das-OCR

Advanced Optical Character Recognition for ID documents

Revisión	Fecha	Descripción	Redactado	Revisado	Aprobado
1	24/01/2018	das-OCR Documentation	SGP	EMR	EAL
2	29/05/18	Revision	LDM	MSY	MSY

1. Introduction

The purpose of this document is to describe the general structure of the services developed by Veridas and for this particular case, the functionalities of das-OCR service.

2. How it works?

2.1. das-OCR Extraction API

das-OCR is a ID document´s reading API.

This microservice performs Optical Character Recognition on the text present in both the obverse and reverse of the document.

The document images (obverse and reverse) can be entered into the API vía:

- Veridas Capture SDK's: mobile and HTML
- Customer capture systems: SDK's, escanners, a batch of files, etc

When the document´s images are located in the server, the installed services perform the following actions:

1. **Document classification**

   das-OCR performs a classification of the type of document, provided that it is indicated through the API the country to which the document corresponds.The country´s information can be inferred through the device's location and language, characteristics obtainable with Veridas´ Contextual Data SDK, available for IOS and Android. das-OCR also works if the user directly selects the type of document to be analyzed.

2. **Preprocessing and geometric adequation**

   in order to improve the quality of the OCR, the system improves the capture and quality of the document's image.

3. **Advanced OCR**

   Thanks to Veridas experience in **real** production environments and **Artificial Intelligence** technologies developed in-house, the API obtains a high quality result of the data written on the ID.

The system will return the OCR data in XML format together with the document type identification.

Once the OCR extraction is done, neither data nor document´s image will be stored.

2.1.1. Document classification

The services classifies the document within the available documents:

2.1.2. Preprocessing and geometric adequation

The system processes and corrects the orientation and the perspective of the document to adapt it to the analysis stages. This module allows to validate documents which have not been captured in optimal conditions.

2.1.3. Advanced OCR

The advanced OCR technology extracts both customer and document data, including text fields in both obverse and reverse and the MRZ (Machine Readable Zone), providing extraordinarily high accuracy levels:

- >99% in scanned documents.
- >98% in documents captured with Veridas mobile SDK.


das
BOI-DAS v3.2.1
Bienvenido eazanza
Nombre / Name
ESTEBAN
Apellidos / Last Names
DNI / DNI
Fecha de Validez / Expiration Date
11 03 2021
Número de Soporte / Support Number
BAZ138732
CAN / CAN
540224
Sexo / Gender
M
Nacionalidad / Nationality
ESP
Fecha de Nacimiento / Date of Birth
09 09 1989
Municipio de Nacimiento / Town of Birth
PAMPLONA
Provincia de Nacimiento / Province of Birth
NAVARRA
Domicilio / Address
PLZA CONDE
Municipio de Domicilio / Town of Residence
PAMPLONA
Provincia de Domicilio / Province of Residence
NAVARRA
ANVERSO SIN FLASH/ OBVERSE WITHOUT FLASH
ESPAÑA
DOCUMENTO NACIONAL DE IDENTIDAD
ESTEBAN
09 09 1989
BAZ138732
11 03 2021
540224
REVERSO SIN FLASH/ REVERSE WITHOUT FLASH
PLZA. CONDE DE
PAKPLONA
NAVARRA
PAKPLONA
NAVARRA
ESTEBAN / MARTA
IDESPE
8909095M2103112ESP<<<<<<<<<<<5
<<ESTEBAN<<<<<<<<<<


Corrected Image
Document OCR Service
Text List

Annex 1: API operation

1. Overview

The present document describes the API for the veriDas Document OCR Service. This service can be used to extract information (OCR) of documents from images of their obverse and reverse sides.

2. General

The following are some general considerations about this API that must be taken into account before consuming this service.

2.1. Requests

- For security reasons, the client should use the HTTPS protocol at all times to make requests to this API.
- The multipart/form-data content type will be used on every request.

2.2. Responses

- All responses will be encoded using the JSON format (regardless of the accepted content-type specified by the client).
- Responses will return a suitable HTTP status code indicating if the request was successful (20x) or not (HTTP error codes).
- Error responses will also include a code field that will provide more information about the concrete error on each case.

Successful responses will return a HTTP success code (usually 200) and the following minimal response body (empty JSON object):
{}
Each endpoint may return additional information (for example a POST request to /document will return the newly created document resource id)

In case of error, the following fields will be returned:

Field	Required	Description
code	yes	Error code, for example: `FormValidationError`, `UnknownDocumentTypeError`, etc.
message	no	A message providing more information about what went wrong.
errors	no	A list of field errors used in the case of the `FormValidationError` error.

Example:

```
{
	“code”: “UnknownDocumentTypeError”,
	“message”: “Unknown document type: Please make sure you are uploading a supported document or using a known document type.”
}
```

2.3. Versioning

The API version will be included in the URL, after the base url and before the endpoint:

```
https://<base_url>/v{number:integer}/<endpoint>
```

Non backwards compatible changes will cause a version increment. As of now, the API only supports the **v1** version.

3. API

This service is a REST API that lets the user extract OCR information about ID documents. The following endpoints are available:

3.1. Upload a new document

Creates a new document resource by uploading it’s obverse image (and optionally the reverse image too). The API will analyze the provided images. Returns the newly created document id (that will be used on subsequent requests) and it’s type.

POST `/v1/document`

Request:

Name	Req.	Type	Description
documentType	no	string	Document type. Example: “ID card20”, “ID card30” etc. If this is not specified, the service will attempt to classify the document.
obverse	yes	file	Document’s observe image file
reverse	no	file	Document’s reverse image file

Response:

Returns the new document id (randomly generated UUID) and it’s document type.

```
{
		“id”: “342c7cd91039472eaa26affb7b32ab71”,
		“documentType”: “ID card20”
```

```
}
```

Errors:

Code	HTTP Status	Message
`UnknownDocumentTypeError`	400	Unknown document type
`FormValidationError`	400	Missing obverse image
`UnexpectedError`	500	An unknown error has occured while processing the image

3.2. Update a document

Updates a document by uploading it's reverse image (reverse parameter), in case it hasn't been uploaded on the previous request.

PUT `/v1/document/{id}`

Request:

Name	Req.	Type	Description
reverse	no	file	Document's reverse image file

Response:

```
{}
```

Errors:

Code	HTTP Status	Message
`FormValidationError`	400	A reverse image must be provided
`ImageAlreadyAnalyzedError`	400	The provided image type has already been uplodaded and analyzed.
`NotFoundError`	404	The requested resource has not been found. Returned when trying to update a non-existent document.
`UnexpectedError`	500	An error has occured while processing the image

3.3. Get document information

Once finished uploading images, the user can retrieve the document information (nodes and scores) using this endpoint:

`GET /v1/document/{id}`

Request:

Nothing

Response:

The following information will be returned:

Field	Optional	Description
`id`	no	Document ID
`documentType`	no	Document type (i.e. `ID card30`)
`createdAt`	no	Creation date
`updatedAt`	no	Last udpate
`nodes`	no	Array of nodes
`scores`	no	Array of scores, indicating if the document is valid (`ValiDasVigenciaExpireDate1`) and the person is of full age (`ValiDasMayorEdadAdulthood1`).

Example:


```
{
	"createdAt": "2018-03-16 12:28:36.191418",
	"updatedAt": "2018-03-16 12:54:47.918027",
	"documentType": "ID card30",
	"id": "6e183452cc9e4c32b09802cd842b8fee",
	"nodes": [
		{
			"fieldName": "Nombre / Name",
			"name": "PD_Name_Out",
			"text": "John"
		},
		{
			"fieldName": "Apellidos / Last Names",
			"name": "PD_LastName_Out",
			"text": "Lennon"
```

```
},
{
            "fieldName": "ID card / ID card",
            "name": "PD_IdentificationNumber_Out",
            "text": "11223344Z"
},
{
            "fieldName": "Fecha de Validez / Expiration Date",
            "name": "DD_ExpirationDate_Out",
            "text": "06 03 2022"
},
{
            "fieldName": "Número de Soporte / Support Number",
            "name": "DD_DocumentNumber_Out",
            "text": "BDN112233"
},
{
            "fieldName": "CAN / CAN",
            "name": "OD_CAN_Out",
            "text": "112233"
},
{
            "fieldName": "Sexo / Gender",
            "name": "PD_Sex_Out",
            "text": "M"
},
{
            "fieldName": "Nacionalidad / Nationality",
            "name": "PD_Nationality_Out",
            "text": "ESP"
},
{
            "fieldName": "Fecha de Nacimiento / Date of Birth",
            "name": "PD_BirthDate_Out",
            "text": "02 04 1945"
},
{
            "fieldName": "Municipio de Nacimiento / Town of Birth
",
            "name": "PD_BirthPlaceMunicipality_Out",
            "text": "TAJONAR"
},
{
            "fieldName": "Provincia de Nacimiento / Province of
Birth",
            "name": "PD_BirthPlaceState_Out",
            "text": "NAVARRA"
},
{
            "fieldName": "Domicilio / Address",
```

```
                        "name": "PD_AddressStreet_Out",
                        "text": "Poligono Industrial Talluntxe II, Calle M-10"
                },
                {
                        "fieldName": "Municipio de Domicilio / Town of
                Residence",
                        "name": "PD_AddressMunicipality_Out",
                        "text": "TAJONAR"
                },
                {
                        "fieldName": "Provincia de Domicilio / Province of
                Residence ",
                        "name": "PD_AddressState_Out",
                        "text": "NAVARRA"
                }
        ],
        "scores": [
                {
                        "name": "ValiDasVigenciaExpireDate1",
                        "value": 1
                },
                {
                        "name": "ValiDasMayorEdadAdulthood1",
                        "value": 1
                }
        ]
}
```


3.3. Get document images

Gets (downloads) document images (both uploaded by the user and "cuts" generated by the system). Downloads the image or gets a 404 error if the image has not been uploaded or does not exist.

Obverse:

Resource	Description
`GET /v1/document/{id}/obverse`	Original obverse image uploaded by the user
`GET /v1/document/{id}/obverse/cut`	Obverse cut image
`GET /v1/document/{id}/obverse/cut_photo`	Photo cut
`GET /v1/document/{id}/obverse/signature`	Signature cut (where available)

Reverse:

Resource	Description
`GET /v1/document/{id}/reverse`	Original reverse image uploaded by the user
`GET /v1/document/{id}/reverse/cut`	Reverse cut image

3.4. Get document OCR data

Gets extracted data using OCR:

`GET /v1/document/{id}/ocr`

Note: The output of this function is the equivalent to the one included under the nodes key in the "Get Document Information" endpoint.

Returns a list of nodes (under the nodes key). Each node represents a piece of read text from the document and contains the following information:

Field	Description
`name`	Node name (unique identifier)
`fieldName`	Friendly node name. May be empty.
`text`	Read text

Response:

```
{
      "nodes": [
            {
                  "fieldName": "Nombre / Name",
                  "name": "PD_Name_Out",
                  "text": "John"
            },
            ...
      ]
}
```

Note: This method may not be available until all the images (obverse and reverse) have been uploaded and processed.

3.5. Get document scores

Gets document scores:

GET /v1/document/{id}/scores

Note: The output of this function is the same as the one included under the scores key in the "Get Document Information" endpoint.

Returns a list of scores (under the scores key). Each score represents a validation performed on the document:

Field	Description
`name`	Score name
`value`	Score value. Float number in the range [0,1].

Response:

```
{
	"scores": [
		{
			"name": "ValiDasVigenciaExpireDate1",
			"value": 1
		},
		{
			"name": "ValiDasMayorEdadAdulthood1",
			"value": 1
		}
	]
}
```

4. Example Workflow

In this section we describe an example workflow to analyze a document using it's obverse and reverse image. Let us suppose that the base URL of this API is:

https://thisisanexampleweb.com/ocrservice/api

and that our API secret key is:

`MySecretApiKey`

Step 1: Upload the obverse image

The first step is to upload the document's obverse image. In order to do that, we must issue a **POST** request to the /v1/document endpoint. For example:

Request:

```
curl -X POST \
  http://thisisanexampleweb.com/exampleocr/api/v1/document \
       -H 'cache-control: no-cache' \
       -H 'content-type: multipart/form-data' \
       -H 'apikey: MySecretApiKey' \
       -F obverse=@my_document_obverse_image.jpeg
```

In this case we're uploading an image called my_document_obverse_image.jpeg without a document type (the service will attempt to guess it)

Response:

HTTP status: 200

```
{
       "documentType": "ID card30",
       "id": "7ac97077fb7541bab96045eb81004e7c"
}
```

Step 2: Upload the reverse image

Now, we must upload the reverse image. We can do that by issuing a **PUT** request to the /v1/document/{id} endpoint specifying the document Id ({id} URL parameter) and the reverse image file (reverse body field):

Request:

```
curl -X PUT \
   http://veri-das.com/ocrservice/api/v1/document/7ac97077fb7541bab96045eb81004e7c
\
  -H 'cache-control: no-cache' \
  -H 'content-type: multipart/form-data' \
  -H 'apikey: MySecretApiKey' \
  -F reverse=@my_document_reverse_image.jpeg
```

Response:

HTTP status: 200

```
{}
```

Step 3: Get extracted text (OCR)

Once both images have been uploaded, we can get the extracted text using OCR by issuing a `GET` request to the `/v1/document/{id}` endpoint:

Request:

```
curl -X GET \

http://veri-das.com/ocrservice/api/v1/document/7ac97077fb7541bab96045eb81004e7c/ocr \
  -H 'apikey: MySecretApiKey' \
  -H 'cache-control: no-cache'
```

Response:

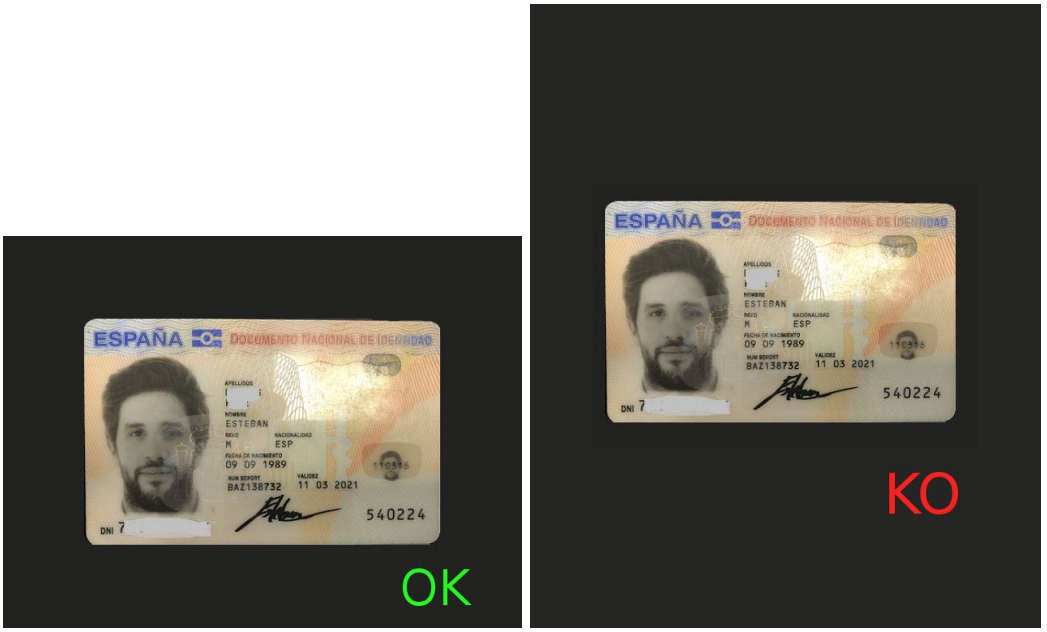
HTTP status: 200

```
{
      “nodes”: [
            {
                  "fieldName": "Nombre / Name",
                  "name": "PD_Name_Out",
                  "text": "John"
            },
            ...
      ]
}
```

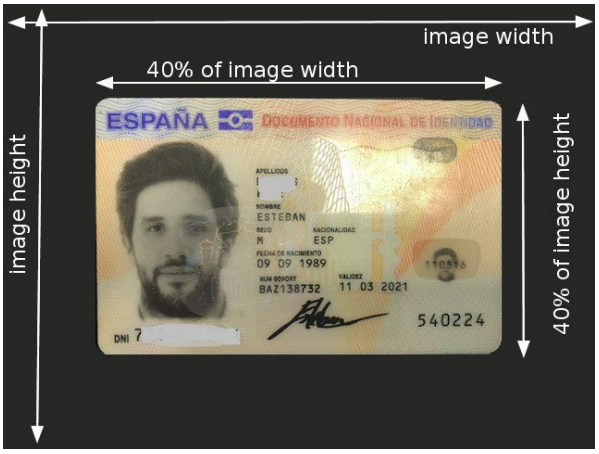
Annex 2: Input images requirements

1. The supported input image formats are JPEG, PNG and BMP.

2. Document orientation must be the same as the orientation of the image, i.e. the widest side of the document must correspond to the widest side of the image.



3. Document's width and height must be at least the 40% of the image's width and height. The document area should have a minimum resolution of 1000 x 600 pixels to enable text extraction. However, higher resolutions are recommended to reach optimal extraction.



4. Rotations up to 10º (left picture) and extreme perspective distortion (right picture) are corrected before document analysis.

5. Pictures that contain both the obverse and the reverse of a document are also supported. In this case, it is necessary to upload the picture to analyze the obverse and upload it again to analyze the reverse.