ZignSec

DOCUMENT FORGERY DETECTION IMPLEMENTATION GUIDELINES

ZignSec AB (publ)

Version 1.0 | Updated 15/02/2024

Fraud Detection Implementation Guidelines

Code:	
Version:	1.0
Date of version:	2024-Feb-15
Created by:	Martin Heikkilä, Head of Support
Approved by:	Jason Coombes, Head of Risk and Compliance
Confidentiality level:	PUBLIC

Table of Contents

Document Forgery Detection

The Forgery Detection system by ZignSec is designed to protect business operations and automated workflows from frauds related to forgery. It employs AI to meticulously analyze financial documents and others like bank statements, invoices, payroll slips, and KYC documents submitted by customers, looking for signs of tampering or inconsistencies. The system can detect modifications, file corruption, and other anomalies, thereby raising alerts for any suspicious activity.

File formats supported: PDF, JPEG, PNG, TIFF and HEIC

As a result of the document analysis, we provide a final verdict (Trusted, Normal, Warning and High Risk), complete set of evidence to support the verdict, a basic metadata overview and parsing outcome, if applicable.

Use cases with limited functionality:

- Images that do not contain characters (e.g. a photo of a house)
- Document was modified/resized in the internal Data Management System prior to submission for analysis (such operations tend to wipe the modification trail and make it hard or impossible to detect)
- Physical modification before the document was scanned or photographed
- Documents with abnormally reduced quality

The rate tested by provider:

<= 0.8% of false positive rates

The statistics from provider: The precision of detecting digital fraud

- on PDFs is very high 95%+,
- for images it culminates between 60-80%

API Overview

This API provides comprehensive services for checking if a document has been tampered with.

Workflow Overview

The service allows for the analysis of uploaded documents to detect manipulations through a structured workflow:

1. **Create a New Scanning Session**: Initiate a session to which media can be uploaded for analysis.
2. **Upload Media**: Add media files to the created session. Multiple files can be uploaded, supported formats include PDF, JPEG, PNG, TIFF, and HEIC.
3. **Manage Media Metadata**: Optionally, you can add tags to media for categorization, update or replace media information to keep track of changes or specific attributes.
4. **Start Analysis**: Once media is uploaded and optionally tagged or described, initiate the analysis to detect manipulations.
5. **Retrieve Results**: After analysis, the state of the session can be queried to obtain results, either through direct requests or via webhook notifications.

Base URLs

- **Test Environment**: `**https://test-gateway.zignsec.com/api/v5/sessions/scanning**`
- **Production Environment:** ` **https://gateway.zignsec.com/api/v5/sessions/scanning**`

API Documentation

- **Swagger UI**: Swagger UI (zignsec.com)

Detailed Endpoint Descriptions

Scanning Session Management

- **POST /scanning**
  - Creates a new scanning session. This is the first step in the workflow where a session ID is generated for subsequent operations.
- **GET /scanning/{sessionId}**
  - Retrieves the current state of a scanning session, including any results or status updates.

Media Management

- **POST /scanning/{sessionId}/media**
  - Uploads a media file to the specified session. This step is essential for adding the documents you want analyzed.
- **POST /scanning/{sessionId}/media/{mediaId}/tags**

- Adds tags to a specific media item. Tags can help categorize or identify media within the session.

- **DELETE /scanning/{sessionId}/media/{mediaId}/tags**
  - Removes tags from a media item, allowing for the correction or update of categorization information.
- **PUT /scanning/{sessionId}/media/{mediaId}/info**
  - Replaces all information about a media item. This can be used to update metadata or other relevant descriptions.
- **PATCH /scanning/{sessionId}/media/{mediaId}/info**
  - Updates specific pieces of information about a media item without replacing everything. This is useful for making partial adjustments.

Analysis Management

- **POST /scanning/{sessionId}/analyses**
  - Starts the analysis process for the uploaded media in the session. This step is crucial for detecting manipulations in the documents.

Requests and responses and parameter tables

Sample of Request Create a Session

```
curl --location 'https://test-gateway.zignsec.com/api/v5/sessions/scanning' \
--header 'Zs-Merchant-Id: Your-Access-key ' \
--header 'Content-Type: application/json' \
--header 'Authorization: Your-Access-key' \
--data '{
  "locale": "en",
  "relay_state": "   my-unique-customer-id",
  "gdpr_user_id": "my-user-id",
  "webhook": “https://my_webhook_url.com”,
  "metadata": {
    "flow": {
      "id": "fraud",
      "input": {
        "mappingProfile": "confident"
      }
    }
  }
}'
```

Request parameters

Parameter	Type	Description
locale	string	Preferred Language to Use. Example: En
metadata	object	Scanning - create session request data. Contains information about the scanning flow
flow	object	Contains details for initiating a scanning flow session.
id	string	Flow ID. Supported values: "fraud". Must be at least 1 character in length.
inputRef	string (nullable)	Input object reference. The input object will be obtained from settings by reference.
input	object	Contains details about the scanning flow input.
mappingProfile	string	Analysis mapping profile. Must be at least 1 character in length.
retryPolicyRef	string (nullable)	Retry policy reference. The retry policy will be obtained from settings by reference.
retryPolicy	ScanningFlowRetryPolicyDto	Specifies the retry policy for the scanning session.
retryConditions	array of string (nullable)	Retry conditions. Supported values: "retryDeclined", "retryTimeout", "retryFailed".
maxRetries	integer ($int32)	Number of retries allowed. Maximum: 10, Minimum: 1.

gdpr_user_id	string	To be used to track single person requests, also to be used to remove all data related to the person when requested. Example: my-unique-gdpr-customer-id
relay_state	string	Optional Custom Parameter to be included in webhook calls. Example: my-unique-customer-id
webhook	string	Webhook URL where your backend will receive session events. Example: https://my_webhook_url.com

Sample Response


```
{
  "id": "65b645c9-3346-49da-afc8-0fe6ad211872",
  "status": "Created",
  "errors": [],
  "result": {
    "media": [],
    "analysisResults": []
  }
}
```


Response parameters

Parameter	Type	Description
id	(string ($uuid))	Id
status	(string)	- Created: Session is created. - GeneratedLink: Web flow link is generated. - ReadyToStart: Documents are uploaded, and the session is ready to start analysis. - Pending: Session is in pross. - Accepted: Session is accepted. - Declined: Session is declined. - Failed: Session has failed. - Timeout: Session has timed out. - Cancelled: Session is cancelled. - OperatorRequired: Indicates a special state where manual review by a human operator is advisable.
errors	(array)	List of errors
result	(object)	The result of the session

media	(array)	Media files received from the client and analysis output media

Sample of Request Upload a media file to session.

```
curl --location 'https://test-gateway.zignsec.com/api/v5/sessions/scanning/65b645c9-3346-49da-afc8-0fe6ad211872/media' \
--header 'Authorization: Your-Access-key ' \
--form 'File=@"front.png"' \
--form 'Name="my-file.png"' \
--form 'MimeType="image/png"' \
--form 'Reference="file-12345"' \
--form 'MetaData="{\"foo\": \"bar\"}"' \
--form 'Tags="flow-1"'
```

Request parameters

Parameter	Type	Description
File	$binary	Media file
Name	string	(optional) File name. If not specified, name of uploaded file will be used
mimeType	string	(optional) MIME type of the file. If not specified, the MIME type will be determined automatically based on the file extension
Reference	string	(optional) User-defined ID of the media file. Typically refers to internal ID of the analyzed media file in user's system
MetaData	Object	Media metadata
Tags	string	Media tags, used to query media

Sample of Request add tags to a Session

```
curl --location --globoff 'https://test-gateway.zignsec.com/api/v5/sessions/scanning//media/{{mediaId}}/tags' \
--header 'Content-Type: application/json' \
--header 'Authorization: Your-Access-key ' \
--data '[
    "tag1",
    "tag2",
    "MEDIA_INPUT"
]'
```

Request parameters

Parameter	Type	Description
Array of tags	string (any value)	Array of tags

Sample of Request Analyse a Session

```
curl --location 'https://test-gateway.zignsec.com/api/v5/sessions/scanning/65b645c9-3346-49da-afc8-0fe6ad211872/analyses' \
--header 'Content-Type: application/json' \
--header 'Authorization: Your-Access-key ' \
--data '{
  "analysisType": "Fraud",
  "folderQuery": {
    "queryType": "ById",
    "mediaIds": [
      "c46e294f-3957-4c40-9dc5-2cfc83f18cbe"
    ]
  }
}
```

Request parameters

Parameter	Type	Description
analysisType	String	Fraud (new values can be introduced.)
folderQuery	Object	queryType
queryType	String	- ById – query by sessionId - ByAllTags – query by tag - QueryFirstMatch – Find first match from the given queries
mediaIds	String	Array of strings <uuid>
Metadata	Object	

Note on folderQuery usage

Please note, the folderQuery object body may contain different fields depending on the queryType (we do not fully utilize openapi specification oneOf though because of bad support from the code generation tools, NSwag is recommended for .NET).

The supported query types for now (new types can be added in the future) :

- ById – uses QueryMediaByIds type with fields:
  - mediaIds – array of string ($uuid) – queries media by the given IDs
- ByAllTags – uses QueryMediaByAllTags type with fields:
  - tags – array of string – queries media by all tags (using logical AND)
- QueryFirstMatch – uses QueryFirstMatch type with fields:
  - queries – array of objects (FolderQuery) – queries the first match of running the given queries one by one

The openapi documentation illustrating the described behavior - https://test-gateway.zignsec.com/api/v5/openapi/scanning/doc/#tag/Scanning-API-Flow/operation/ScanningApiFlow_StartAnalysis;
And swaggerui (https://test-gateway.zignsec.com/api/v5/openapi/scanning/) has a good representation for all of the mentioned models

Sample of Request GET a Session

```
curl --location 'https://test-gateway.zignsec.com/api/v5/sessions/scanning/65b645c9-3346-49da-afc8-0fe6ad211872' \
--header 'Authorization: Your-Access-key ' \
--data
```

Sample Response


```
{
  "id": "65b645c9-3346-49da-afc8-0fe6ad211872",
  "status": "Declined",
  "errors": [],
  "result": {
    "media": [
      {
        "id": "c46e294f-3957-4c40-9dc5-2cfc83f18cbe",
        "name": "my-file.png",
        "url": "https://test-gateway.zignsec.com/folders/7000167a-a5ce-4677-aa06-be4833e2f816/media/c46e294f-3957-4c40-9dc5-2cfc83f18cbe",
        "mimeType": "image/png",
        "tags": [
          "flow-1",
          "MEDIA_INPUT",
          "MEDIA_RAW"
        ],
        "metadata": {
          "foo": "bar"
        },
        "reference": "file-12345"
      }
    ],
    "analysisResults": [
      {
        "analysisType": "FraudDp45",
        "fraudDp45": {
          "providerResults": [
```

```
{
  "mediaId": "c46e294f-3957-4c40-9dc5-2cfc83f18cbe",
  "providerTransactionId": "d17f9b7f-998b-407f-9c21-2f63177c0637",
  "status": "Success",
  "analysisTime": "2024-02-19T15:42:58.523784+00:00",
  "fileType": "Image",
  "mimeType": "image/jpeg",
  "deploymentVersion": "73b5ce98",
  "queryId": "75dc0584-462c-4f11-99fa-3547d7db2e20",
  "sha256": "d5f40cc616465edc14c6af2c9779f458c1ac8fe710781b1d06d5f15b57d67a2d",
  "score": "Warning",
  "sampleMetadata": {},
  "indicators": [
    {
      "indicatorId": "screenshot",
      "type": "Risk",
      "category": "content_hiding",
      "title": "Screenshot",
      "description": "This image contains screenshot which may be a policy violation.",
      "origin": "Fraud"
    },
    {
      "indicatorId": "low_resolution",
      "type": "Info",
      "category": "low_quality_image",
      "title": "Document has low resolution",
      "description": "Resolution of images stored in this document is too low for reliable analysis. Some detectors may be disabled.",
      "metadata": {
        "type": "DataOnly",
        "columnNames": [
          "character_height",
          "effective_resolution"
        ],
        "columnsDefinition": []
      },
      "origin": "Fraud"
    }
  ],
  "documentClass": {
    "id": "unknown",
    "type": "unknown",
```

```
                    "documentClassType": "DocumentClass"
                  }
                }
              ]
            },
            "id": "75dc0584-462c-4f11-99fa-3547d7db2e20",
            "errors": [],
            "inputMedia": [
              {
                "id": "c46e294f-3957-4c40-9dc5-2cfc83f18cbe",
                "name": "my-file.png",
                "url": "https://test-gateway.zignsec.com/folders/7000167a-a5ce-4677-aa06-
be4833e2f816/media/c46e294f-3957-4c40-9dc5-2cfc83f18cbe",
                "mimeType": "image/png",
                "tags": [
                  "flow-1",
                  "MEDIA_INPUT",
                  "MEDIA_RAW"
                ],
                "metadata": {
                  "foo": "bar"
                },
                "reference": "file-12345"
              }
            ],
            "outputMedia": [],
            "status": "OperatorRequired",
            "startedAt": "2024-02-19T15:42:56.1188277Z",
            "finishedAt": "2024-02-19T15:43:06.0250412Z"
          }
        ]
      }
    }
```


Response summary

The response body of the scanning session endpoint is structured into several parts:

1. **ID** and **Status**: It begins with an ID (unique identifier) and a status indicating the session's current phase (e.g., Created, Pending, Accepted).
2. **Errors**: A list of errors, if any, detailing the error code, description, and additional details.
3. **Result**: This section encompasses media files received, analysis results, and specific outcomes for different analyses (e.g., FraudDp45, DocumentDp50). It includes:

a. **Media**: Details about each media file, including ID, name, URL, mimeType, tags, metadata, and reference.
b. **AnalysisResults**: Summary of each analysis performed, including type, status, errors, input and output media, and detailed results of specific analyses like fraud or document analysis.

Response parameters

Parameter	Type	Description
id	(string ($uuid))	Id
status	(string)	Overall status of session • Created - Session is created, • GeneratedLink - Web flow link generated, • ReadyToStart - Documents uploaded and session is ready to start, • Pending - Session is in progress, • Accepted - Session accepted, • Declined - Session declined, • Failed - Session failed, • Timeout - Session timed out, • Cancelled - Session cancelled, • OperatorRequired - Special state noting that it's better to review results by a human operator
errors	(array)	List of errors
result	(object)	The result of the session
media	(array)	Media files received from the client and analysis output media
id	(string ($uuid))	ID of the uploaded media
name	(string)	Name of the uploaded media
url	(string)	URL of the uploaded media
mimeType	(string)	mimeType of the uploaded media
tags	(array)	tags of the uploaded media
metadata	(object)	metadata of the uploaded media
reference	(string)	Reference of the uploaded media
analysisResults	(array)	Universal result - contains all the possible results inside

analysisType	(string)	Analysis type. Currently supported: FraudDp45 | DocumentDp50 **Important Note:** New analyses can be added without new version release, please implement your integration accordingly.
id	(string ($uuid))	Analysis id
errors	(array)	array designed to report issues encountered during a session, containing **SessionError** objects. Each **SessionError** details an error with a unique code, a human-readable description, and additional context or information.
inputMedia	(array)	Media sent to the analysis
outputMedia	(array)	Media generated by the analysis
status	(string)	Status of Analysis • NotRequested - Analysis not requested, • Requested - Analysis requested, • Pending - Analysis is in progress, • Accepted - Analysis accepted, • Declined - Analysis declined, • Failed - Analysis failed, • Cancelled - Analysis cancelled, • OperatorRequired - Operator required, • Timeout - Timeout
startedAt	(string)	Analysis start date
finishedAt	(string)	Analysis finish date
fraudDp45	(object)	result from dataprovider - DP45 (Data provider 45)
providerResults	(array)	Fraud check results for each media file
mediaId	(string($uuid))	ID of the media file that was analyzed
status	(string)	Fraud analysis status • Success • Failed • InvalidInput • Skipped

		Note, new values can be introduced
errorMessage	(string)	Error message
analysisTime	(string($date-time))	Timestamp of analysis
fileType	(string)	• Unsupported • Pdf • Image Note: new values can be introduced.
mimeType	(string)	Mime type of the input file, e.g. application/pdf, image/jpeg, etc.
deploymentVersion	(string)	Version of quality engine used for analysis
queryId	(string)	Customer's own submission ID. This field may contain customer's own internal ID of analyzed file (e.g. ID in database).
sha256	(string)	Sha256 of the input file
score	(string)	• Normal • Trusted • Warning • HighRisk Note: new values can be introduced.
sampleMetadata	(object)	Metadata about the document, including information about its production and content
producer	(string)	The producer of the document/file
creator	(string)	Tool used to create the document/file
creationDate	(string($date-time))	Date when document/file was originally created
modDate	(string($date-time))	Date when document was modified
author	(string)	The author of the document/file
title	(string)	The title of the document/file
keywords	(string)	Keywords of the document/file

subject	(string)	Subject of document/file
indicators	(array)	Indicators contains detailed information about the specific results of our analysis. This field contains detailed observations that the system was able to make about the file. Some indicators may be positive and may increase the Trust score, while the others would be negative and would increase the Risk. For complete list see reference in this document.
indicatorId	(string)	Unique identifier for the indicator. Example: “has_suspicious_adobe_distiller_metadata”, “is_pdf_unmodified_per_metadata” etc.
type	(string)	Type of a quality indicator. • Risk • Trust • Info Note: new values can be introduced.
category	(string)	Category of indicator Example: “modifications”, “text_hiding”, “content_hiding” etc.)
title	(string)	Name of indicator Example: “Saved in Adobe Distiller”, “No modification in document metadata” etc.
description	(string)	Detailed description of the indicator.
metadata	(object)	Additional metadata associated with the indicator. Content of metadata evolves and can change without prior notice. It is intended mainly for visualization in UI and tracking purposes.
indicatorAttributes	(object)	Attributes for indicators, including an optional identifier, primary logo details (issuing authority's ID, name, type, and country using ISO-3166-1 alpha-3 codes), detected logos with similar attributes, and identifiers related to potential fraud clusters.

		See swagger for complete list.
origin		• Fraud • Quality NOTE: new values can be introduced.
documentClass	(object)	Document class that describes type of file submitted for analysis, e.g. invoice, account statement, payslip, etc.
id	(string)	Class id Example: “id_document_driving_license_swe”
type	(string)	A unique identifier for the document class Example: “invoice_bouygues_fra”, “receipt_github”, “payslip_sdworx”, “id_document” etc.
documentClassType	(string)	Class type of document: Example: “DocumentClass” etc
detailedType	(string)	Type of the utility bill. **Note** that the list is not fixed, and additional types may be added without further notice.
issuingCountry	(string)	Country that issued this ID document, e.g. USA, DEU, FRA, CZE, SVK, etc. The country code is encoded with ISO-31661 alpha-3 code, see https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3#Officially_assigned_code_elements.
documentDp50	(object)	Document analysis result - DP50 - just an example how other analyses will look like

Sample Error

```
{
  "detail": "Malformed payload",
  "instance": "api/v5/sessions/scanning/",
  "status": 400,
  "title": "Bad Request",
  "type": "about:blank",
  "violations": [
```

```
    "metadata/ui_data :Invalid schema.type. Got: nil"
  ]
}
```


Errors

Parameter	Type	Description
detail	string	Error detail. Might contain technical information.
instance	string($uri)	A URI reference that identifies the problem type
status	string($uri)	The HTTP status code generated by origin server for this occurrence of the problem.
title	string	Error description. Human readable text.
type	string($uri)	A URI reference that identifies the problem type

Reference

List of Indicators – Complete list of indicators

Change History

Date of Change	Changed By	Summary of Change
March 2024	Martin Heikkilä	First version