Documentación de la API de Financial Intelligence v0.9Jul24

Endpoint: `/cajaficheros/ask`

Descripción

Este endpoint acepta un archivo y un tipo de archivo del cliente y devuelve un UUID para consultar el estado y los resultados posteriormente a través del endpoint `/cajaficheros/tasks/{id}`.

URL de Base

https

```
https://api-if.smartescrow.eu/cajaficheros
```

Método HTTP

POST

Autenticación

Es necesario incluir una cabecera con la clave api-key y el valor de la clave proporcionado por Smart Escrow.

Parámetros de la solicitud

- Content-Type: `multipart/form-data`
- fileType: Tipo de archivo que se está subiendo (p.ej., "Factura")
- file: El archivo que se sube

Ejemplos de implementación

JavaScript (Node.js con la librería Axios)

```
const axios = require('axios');
const FormData = require('form-data');
const fs = require('fs');

let data = new FormData();
data.append('fileType', 'Factura');
data.append('file', fs.createReadStream('/path/to/your/file.txt'));

axios({
  method: 'post',
  url: 'https://api-if.smartescrow.eu/cajaficheros/ask',
  headers: {
    'Accept': 'application/json',
    'Content-Type': `multipart/form-data; boundary=${data._boundary}`,
    'api-key': 'your_api_key_here'
  },
  data: data,
})
.then(function (response) {
  console.log(JSON.stringify(response.data));
})
.catch(function (error) {
  console.log(error);
});
```

PHP (con la librería Guzzle)

```php
<?php
require 'vendor/autoload.php';

use GuzzleHttp\Client;

$client = new Client();
$response = $client->request('POST', 'https://api-
if.smartescrow.eu/cajaficheros/ask', [
    'headers' => [
        'Accept' => 'application/json',
        'api-key' => 'your_api_key_here',
    ],
    'multipart' => [
        [
            'name'     => 'fileType',
            'contents' => 'Factura'
        ],
        [
            'name'     => 'file',
            'contents' => fopen('/path/to/your/file.txt', 'r')
        ]
    ]
]);

echo $response->getBody();
?>
```

Curl (linux)

```
curl --location 'https://cajaficheros.smartescrow.eu/api/ask' \
--header 'Accept: application/json' \
--header 'Content-Type: multipart/form-data' \
--header 'api-key: your_api_key_here' \
--form 'fileType="Factura"' \
--form 'file=@"/path/to/your/file.txt"'
```

IMPORTANTE:

+ Para ver los resultados de su archivo, una vez concluida la petición **POST**, podra consultar regularmente el estado de su tarea con un **GET** a:

`https://api-if.smartescrow.eu/cajaficheros/tasks/{id}`

+ Para ver los **fileType** disponibles para su uso, solo haz un **GET** a:

`https://api-if.smartescrow.eu/cajaficheros/files`

+ Parámetros adicionales: Así como puedes pasar fileType en cada ejemplo, puedes agregar también los siguientes parámetros opcionales:

- question: En esta variable puedes colocar “opcionalmente” alguna “**PREGUNTA**“ o “**PREMISA**” que desees hacer sobre el fichero.
  El formato que recibe la API es en formato Array de Objetos:

  Ej del payload de carga:

  file: Factura

  question: [{"key":"premisa","value":"123"},{"key":"pregunta","value":"la factura es valida?"},{"key":"pregunta","value":"cual es el importe total?"}]

  fileType: Factura

  La key puede ser:

  + premisa: Si es premisa, es una condición a aplicar a la respuesta
  + pregunta: Si se coloca este tipo, se agregara a la respuesta del JSON final una variable con la respuesta a dicha pregunta en el orden que fue hechas.
  Ej: { pregunta1: true, pregunta2: $210,75 }

  opcionalmente, también, si la **key** es distinta a “premisa” y “pregunta” también será considerada como una variable a agregar al JSON Final, pero no es recomendable, para no sobreescribir variables existentes.

- askIndex: Es una variable opcional por si quieres subir varias veces el mismo fichero, solo aumentas el número de askIndex para que te lo tome como una petición diferente.
- websocketId: Puedes definir tu propio ID con el cual recibir tus resultados. En caso de no proveer esta variable, se te asignará un ID automáticamente.

+ Por favor, asegúrate de reemplazar '/path/to/your/file.txt' con la ruta al archivo que deseas subir en tu sistema local. Asegúrate de tener instaladas y configuradas todas las dependencias necesarias para que los ejemplos de código funcionen correctamente.

Lista de “Archivos” [20-Jun-23]

+ Por favor, tenga en cuenta que este es el listado de ficheros públicos actuales a la fecha 20-Jun-23. También existen ficheros de tipo privado que no fueron listados aquí.

Recuerde, si desea ver el listado actualizado puede verlos haciendo **GET** a:
`https://api-if.smartescrow.eu/cajaficheros/files`

Listado al 12-Jul-24:

Albaran
Balance
Carta de Credito
Certificado de Origen
Certificados de Deuda
CIRBE
Conocimiento Aereo
Contratos
CVs
DNIs y NIEs
Documento NIF
DUA de Exportacion
DUA de Importacion
Factura
Factura Electronica
Justificante Bancario
Nomina
Pagares
Pasaporte
Pedidos
Referencia Catastral
Situacion Censal
Tickets
Transito
Vida Laboral
Apuntes Contables
Bill of Landing
BORME
Carta de Porte (CMR)
Cuadernos Bancarios
Impuestos Hacienda
ITA Informe de Trabajadores
Listado de Deudas
Listado de Socios
Movimientos Bancarios
Packing List

Perdidas y Ganancias
Referencia Catastral
Contratos
Documentos y Certificados
Escritura Apoderamiento
Escrituras
Nota Simple Registro de la Propiedad
Nota Simple Registro Mercantil
Titularidad Real

Update resumen final [09-Sep-23]

Nos complace anunciar que nuestra API ha sido actualizada para proporcionar información adicional sobre los documentos analizados al solicitar el endpoint principal `https://api-if.smartescrow.eu/cajaficheros/tasks/{id}`.

A continuación, presentamos una descripción detallada del nuevo campo disponible:

Estructura de Datos

Cuando envía una solicitud a nuestra API y se procesa correctamente, recibirá una respuesta en formato JSON. En este JSON, dentro de outputs, siempre al final del mismo (siempre será el último elemento de outputs), encontrará el nuevo set de datos que proporciona un análisis detallado del documento que ha enviado. Este es un ejemplo de la estructura de datos que recibirá:

```
{
    ...
    "result": {
        "createdAt": "04/09/2023",
        "metadata": {
            "documentName": "factura.txt",
            "documentType": "Factura",
            "documentSubcategory": {
              "category": {
                  "categoryMatch": "factura”,
                  "confidence": 0.9
              },
              "subCategory": {
                  "categoryMatch": "recibida”,
                  "confidence": 0.9
              },
            },
            "documentExtension": "txt"
        },
```

```
        "extractedVariables": { ... },
        "verificators": {
            "verificador1": {
                  "description": "[\"éxito\"]",
                  "data": { ... }
              },
            "verificador2": {
                  "description": "[\"error\"]",
                  "data": { ... }
              },
         },
        "finalStatus": "FinalizadoOK"
    }
}
```


Detalles del Análisis del Documento

Cuando reciba el campo "result", podrá interpretar la información de la siguiente manera:

- **createdAt: Fecha en que se generó el análisis.**
- **metadata: Información metadatos del documento.**
- **extractedVariables: Variables y datos extraídos del documento.**
- **verificators: Verificaciones realizadas para autenticar y corroborar la información del documento.**
- **finalStatus: Estado final del análisis.**

Ejemplo:

Se ha generado este fichero el `04/09/2023` **como consecuencia del análisis del documento aportado, constituyendo un resumen de los datos y conclusiones más significativos del mismo, que, a continuación, presentamos:**

- **Nombre de Documento:** `factura.txt`
- **Tipo de Documento:** `Factura`
- **Subcategoría del Documento: (No especificada)**
- **Extensión del Documento:** `txt`
- **Estado de Análisis del Fichero:** `FinalizadoOK`

Datos Extraídos del Documento:

```
Nombre Variable            Valor

...
```

Además, se han realizado distintos procesos con la finalidad de verificar la autenticidad y cohesión de los datos contenidos en el documento aportado, que, a continuación, procedemos a detallar:

- **Document Verification: (Descripción y datos de la verificación)**
- **... (Otros verificadores según estén presentes)**

Consideraciones Finales

Por favor, tenga en cuenta que estos datos serán siempre los últimos dentro del campo output en el JSON que devuelve la API. Si tiene alguna pregunta o inquietud sobre cómo interpretar o utilizar esta nueva característica, no dude en ponerse en contacto con nuestro equipo de soporte.

Tenga en cuenta que los campos `documentSubcategory` o los contenidos en `verificators` están sujetos a la configuración del paquete para dicho documento, por lo cual, pueden o no estar presentes o incluso variar en el resultado final, mostrando asi algun valor u otro o nulo incluso, dependiendo de la configuración del administrador para dicho paquete.

Gracias por confiar en nuestros servicios. ¡Esperamos que encuentre útil esta nueva adición a nuestra API!

Update 26/01/2024

Feedback

Documentación para el Endpoint `/feedback/{status}`

Descripción

Este endpoint procesa feedbacks sobre operaciones realizadas, permitiendo reintentar procesamientos, marcar como buenos o malos resultados, o indicar fallos críticos.

URL

`https://api-if.smartescrow.eu/cajaficheros/feedback/{status}`

Método HTTP

`POST`

Parámetros URL

- `status`**: Estado del feedback. Valores permitidos:**
  - `retry`**: Reintenta el procesamiento.**
  - `good`**: Marca el procesamiento como exitoso y guarda los datos.**
  - `bad`**: Marca el procesamiento como no exitoso y guarda los datos para revisión.**
  - `critical`**: Marca el procesamiento como fallo crítico.**

Headers

- `api-key`**: Clave API requerida para autenticación.**

Cuerpo de la Petición (Body)

El cuerpo de la petición debe ser un objeto JSON con la siguiente estructura:

json

{

```
 "message": {
 "originalName": "string",
 "askIndex": "string",
 "websocketId": "string"
 },
 "comment": "string (opcional)"
}
```

Respuestas

- 200 OK: Feedback procesado con éxito.
- 400 Bad Request: Error en la petición (por ejemplo, falta de parámetros requeridos).
- 401 Unauthorized: API key incorrecta o ausente.

Ejemplos de Uso

Ejemplo con cURL:

bash

```
curl -X POST 'https://api-if.smartescrow.eu/cajaficheros/feedback/good' \
-H 'api-key: tu_api_key' \
-H 'Content-Type: application/json' \
-d '{
 "message": {
 "originalName": "nombre_original.mp4",
 "askIndex": "123abc",
```

```
 "websocketId": "456def"
 },
 "comment": "Tu comentario aquí"
}'
```

Ejemplo con Fetch (Node.js):

javascript

```
const url = 'https://api-if.smartescrow.eu/cajaficheros/feedback/good';
const apiKey = 'tu_api_key';
const data = {
 message: {
 originalName: "nombre_original.mp4",
 askIndex: "123abc",
 websocketId: "456def"
 },
 comment: "Tu comentario aquí"
};

fetch(url, {
 method: 'POST',
 headers: {
 'api-key': apiKey,
```

```
 'Content-Type': 'application/json'
 },
 body: JSON.stringify(data)
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

Ejemplo con Guzzle (PHP):

php

```
$client = new \GuzzleHttp\Client();
$response = $client->request('POST', 'https://api-
if.smartescrow.eu/cajaficheros/feedback/good', [
 'headers' => [
 'api-key' => 'tu_api_key',
 'Content-Type' => 'application/json'
 ],
 'json' => [
 'message' => [
 'originalName' => 'nombre_original.mp4',
 'askIndex' => '123abc',
 'websocketId' => '456def'
```

```
 ],
 'comment' => 'Tu comentario aquí'
 ]
]);

echo $response->getBody();
```

Recuerda reemplazar `'https://tuservidor.com/api/feedback/good'` **y** `'tu_api_key'` **con la URL real de tu servidor y tu clave API. Asegúrate de que la estructura del objeto** `message` **en el cuerpo de la petición coincida con la estructura esperada por tu endpoint.**

Documentación Adicional para la API de Financial Intelligence

Procesador "FinalSummary"

Al final de todas las tareas en la cola, se ejecuta un procesador común llamado "FinalSummary". Este procesador recopila y exporta una serie de variables e información importante sobre la tarea y el documento procesado.

Estructura de Datos de "FinalSummary"

El procesador "FinalSummary" genera un resumen detallado que incluye los siguientes campos:

- `createdAt`**: Fecha y hora de creación del resumen.**
- `metadata`**: Metadatos del documento, incluyendo:**
  - `documentName`**: Nombre del documento.**
  - `documentType`**: Tipo de documento.**
  - `documentSubcategory`**: Subcategoría del documento (si está disponible).**
  - `documentExtension`**: Extensión del archivo del documento.**
- `extractedVariables`**: Variables y datos extraídos del documento durante el procesamiento.**
- `verificators`**: Resultados de las verificaciones realizadas en el documento.**
- `queueStatus`**: Estado de la cola en la que se procesó el documento. Valores posibles:**
  - `Stand-By`
  - `Procesando`
  - `Error`
  - `FinalizadoOK`
  - `FinalizadoNecesitaRevision`
  - `FinalizadoOKVerificacionKO`
  - `FinalizadoOKVerificacionDoble`
  - `FinalizadoSinRevisión`
- `isTaskDone`**: Indica si la tarea ha terminado** `(true` **o** `false)`

Cuando el procesador "FinalSummary" marca uno de los siguientes estados, significa que la tarea de procesamiento ha concluido completamente:

- `FinalizadoOK`

- `FinalizadoNecesitaRevision`
- `FinalizadoOKVerificacionDoble`
- `FinalizadoSinRevisión`

Entidad "Output"

La entidad "Output" representa la salida de cada procesador en la cola relacionada con la entidad "Task". Contiene las siguientes variables:

- `id`**: Identificador único del output.**
- `task`**: Tarea asociada al output.**
- `msgClass`**: Clase del mensaje o nombre del procesador.**
- `processType`**: Tipo de proceso realizado.**
- `isTaskDone`**: Indica si la tarea ha finalizado (`true` o `false`).**
- `queueStatus`**: Estado de la cola en la que se procesó el documento.**
- `sourceStatus`**: Estado del procesador o entidad en ese momento específico.**
- `result`**: Resultado del procesamiento en formato JSON.**
- `inputFileName`**: Nombre del archivo de entrada.**
- `cost`**: Costo asociado al procesamiento.**
- `tokens`**: Tokens utilizados en el procesamiento.**

Ejemplo de JSON Final de "FinalSummary"

json

Copy code

```
{

 "createdAt": "04/09/2023",

 "metadata": {

 "documentName": "factura.txt",

 "documentType": "Factura",

 "documentSubcategory": {
```

```
  "category": "factura",
  "confidence": 0.9
 },
 "documentExtension": "txt"
},
"extractedVariables": { ... },
"verificators": { ... },
"queueStatus": "FinalizadoOK",
"isTaskDone": true
}
```

Consideraciones Importantes

- **Si el campo `documentSubcategory` o los contenidos en `verificators` están presentes, estos dependen de la configuración del paquete para el documento específico, y pueden variar en el resultado final.**
- **El campo `queueStatus` refleja el estado final de la cola, mientras que `isTaskDone` indica si la tarea se ha completado con un valor booleano.**

Esta documentación adicional proporciona una comprensión detallada del procesador "FinalSummary" y su relevancia en el análisis y verificación de documentos procesados por la API de Financial Intelligence.

Junto a la documentación se incluye un fichero Excel con el listado de ficheros generales de Smart Escrow, explicaciones y JSON de ficheros de prueba para que se pueda ver mejor la respuesta de la API y la forma de capturar la información.

Nombre Conector

WebScrapperVision
MyGestion
Testing
Borme
Csv Hacienda
ContaSimple
TeahorrrounclickVision
Holded
Buscador de bing
Catastro
Dalle-3
TeahorrrounclickScrapper
Analisis avanzado de videos
Transcribir Videos
Billage
Delsol
Shopify
WebScrapperPDF
Prestashop

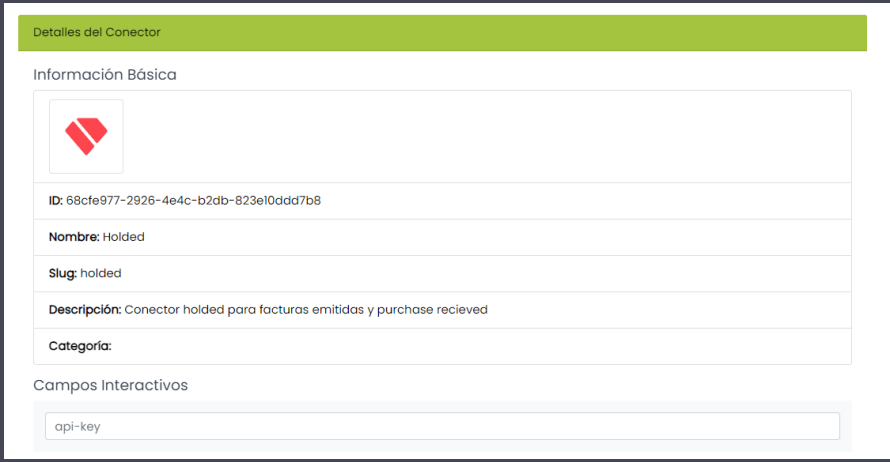
Información de Conectores

Los conectores difieren en términos de la información requerida para su uso. En el caso de ficheros, lo principal que se envía es la URL del fichero. Sin embargo, para los conectores, los requisitos son más variados e incluyen distintos tipos de datos como la clave API (Api-key), nombre de usuario y contraseña, entre otros.

A continuación, presentamos el listado actualizado de conectores disponibles en nuestra plataforma:

Para ver el funcionamiento y los requisitos de cada conector debemos utilizar la siguiente URL: cajaficheros.smartescrow.eu/help/connector/Nombre del conector

En este caso como ejemplo vamos a utilizar el conector Holded: cajaficheros.smartescrow.eu/help/connector/Holded



Podremos ver los detalles mas importantes del conector, una pequeña explicación de su uso y además los campos interactivos necesarios para su funcionamiento. Además, se nos muestra un ejemplo de la petición de Post a realizar:

Ejemplo de Petición para Conector con Guzzle

Para hacer una petición POST para un Conector usando Guzzle:

```
use GuzzleHttp\Client;

$client = new Client();
$response = $client->post('http://cajaficheros.smartescrow.eu/api/ask', [
  'headers' => [
    'api-key' => 'YOUR_API_KEY'
  ],
  'multipart' => [
    [
      'name'     => 'fileType',
      'contents' => 'Holded'
    ],
    [
      'name'     => 'jsonToFile',
      'contents' => '{"tipo":null,"api-key":"api-key"}'
    ]
  ]
]);
```

Nota: Reemplaza "YOUR_API_KEY" con tu clave de API.

En esta situación, el usuario solo necesita proporcionar su clave de API y usar este texto. Nuestro sistema, utilizando esta clave, automáticamente extrae la información del conector y genera un fichero. Este fichero es luego procesado por nuestro sistema

estándar, de la misma manera que lo haríamos con cualquier otro fichero que se suba a través de nuestros métodos habituales.