# Contributing

This is based on information seen [here](https://github.com/mgp25/Instagram-API/wiki/Technical-information).

#### Table of contents

1. [Common Errors](#common-errors)

2. [Capturing Endpoints](#capturing-endpoints)

3. [Reading the Signature Key](#reading-the-signature-key)

4. [Capturing MQTT (Notifications)](#capturing-tls-requests)

## Common Errors

### IgActionSpamError or feedback\_required

Occurs as the name suggests after a large amount of requests (in a short amount of time).

\*\*Fix:\*\* Add a delay between the requests and/or use proxies for multiple accounts.

### IgCheckpointError or challenge\_required

See [this](https://github.com/dilame/instagram-private-api/blob/master/examples/checkpoint.example.ts).

### login\_required

This is thrown whenever your session is invalid.

You can fix it by not loading the session/cookies.

Most of the time you'll face another error afterwards.

## Capturing Endpoints

In order to capture endpoints you have to monitor the requests using a proxy.

Instagram itself prevents that as anyone could monitor your requests, so you'll have to modify the app.

Currently, [this](https://github.com/itsMoji/Instagram\_SSL\_Pinning) project by [itsMoji](https://github.com/itsMoji)

allows you to disable the certificate-pinning which effectively makes the app accept every certificate.

\*\*Only use it in \_your\_ network as anyone with the certificate can then monitor your requests!\*\*

#### Step by step guide

The first steps are just for setting up the apk. You can also use [these](https://github.com/itsMoji/Instagram\_SSL\_Pinning#instagram-ssl-pinning) instructions.

1. \*\*Install\*\* either the patched apk provided by itsMoji [here](https://github.com/itsMoji/Instagram\_SSL\_Pinning/tree/master/non-root)

\*\*or\*\* manually patch the apk (requires root access)

by following [these](https://github.com/itsMoji/Instagram\_SSL\_Pinning#root-method) instructions.

2. \*\*Install\*\* a http proxy on your host machine.

- Currently, the only proxy able to decrypt \*\*TLS 1.3\*\* is [Burp 1.7.x](https://portswigger.net/burp/releasesarchive/community)

(use version 1.7 although it's outdated, 2.x \*\*won't work\*\*) with [Java 11 or above](https://www.oracle.com/technetwork/java/javase/downloads/index.html).

3. \*\*Configure\*\* your proxy to decrypt TLS 1.3 and \*\*export\*\* the \*root certificate\* to the phone/emulator.

4. \*\*Capture\*\* the requests.

#### General Infos

- A request looking like `signed\_body={HEX}.{Request}&ig\_sig\_key\_version=4` has to be signed.

## Reading the Signature Key

The signature key is used to sign requests.

#### Step by step guide

1. [Setup Instagram on your phone like this](#capturing-endpoints)

2. Install frida on your [device](https://www.frida.re/docs/android/) and [host machine](https://www.frida.re/docs/installation/).

3. Start the frida-server on your device

4. Connect to frida and the `com.instagram.android` process (using an emulator e.g. run `frida -U -n com.instagram.android`)

5. Run ```

fscrambler = Module.findExportByName("libstrings.so","\_ZN9Scrambler9getStringESs");

Interceptor.attach(ptr(fscrambler), {

onLeave: function (retval) {

send(Memory.readCString(retval));

}

});```

6. Force a signed request inside the app by for example liking an image.

7. You should now see `message: {'type': 'send', 'payload': '{SIGNATURE\_KEY}'} data: none`

## Capturing TLS Requests

MQTT and FBNS are currently not implemented in the main library as they're not ready.

You can see the current development [here](https://github.com/dilame/instagram-private-api/issues/845).

They are built using MQTT (v3 and for FBNS a custom implementation of v3) and [Thrift](https://thrift.apache.org/)

You'll probably have to use a Hex Editor for that.

Currently, the only way of capturing these packets is using another proxy.

#### Step by step guide

If you are using Burp, \*\*only\*\* enable the proxy on `127.0.0.1`.

1. [Setup Instagram on your phone like this](#capturing-endpoints)

2. \*\*Install\*\* [Charles](https://www.charlesproxy.com/download/) (the test version only supports 30min per session, so save your results in another editor).

3. \*\*Configure\*\* Charles as a \*\*SOCKS\*\* proxy (\*Proxy\* > \*Proxy Settings...\* > \*Enable SOCKS Proxy\*).

4. \*\*Export and install\*\* the certificate (\*Help\* > \*SSL Proxying\* > \*Install certificate on a Mobile Device or Remote Browser\*).

5. \*\*Enable\*\* SSL proxying for the domains `mqtt-mini.facebook.com:\* and edge-mqtt.facebook.com:\*` (\*Proxy\* > \*SSL Proxying Settings...\*).

6. (optional) forward HTTP Packets to Burp (Charles only decrypts TLS 1.2):

\* Go to \*Proxy\* > \*External proxy settings...\*, enable it and forward HTTP and HTTPS to `127.0.0.1:{BURP\_PORT}`.

\* See your HTTP(S) requests in Burp.