SPEKTRUM

What are you looking for?

## DeepFake API

- Introduction
- Getting Started

## Test Definitions

- Paper
- DeepFake
- Error
- ID

## POST Data

- Upload Image
- Secret Key
- Sample Call
- Sample Response
- Test Parameters

## Post Examples

- HTTP
- JavaScript - jQuery
- JavaScript - Fetch
- Java - OkHttp
- NodeJs - Axios
- NodeJs - Native
- NodeJs - Request
- NodeJs - Unirest
- PHP - cURL
- PHP - HTTP_Request2

PHP - pecl_http

Python - http.client

Ruby - Net::HTTP

## Mobile SDK

How it works

Flutter

React Native


# Spektrum DeepFake API

**Introduction**

Our DeepFake API features advanced machine learning algorithms that can dig up forged identity cards and common deep fakes. This gives your business a cutting edge solution that blends well with most fraud prevention approaches when it comes to verifying consumer data to detect possible vulnerabilities and expose deep fakes. Its two-step risk identity and assessment model significantly reduces and/or eliminates the entire manual review process.



**Getting Started**

Our API will detect deepfake drivers licenses, risky identification cards, passports, visas, green cards and credit cards, and even currency.

**The App Test Definitions**

**Paper Test**

The paper test adds an extra layer of security when assessing consumer's verification data. This test's role is to ensure that customers don't exploit forged identification cards that look just like physical identifications used in passing onboarding and verification requests. The A.I used in this technology validates the identification cards and screens the documents to check if they are colored copies printed on paper.

**DeepFake Test**

Detects fraudulent documents, Most common deep fakes are sourced from the same or similar templates that are passed around and gets sold and resold online. The DeepFake test was designed to be an intelligent and spot-on way of detecting fraudulent documents..

**Error Test**

Error Test checks submitted data against the approved industry protocols. The screening here helps businesses to check if the submitted data are invalid or don't fulfill the required minimums to provide accurate results.

**ID Test**

This test checks consumer data against specified thresholds. It helps determine that the submitted data is within the required limit to be approved as a valid identification card in all the 50 states.

**Upload Image**

This api end point takes an image. Its a POST end point. It takes image data uploads it to the server and performs the ID, DeepFake, Error and Paper test and generates integrated output.

**Secret Key**

In order to communicate with API you must provide secret 'key' when calling API.

key=A0Zr98j/3yX%20R~XHH!jmN]LWX/,?RT'

**Sample Call**

r = requests.post( 'http://127.0.0.1:80/uploader' ,  files ={ 'file' :  open (image_path,  'rb' )})

curl --location --request POST 'http://127.0.0.1:80/uploader?key=A0Zr98j/3yX%20R~XHH!jmN]LWX/,?RT' \ --form 'file=@/IMG122.jpg'

**Sample Response**

{"Data":{"DEEP FAKE":"HIGH RISK","ERROR":"PASS","ID":"FAIL","PAPER":"PASS"},"Message":"Success","status":true}


```
{
    "Data": {
        "DEEP FAKE": "HIGH RISK",
        "ERROR": "PASS",
        "ID": "FAIL",
        "PAPER": "PASS"
    },
    "Message": "Success",
    "status": true
}
```


**Test Parameters**

**DeepFake HighRisk**

The technology will mark submitted data as DeepFake HighRisk if the data has higher odds of being fraudulent. This would then make it easy for businesses to take the necessary containment measures like disposing of them or blocking associated IPs.

**DeepFake LowRisk**

Low Risk is acceptable, but businesses could still perform further scrutiny on the data.

**DeepFake MediumRisk**

Medium Risk would be considered acceptable, but businesses could still perform further scrutiny on the data. The technology will mark data that have low to medium range risks of being fraudulent as DeepFake MediumRisk.

**I.D. Pass**

Usually, the older generation identification cards will output "PASS". I.D. Pass indicates that the data submitted has met the minimum to medium requirements for approval.

**I.D. Fail**

ID Fail report by Spektrum A.I indicates that provided data was either of poor quality or wasn't an identity card. The technology may also output I.D. Fail if the submitted data doesn't meet the required minimums for approval.

**I.D. Verified**

Only the new generation identification cards will output "VERIFIED". This output is only possible if the submitted data has met the highest available verification score.

**Paper Test Pass**

The submitted data is genuine. It is neither a photoshopped document that has been printed on paper nor a modified template.

**Error Test Pass**

The submitted data is tolerable for a test.

**Error Test Fail**

The submitted image could be within very low-quality ranges. The output may also mean that the submitted image was a piece of document or something that couldn't be detected to be a government-issued identity card.

**HTTP**

```
POST /uploader?key=A0Zr98j/3yX R~XHH!jmN]LWX/,?RT HTTP/1.1
Host: 127.0.0.1:5001
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="file"; filename="WA_EDL_Front.jpg"
Content-Type: image/jpeg

(data)
----WebKitFormBoundary7MA4YWxkTrZu0gW

```

**JavaScript - jQuery**

```
var form = new FormData();
form.append("file", fileInput.files[0], "WA_EDL_Front.jpg");

var settings = {
  "url": "http://127.0.0.1:5001/uploader?key=A0Zr98j/3yX R~XHH!jmN]LWX/,?RT",
  "method": "POST",
  "timeout": 0,
  "processData": false,
  "mimeType": "multipart/form-data",
  "contentType": false,
  "data": form
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

**JavaScript - Fetch**

```
var formdata = new FormData();
formdata.append("file", fileInput.files[0], "WA_EDL_Front.jpg");

var requestOptions = {
  method: 'POST',
  body: formdata,
  redirect: 'follow'
};

fetch("http://127.0.0.1:5001/uploader?key=A0Zr98j/3yX R~XHH!jmN]LWX/,?RT",
  requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

**Java - OkHttp**

```java
OkHttpClient client = new OkHttpClient().newBuilder()
  .build();
MediaType mediaType = MediaType.parse("text/plain");
RequestBody body = new MultipartBody.Builder().setType(MultipartBody.FORM)
  .addFormDataPart("file","WA_EDL_Front.jpg",
   RequestBody.create(MediaType.parse("application/octet-stream"),
   new File("/media/hiveforensics/Backup/HiveAI/cyber-security/spektrum/images/risky/
     WA_EDL_Front.jpg")))
  .build();
Request request = new Request.Builder()
  .url("http://127.0.0.1:5001/uploader?key=A0Zr98j/3yX R~XHH!jmN]LWX/,?RT")
  .method("POST", body)
  .build();
Response response = client.newCall(request).execute();
```

**NodeJs - Axios**

```
var axios = require('axios');
var FormData = require('form-data');
var fs = require('fs');
var data = new FormData();
data.append('file', fs.createReadStream('/media/hiveforensics/Backup/HiveAI/
  cyber-security/spektrum/images/risky/WA_EDL_Front.jpg'));

var config = {
  method: 'post',
  url: 'http://127.0.0.1:5001/uploader?key=A0Zr98j/3yX R~XHH!jmN]LWX/,?RT',
  headers: {
    ...data.getHeaders()
  },
  data : data
};

axios(config)
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});

```

**NodeJs - Native**

```nodejs
var http = require('follow-redirects').http;
var fs = require('fs');

var options = {
  'method': 'POST',
  'hostname': '127.0.0.1',
  'port': 5001,
  'path': '/uploader?key=A0Zr98j%2F3yX%20R~XHH!jmN%5DLWX%2F%2C%3FRT',
  'headers': {
  },
  'maxRedirects': 20
};

var req = http.request(options, function (res) {
  var chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function (chunk) {
    var body = Buffer.concat(chunks);
    console.log(body.toString());
  });

  res.on("error", function (error) {
    console.error(error);
  });
});

var postData = "------WebKitFormBoundary7MA4YWxkTrZu0gW\r\nContent-Disposition: form-data; name=\"file\"; filename=\"WA_EDL_Front.jpg\"\r\nContent-Type: \"{Insert_File_Content_Type}\"\r\n\r\n" + fs.readFileSync('/media/hiveforensics/Backup/HiveAI/cyber-security/spektrum/images/risky/WA_EDL_Front.jpg') + "\r\n------WebKitFormBoundary7MA4YWxkTrZu0gW--";

req.setHeader('content-type', 'multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW');

req.write(postData);

req.end();
```

**NodeJs - Request**

```nodejs
var request = require('request');
var fs = require('fs');
var options = {
  'method': 'POST',
  'url': 'http://127.0.0.1:5001/uploader?key=A0Zr98j/3yX R~XHH!jmN]LWX/,?RT',
  'headers': {
  },
  formData: {
    'file': {
      'value': fs.createReadStream('/media/hiveforensics/Backup/HiveAI/cyber-security/spektrum/images/risky/WA_EDL_Front.jpg'),
      'options': {
        'filename': 'WA_EDL_Front.jpg',
        'contentType': null
      }
    }
  }
};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});

```

**NodeJs - Unirest**

```
var unirest = require('unirest');
var req = unirest('POST', 'http://127.0.0.1:5001/uploader?key=A0Zr98j/3yX R~XHH!jmN]
  LWX/,?RT')
  .attach('file', '/media/hiveforensics/Backup/HiveAI/cyber-security/spektrum/images/
    risky/WA_EDL_Front.jpg')
  .end(function (res) {
    if (res.error) throw new Error(res.error);
    console.log(res.raw_body);
  });

```

**PHP - cURL**

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "http://127.0.0.1:5001/uploader?key=A0Zr98j/3yX%20R~XHH!jmN%5DLWX/,?RT",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_POSTFIELDS => array('file'=> new CURLFILE('/media/hiveforensics/Backup/HiveAI/cyber-security/spektrum/images/risky/WA_EDL_Front.jpg')),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;

```

**PHP - HTTP_Request2**

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('http://127.0.0.1:5001/uploader?key=A0Zr98j/3yX R~XHH!jmN]LWX/,?RT');
$request->setMethod(HTTP_Request2::METHOD_POST);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->addUpload('file', '/media/hiveforensics/Backup/HiveAI/cyber-security/
  spektrum/images/risky/WA_EDL_Front.jpg', 'WA_EDL_Front.jpg', '<Content-Type Header>')
  ;
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

## SDK

**How our verification works**

Content

**Flutter**

Content

**React Native**

Content

For PDF Documentation, click here

The End