- API SUMMARY
- API METHODS - DEFAULT
  - fraudScannerPost

Vaarhaft API

API and SDK Documentation

Version: 5.0.0

API for image verification

Use the following base link for the API:
`https://api.vaarhaft.com`

Maximum upload size is 10 MB.

fraudScannerPost

Send a ZIP file containing images for processing

POST

`/fraudScanner`

Usage and SDK Samples

Python

```python
import requests
import json
import os
import re

def verifyImages(url, file_path, api_key, caseNumber, issueDate):

    # Create a folder named after the caseNumber
    folder_path = caseNumber
    if not os.path.exists(folder_path):
        os.makedirs(folder_path)

    with open(file_path, 'rb') as file:
        files = {'file': (file_path.split('/')[-1], file, 'application/zip')}
        headers = {'x-api-key': api_key, 'caseNumber': caseNumber, "issueDate": issueDate}

        response = requests.post(url, files=files, headers=headers)
        try:
            response.raise_for_status()
        except requests.exceptions.HTTPError as e:
            print("HTTP error occurred:", e)
            return


    if response.status_code == 200:
        content_type = response.headers.get('Content-Type')
        if content_type:
            if 'application/json' in content_type:
                # Save the JSON response in the new folder
                response_json = response.json()
                json_file_path = os.path.join(folder_path, "response.json")
                with open(json_file_path, "w") as json_file:
                    json.dump(response_json, json_file, indent=4)

            elif 'multipart/mixed' in content_type:
                boundary = content_type.split("boundary=")[1]
                parts = response.content.split(f'--{boundary}'.encode())
                for part in parts:
                    # Extract the filename from the Content-Disposition header
                    content_disposition = re.search(r'filename="(.+?)"', part.decode(errors='ignore'))

                    if b'Content-Type: application/json' in part:
                        # Save JSON part in the new folder
                        json_part = part.split(b'\r\n\r\n')[1].strip(b'\r\n')
                        response_json = json.loads(json_part)
                        json_file_path = os.path.join(folder_path, "response.json")
                        with open(json_file_path, "w") as json_file:
                            json.dump(response_json, json_file, indent=4)

                    elif b'Content-Type: application/zip' in part and content_disposition:
                        # Extract the actual filename from Content-Disposition header
                        filename = content_disposition.group(1)
                        zip_part = part.split(b'\r\n\r\n')[1].strip(b'\r\n')
                        zip_file_path = os.path.join(folder_path, filename)

                        # Save the ZIP file with the correct filename
                        with open(zip_file_path, 'wb') as zip_file:
                            zip_file.write(zip_part)
        else:
            print("No Content-Type header in the response")
    else:
        print("Failed to upload file.")
        print("Status Code:", response.status_code)
        try:
            print("Server response:", response.json())
        except json.JSONDecodeError:
            print("Server response is not JSON")



# Example usage
server_url = 'https://api.vaarhaft.com/fraudScanner'
zip_file_path = ''
api_key = ''
caseNumber = ''
issueDate = ''
send_zip_file_to_server(server_url, zip_file_path, api_key, caseNumber, issueDate)
```

Parameters

Header parameters

Name	Description
x-api-key*	String *API key for authentication* Required

Body parameters

Name	Description

body *	{
	Required: caseNumber,x-api-key,zipFile
	zipFile: string (binary) *ZIP file containing images*
	caseNumber: string *Case number for tracking*
	issueDate: string (date-time) *Date of the case*
	}

Responses

Status: 200 - ZIP file successfully processed

Schema

```
{
  caseNumber: [
    {
      imageId1: {
        imageQuality: {
          result: boolean
          error: boolean
          enabled: boolean
          description: This is an upstream check that checks whether the image meets certain minimum requirements. (resolution, sharpness ...)
        }
        generatedDetection: {
          confidence: number
          predictedClassName: string
          error: boolean
          enabled: boolean
          description: Images that have been fully generated by an artificial intelligence are recognized with this feature. The predictedClass can be either 'gen' (generated) or 'real' (Real). The confidence indicates how certain our model is.
        }
        tamperedDetection: {
          confidence: number
          predictedClassName: string
          error: boolean
          enabled: boolean
          description: Images that have been subsequently processed by an AI or other software are recognized with this feature. The predictedClass can assume either 'tp' (tampered) or 'real' (Real). The confidence indicates how certain our model is.
        }
        doubletCheck: {
          result: boolean
          intern: boolean
          similarityPercentage: number
          error: boolean
          enabled: boolean
          description: With the similarity comparison, the Fraud Scanner checks whether the analyzed image has already been submitted to you or another insurance company in our database.
        }
        reverseSearch: {
          result: boolean
          matches: [
            [
              string (uri)
            ]
          ]
          error: boolean
          enabled: boolean
          description: This feature checks whether the submitted image originates from the Internet and has already been uploaded there once. If the image was found on the Internet, we also return the links to the websites found.
        }
        metadata: {
          exifData: {
          }
          FileType: string
          Mode: string
          Width: integer
          Height: integer
          error: boolean
          enabled: boolean
          description: The metadata for each image is extracted and displayed in an organized manner so that you can quickly obtain all important additional information about the image. This is a rudimentary check that checks existing metadata.
        }
      }
    }
  ]
}
```

application/zip :

> Filename : heatmaps.zip
>
> Description : This ZIP file is generated only if tampering is detected in one or more images. It contains heatmap images that highlight areas of detected tampering. Each heatmap file is named to match its corresponding image, ensuring easy identification (e.g., 'image1_heatmap.png' for 'image1.png').

application/zip :

> Filename : thumbnail.zip
>
> Description : This ZIP file contains images extracted from PDFs that were submitted with embedded images. Each image corresponds to one extracted from the original PDF.

Example Result


```json
{
  "statusCode":  200,
  "body":  {
    "TestCase":  {
      "stable3_0111c29e-cd60-42e4-9c3a-31093a210603.png":  {
        "imageQuality":  {
          "result":  true,
          "error":  false
        },
        "doubletCheck":  {
          "result":  true,
          "caseNumber":  "erv",
          "intern":  false,
          "error":  false,
          "enabled":  true
        },
        "reverseSearch":  {
          "result":  false,
          "matches":  [],
          "error":  false,
          "enabled":  false
        },
        "metadata":  {
          "analysed":  {
            "creationDate":  {
              "cDate":  "-",
              "cTime":  "-",
              "isSus":  false,
              "diffInDays":  0,
              "isTodayUsed":  false,
              "refDate":  "-"
            },
            "imgRanking":  null,
            "fieldsMarkedSus":  {}
          },
          "GPSInfo":  {},
          "raw":  {
            "Rating":  {},
            "Dates":  {},
            "GPS":  {},
            "Other":  [
              {},
              {
                "Image width":  "1024 pixels",
                "Image height":  "1024 pixels",
                "Bits/pixel":  "24",
                "Pixel format":  "RGB",
                "Compression rate":  "1.3x",
                "Compression":  "deflate",
                "MIME type":  "image/png",
                "Endianness":  "Big endian"
              }
            ]
          },
          "error":  false,
          "enabled":  true
        },
        "generatedDetection":  {
          "predictedClassName":  "gen",
          "confidence":  0.9999997615814209,
          "error":  false,
          "enabled":  true
        },
        "tamperedDetection":  {
          "predictedClassName":  "real",
          "confidence":  0.9994547347868388,
          "error":  false,
          "enabled":  true
        }
      }
    }
  },
  "headers":  {
    "Access-Control-Allow-Headers":  "Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token",
    "Access-Control-Allow-Methods":  "OPTIONS,POST",
    "Access-Control-Allow-Origin":  "*"
  },
  "sessionId":  "065a9dbc-aeeb-4a0f-890a-574fe03b5f08"
}
```