android:clearTaskOnLaunch="true"
      android:hardwareAccelerated="true" />
    .....
  </application>
</manifest>
```

User enquiries

The Element Face SDK provides a few classes and methods to query user status.

ElementFaceSDK

- Find out if an user is enrolled

```
public static boolean isEnrolled(String userId)
```

ProviderUtil

- List alls users

```
public static List<UserInfo> getUsers(@NonNull Context context, @NonNull S
```

- Get an user

```
public static UserInfo getUser(@NonNull Context context, @NonNull String a
```

- Delete users

```
public static boolean deleteUser(@NonNull Context context, @NonNull String
public static void deleteAllUsers(@NonNull Context context, @NonNull Strin
```

- Update an user

```
public static int updateUserInfo(@NonNull Context context, @NonNull UserIn
public static void insertUserInfo(@NonNull Context context, @NonNull UserI
```

Questions?

If you have questions, please contact devsupport@discoverelement.com.