IDentidas

Document Validation Microservice - API

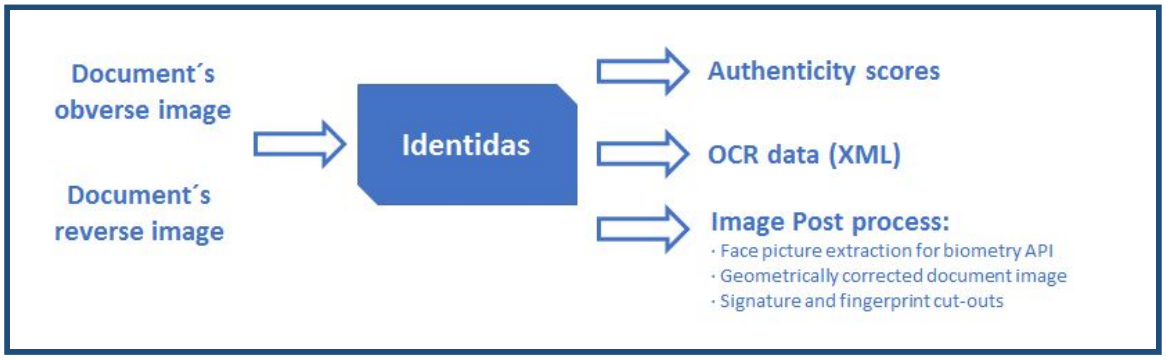
Revisión	Fecha	Descripción	Redactado	Revisado	Aprobado
1	24/01/2018	IDentidas Documentation	SGP	EMR	EAL
2	29/05/18	Revision	LDM	MSY	MSY

1. Identidas - Document Validation API | 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 Identidas: document validation API.

Thanks to das-Nano's experience in the High Security Printing Industry, Veridas has developed, patented and put in **real production** its Identidas technology that efficiently verifies the authenticity of documents using digital means.

Identidas validation API comprises a set of functions devoted to detect valid documents. In the following graphic, the input and outputs of the API are shown:



1. das-OCR - **(Identidas always needs das-OCR microservice to operate.)**
    a. Document Classification
    b. Document Adequation
    c. Document OCR
2. Document Validation
    a. Document anti-spoofing
    b. Pattern recognition
3. Document image post process - the system also provides as a result:
    a. Face picture extraction: that can be used in das-Face biometry engine.
    b. Front and Back geometry corrected Images
    c. Signature and Fingerprint cut-outs.

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

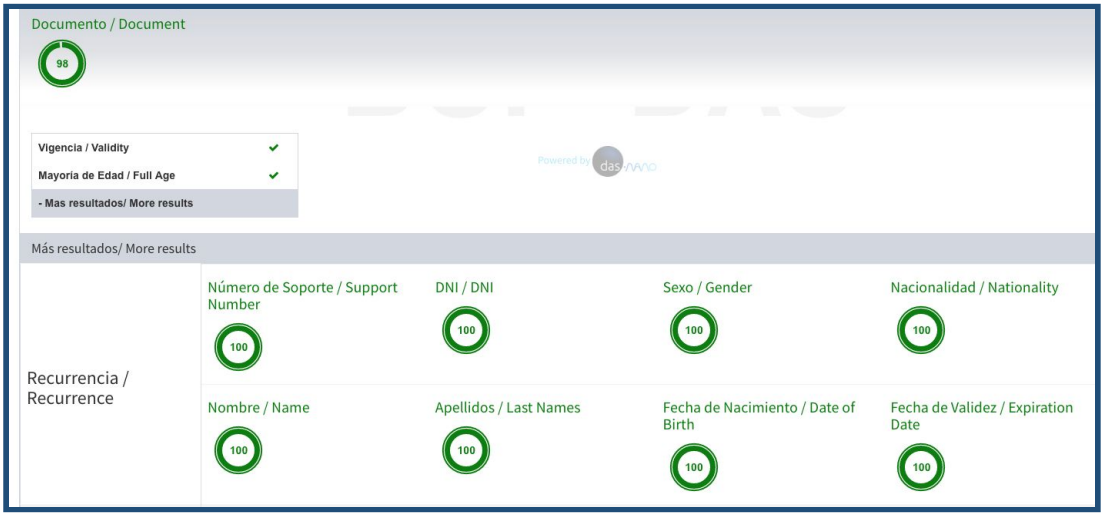
- Veridas Capture SDK's: mobile and HTML
- Customer capture systems: SDK's, escanners, a batch of files,etc.

At the end of the process, the system will provide a **document validity scores** and will **delete** all the data (except if the customer requires a Master Service with Data Storage).
The customer can set its own thresholds in order to determine whether a document is valid or not.

The following pictures illustrates, the full input and output of Identidas.

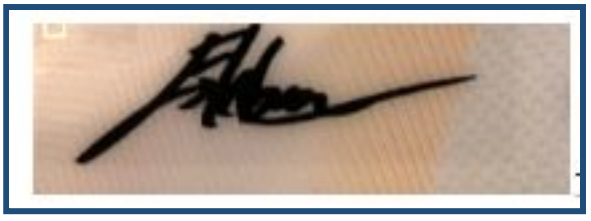
1.1. OCR data

1.2. Authenticity Scores



1.3. Post-processed Images

The Identidas API returns geometrically corrected pictures of the document, Face (for further biometric comparison), and signature and fingerprint (when requested).



2. How it works?

Identidas is the document validation service: In all cases it reads and classifies the document, in the same way as the das-OCR service does, and then it performs the document verification.

The scope of the document validation is divided into LITE or FULL validation depending on the technologies applied for its validation.

The documents bolded in the list below have FULL validation process. A Customer can request to move a document from LITE to FULL validation. *This specific development will be quoted separately.*

Identidas includes a set of patented technologies to detect invalid, forged or counterfeited documents that can be classified into four groups:

1. The coherence of the text reading (recurrence, MRZ checksums, logic, etc)
2. The detection of replicable patterns and features (patterns recognition, metatext, chip detection)
3. Other patterns and features that can´t be replicated without using a special printer or without the material used for the documents´production (micro-printing, CLI, Kinegram, OVI ink...). This elements are commonly defined, in the HSP sector, as level 1 security measures.
4. Verification of the Chip NFC present in certain documents: eg: Spanish ID card 3.0 and Passport according ICAO regulation (ePassport). These verifications provide a very high security level since the Chip contains not only the person's picture but also its personal data. For this reason, any physical manipulation of the document is detected immediately when reading the Chip in the verification process. Nowadays, only Android iOS and/or specific NFC reader allows to interact with the mentioned chips.

The validation level of the document is tightly linked to the type of device used (mobile, OS, HTML, and the type of document (NFC chip, security measures of the document itself,etc...)

In a decreasing order of security level:

1) Mobile Client - Android:
    - High definition camera: microprinting detection
    - Automatic capture SDK by Veridas: Improves the user capture, allowing a best evaluation of the document.
    - Control and use of the flash: Veridas patented detection technology of the level 1 security elements and document substrate.
    - Reading NFC of the document (in case it sports readable NFC chip)

2) Mobile Customer-iOS:
    - High definition camera: microprinting detection
    - Automatic capture SDK by Veridas: Improves the user capture, allowing a best evaluation of the document.
    - Control and use of the flash: Veridas patented detection technology of the level 1 security elements and document substrate.

3) HTML Client-Mobile & WEB:
    - High definition camera: Possible microprinting detection

This scope depends on the technology that is used to capture the image, the available hardware, as well as the document that is being considered.

2.1. LITE validation

LITE validation involves checking the identity document basic security measures. That includes, in general, the following concepts:

- Validation of the legal age of the subject.
- Validation of the validity of the document.
- Validation of the checksums and other logic aspects on the document (mod23, 731 rule, etc.).
- Validation of recurrent data on the document. This allows the service to obtain a consistency measure of the data that is present at least twice on the document.
- Text manipulation.

If the technology used in the images capture is HTML-based in a web cam, the document validation will be LITE. This is because the images captured with this technology do not have enough quality to be analyzed with the validation algorithm used by the FULL version, it doesn't allow to control the flashlight in a mobile environment, and it doesn't have access to NFC Chip.

2.2. FULL validation

FULL validation involves checking the identity document advanced security measures. It is understood that a document is available in a FULL version when it includes at least one more security measure from those available at the LITE version. It includes, in general, the following concepts:

- Validation of the authenticity of the document.
- Detection of document photocopies.
- Reading of the chip with NFC technology.
- Other security features.

In order to obtain a FULL validation level, the following requirements are needed:

1. iOS and Android mobile SDK from Veridas
2. HMTL capture in mobile environment: only minor verifications will be carried out.

Nevertheless, not every document in FULL version has the same degree of anti-counterfeiting verification: some countries implements in its ID documents different security measures, some of them verifiable by a mobile camera (holographic patches, OVI inks, 2D codas, etc) and NFC chips.

As an example, the Spanish ID 3.0 is currently the document with a higher security level among all the documents available, if considered in the scope of an Android capture SDK and a device with NFC reader. Under these conditions, the system is capable of reading the data contained in the chip (name, surnames, date of birth, etc.) and it compares them with the text fields printed in the document. The chip also includes a digitally stored copy of the photography printed in the document. Veridas system can read this image and verify its integrity with regard to the printed photo, thereby preventing fraud attempts that use a replacement of the printed image at the identity document.

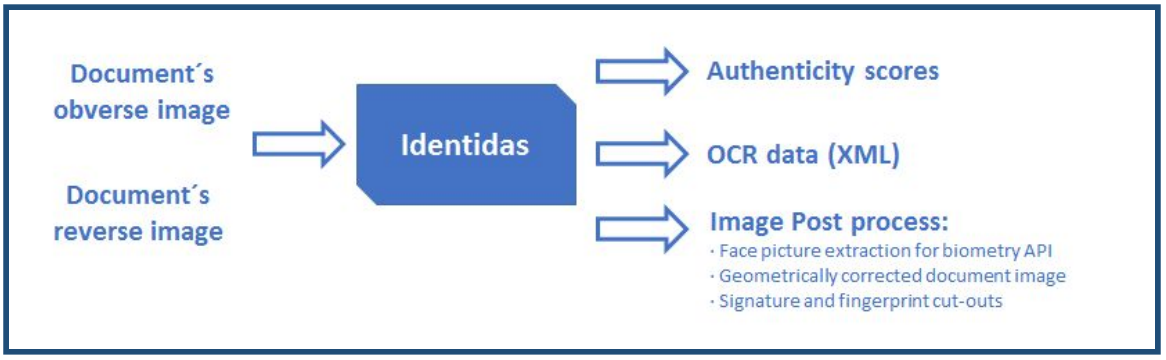
It is also remarkable that Identidas integrates the reading of MRZ in ICAO-compliant passports[1].

These types of documents are spread widely in most countries. In addition, for those passports that include a chip with NFC technology, Veridas' Android capture SDK allows the reading of this information and its subsequent comparison at the server in the same way as it is done for the Spanish ID 3.0.

[1] *See https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf*

3. Identidas flow

As presented in the introduction, Identidas API, has the following flow:



The Identidas API includes some sub-services inside:

3.1. das-OCR

As mentioned before, the Identidas API, needs the das-OCR API to operate. See das-OCR API documentation for more information.

3.2. Document anti-spoofing

Document Anti-spoofing step is created to detect possible document counterfeits. If a document is photocopied or plastified the anti-spoofing service will detect it.

3.3. Pattern Recognition

Pattern Recognition step extracts the different elements contained in the document and returns a list containing all the extracted patterns.
This service can detect different patterns of the document (chip, face, text patterns, security elements...)

3.4. Final score

After completing the steps mentioned above, the system will provide a document validity score. The customer will set its own thresholds in order to determine whether a document is valid or not.

3.5 Document´s Face Image Extraction

Identidas extracts the face image present in the document so it can be used for further facial recognition analysis.
As an example of this, das-Face uses the mentioned image to compare it to the selfie captured in the digital on-boarding system carried out by Validas.

Annex 1: API operation

1. Overview

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

1.1. Requests

- The multipart/form-data content type must be used on every request.

1.2. 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 new 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.”
}
```

1.3. 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.

2. API

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

2.1. Upload a new document (obverse image)

Creates a new document resource by uploading it’s obverse image (and optionally the reverse and/or obverse with flash images too) using a POST request. The API will analyze the provided images. Returns the new 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: “DNI20”, “DNI30” etc. If this is not specified, the service will attempt to classify the document.
obverse	yes	file	Document’s observe image file (jpg or png)
reverse	no	file	Document’s reverse image file (jpg or png)
obverseFlash	no	file	Document’s obverse with flash image file (jpg or png). This should only be used on the “Validation” service mode (it will be ignored on the “OCR” mode as it is not needed).
supportsNfc	no	boolean	Indicates if the user’s device supports NFC.

The following document types are available:

- `DNI20` : DNI 2.0 (Spain)
- `DNI30` : DNI 3.0 (Spain)
- `NIE2003`: NIE 2003 (Spain)
- `NIE2010`: NIE 2010 (Spain)
- `NIE2011` : NIE 2011 (Spain)
- `IFE2002`: Mexico 2002 (Mexico)
- `IFE2008`: Mexico 2008 (Mexico)
- `IFE2013`: Mexico 2013 (Mexico)
- `IFEE`: Mexico E (Mexico)
- `MYS2001`: Malaysia 2001 (Malaysia)
- `ARG2009`: Argentina 2009 (Argentina)
- `ARG2012`: Argentina 2012 (Argentina)
- `PE_IDCard_2007`: Peru 2007 (Peru)
- `es_PE_IDCard_2013`: Peru 2013 (Peru)
- `CO_IDCard_2000`: Colombia 2000 (Colombia)
- `Passport`: Passport (International)

The classifier is able to recognize the following document types (used if the user does not specify one):

- `DNI20`
- `DNI30`
- `NIE2010`
- `NIE2011`

Response:

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

```
{
        "id": "342c7cd91039472eaa26affb7b32ab71",
        "documentType": "DNI20"
}
```

Errors:

Below is a list of possible errors returned by this endpoint and their meaning:

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

2.2. Update a document (additional images)

Updates a document by uploading its reverse image (reverse parameter) and/or obverse with flash image (obverseFlash parameter) when using the validation mode.

PUT `/v1/document/{id}`

Request:

Name	Req.	Type	Description
reverse	no	file	Document's reverse image file (jpg or png)
obverseFlash	no	file	Document's obverse with flash image file (jpg or png)

Response:

```
{}
```

Errors:

Code	HTTP Status	Message
`FormValidationError`	400	At least one 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.
`InvalidDocumentServiceM`	400	Document was created using a different

odeError		service mode
UnexpectedError	500	An error has occured while processing the image

2.3. Get document NFC configuration

This endpoint retrieves the NFC configuration for the document, extracted from the document type and the OCR text found in the obverse image:

GET /v1/document/{id}/nfc

Request:

Nothing

Response:

Returns the document's NFC keys (fields stored on the the document's NFC chip that contain personal information) and pins, used to unlock and read the NFC chip on the user device:

```
{
	"nfcKeys": [
		"DG1_NAME",
		"DG1_SURNAME",
		"DG1_DOC_NUMBER",
		"DG1_DOB",
		"DG1_DOE",
		"DG1_NATIONALITY",
		"DG1_OPT_DATA",
		"DG1_SEX",
		"DG11_PERS_NUM",
		"DG1_ADDR_LINE",
		"DG1_ADDR_CITY",
		"DG1_ADDR_REGION",
		"DG11_BIRTH_PLACE”
	],
	"pinValues": [
		"BDN1112233",
		"112233",
		"870402"
	]
}
```

2.4. Upload NFC data

This endpoint allows the user to upload NFC information. This information must be previously obtained from the document's chip using a NFC compatible user device. The user must upload the required fields specified by the "Get Document NFC Configuration" endpoint (nfcKeys).

PUT `/v1/document/{id}/nfc`

Request:

A multipart/form-data containing all the required fields. Example for the DNI30 document type:

Name	Req.	Type	Description
dg1_sex	yes	string	Sex
dg1_opt_data	yes	string	Optional data
dg1_doc_number	yes	string	Document number
dg1_nationality	yes	string	Nationality (i.e. ESP for Spain)
dg11_pers_num	yes	string	Document ID
dg1_name	yes	string	First Name
dg1_addr_region	yes	string	Region
dg1_addr_city	yes	string	City
dg1_surname	yes	string	Last Name
dg1_addr_line	yes	string	Address
dg1_dob	yes	string	Date of birth
dg1_doe	yes	string	Date of expiration
dg11_birth_place	yes	string	Place of birth

Response:

```
{}
```

Errors:

Code	HTTP Status	Message
`FormValidationError`	400	
`UnexpectedError`	500	

2.5. Get document information

Once finished uploading images and/or NFC data, 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. `DNI30`)
`createdAt`	no	Creation date
`updatedAt`	no	Last udpate
`nodes`	no	Array of nodes
`scores`	no	Array of scores

Example:


```
{
        "createdAt": "2018-03-16 12:28:36.191418",
        "updatedAt": "2018-03-16 12:54:47.918027",
        "documentType": "DNI30",
        "id": "6e183452cc9e4c32b09802cd842b8fee",
        "nodes": [
                {
```

```
            "fieldName": "Nombre / Name",
            "name": "PD_Name_Out",
            "text": "John"
        },
        {
            "fieldName": "Apellidos / Last Names",
            "name": "PD_LastName_Out",
            "text": "Lennon"
        },
        {
            "fieldName": "DNI / DNI",
            "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": "ValiDasMRZPaisExpedicionValue1",
                "value": 1
            },
            {
                "name": "ValiDasMRZFechaDeValidezRegular1",
                "value": 1
            },
            {
                "name": "ValiDasMRZFechaDeValidezDate2",
                "value": 1
            },
            ...
        ]
}
```


2.6. Get a list of documents

The user can retrieve the document information (nodes and scores) of all the analysed documents by using this endpoint:

`GET` /v1/document?page=1&perPage=5&sort=-createdAt

This endpoint has few basic functionalities for pagination and ordering through these parameters:

- **page**: Current page
- **perPage**: number of items per page (to a maximum of 100)
- **sort**: Ordering field (in camel case). If it is prefixed with a dash (“-”) the ordering is ascending.
- **include**: It says if the respones has to include nodes and/or scores (just for validation). to include both, the value “nodes,scores” (separated with a comma).

Response:

The following information will be returned:

Field	Optional	Description
`id`	no	Document ID
`documentType`	no	Document type (i.e. `DNI30`)
`createdAt`	no	Creation date
`updatedAt`	no	Last udpate
`nodes`	no	Array of nodes
`scores`	no	Array of scores

Example:

GET /api/v1/document?page=1&perPage=5&sort=-createdAt

que devolverá una respuesta similar a la siguiente:


```
{
        "items": [
        {
        "createdAt": "2018-07-24 10:36:34.107677",
        "documentType": "DNI30",
        "id": "53c2576069054396bd691854c2db3da1",
        "updatedAt": "2018-07-24 10:37:30.515838"
        },
        {
        "createdAt": "2018-07-24 10:23:08.455553",
        "documentType": "DNI30",
        "id": "08312526e0514f1cb493b5507d5babe4",
        "updatedAt": null
        },
        {
        "createdAt": "2018-07-24 09:44:15.851416",
        "documentType": "DNI20",
        "id": "bf823598ec904d3ab38dbc6f67649a79",
```

```
            "updatedAt": null
            },
            {
            "createdAt": "2018-07-24 09:44:02.550436",
            "documentType": "DNI20",
            "id": "2fe7330e28a6424383ea86d503fcad49",
            "updatedAt": null
            },
            {
            "createdAt": "2018-07-24 09:43:37.467386",
            "documentType": "DNI20",
            "id": "1837d394df754f3787670ea6cbccfa32",
            "updatedAt": null
            }
            ],
            "page": 1,
            "pages": 3,
            "perPage": 5,
            "total": 14
}
```


where:

items: list with the retrieved documents. Each document contains the same information and style than the response of one document.
page: current page
pages: number of pages
perPage: items per page
total: total number of documents

Note: If a page that does not exists is requested, a 404 error is returned.

2.7. 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

Obverse with flash:

Resource	Description
GET /v1/document/{id}/obverse_flash	Original obverse with flash image uploaded by the user
GET /v1/document/{id}/obverse_flash/cut	Obverse with flash cut image

2.8. 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”: [
		{
			“name”: “MRZ”,
			“fieldName”: “MRZ”,
			“text”: “Texto del MRZ leído”
		}
	]
}
```

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

2.9. 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": "ValiDasMRZPaisExpedicionValue1",
			"value": 1
		},
		{
			"name": "ValiDasMRZFechaDeValidezRegular1",
			"value": 1
		},
		{
			"name": "ValiDasMRZFechaDeValidezDate2",
			"value": 1
```

```
            },
            ...
        ]
}
```

3 Example Workflow

In this section we describe an example workflow (with CURL examples) to show how this API can be used in the "Validation" mode.

Let us suppose that the base URL of our service is:

`https://anyweb.com/anyid/api`

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://anyweb.com/anyid/api/v1/document \
       -H 'cache-control: no-cache' \
       -H 'content-type: multipart/form-data' \
       -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": "DNI30",
       "id": "7ac97077fb7541bab96045eb81004e7c"
}
```

The service returns the new document resource id (7ac97077fb7541bab96045eb81004e7c) and its type (DNI30). The id field will be used in the following requests.

Step 2: Upload the obverse with flash image

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

Request:

```
curl -X PUT \
  http://anyweb.com/anyid/api/v1/document/7ac97077fb7541bab96045eb81004e7c \
  -H 'cache-control: no-cache' \
  -H 'content-type: multipart/form-data' \
  -F obverseFlash=@my_document_obverse_flash_image.jpeg
```

Response:

HTTP status: 200

```
{}
```

Step 3: 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://anyweb.com/anyid/api/v1/document/7ac97077fb7541bab96045eb81004e7c \
  -H 'cache-control: no-cache' \
  -H 'content-type: multipart/form-data' \
  -F reverse=@my_document_reverse_image.jpeg
```

Response:

HTTP status: 200

```
{}
```

Step 4: Get document NFC configuration

Once the images have been uploaded, we can retrieve the NFC configuration (on compatible document types) and use the returned pins and keys to read the NFC data on the client side (using a compatible phone). In order to do that we must issue a **GET** request to the /v1/document/{id}/nfc endpoint:

Request:

```
curl -X GET \
  http://anyweb.com/anyid/api/v1/document/7ac97077fb7541bab96045eb81004e7c/nfc \
  -H 'cache-control: no-cache' \
  -H 'content-type: multipart/form-data'
```

Response:

HTTP status: 200

```
{
	"nfcKeys": [
		"DG1_NAME",
		"DG1_SURNAME",
		"DG1_DOC_NUMBER",
		"DG1_DOB",
		"DG1_DOE",
		"DG1_NATIONALITY",
		"DG1_OPT_DATA",
		"DG1_SEX",
		"DG11_PERS_NUM",
		"DG1_ADDR_LINE",
		"DG1_ADDR_CITY",
		"DG1_ADDR_REGION",
		"DG11_BIRTH_PLACE”
	],
	"pinValues": [
		"BDN1112233",
		"112233",
		"870402"
	]
}
```

Step 5: Upload document NFC data

With the NFC pins and keys values retrieved in the previous step, we can unlock and read the data contained in the NFC chip of the document. Once we have that information, we can upload it to the service by doing a **PUT** request to the endpoint /v1/document/{id}/nfc:

Request:

```
curl -X PUT \
  http://anyweb.com/anyid/api/v1/document/7ac97077fb7541bab96045eb81004e7c/nfc \
  -H 'cache-control: no-cache' \
  -H 'content-type: multipart/form-data’
  -F dg1_sex=M \
  -F dg1_opt_data=11223344W \
  -F dg1_doc_number=BDN112233 \
  -F dg1_nationality=ESP \
  -F dg11_pers_num=11223344W \
  -F dg1_name=John \
  -F dg1_addr_region=NAVARRA \
  -F dg1_addr_city=TAJONAR \
  -F 'dg1_surname=Lennon' \
  -F 'dg1_addr_line=Poligono Industrial Talluntxe II, Calle M-10' \
  -F 'dg1_dob=02 04 1987' \
  -F 'dg1_doe=06 03 2022' \
  -F 'dg11_birth_place=TAJONAR NAVARRA'
```

Response:

HTTP status: 200

```
{}
```

Step 6: Get document information

At this point, we've finished uploading data and the service has analyzed the document images and NFC data. We can issue a **GET** request to the `/v1/document/{id}` endpoint to get the nodes and scores:

Request:

```
curl -X GET \
  http://anyweb.com/anyid/api/v1/document/7ac97077fb7541bab96045eb81004e7c \
  -H 'cache-control: no-cache' \
  -H 'content-type: multipart/form-data'
```

Response:

HTTP status: 200

```
{
```

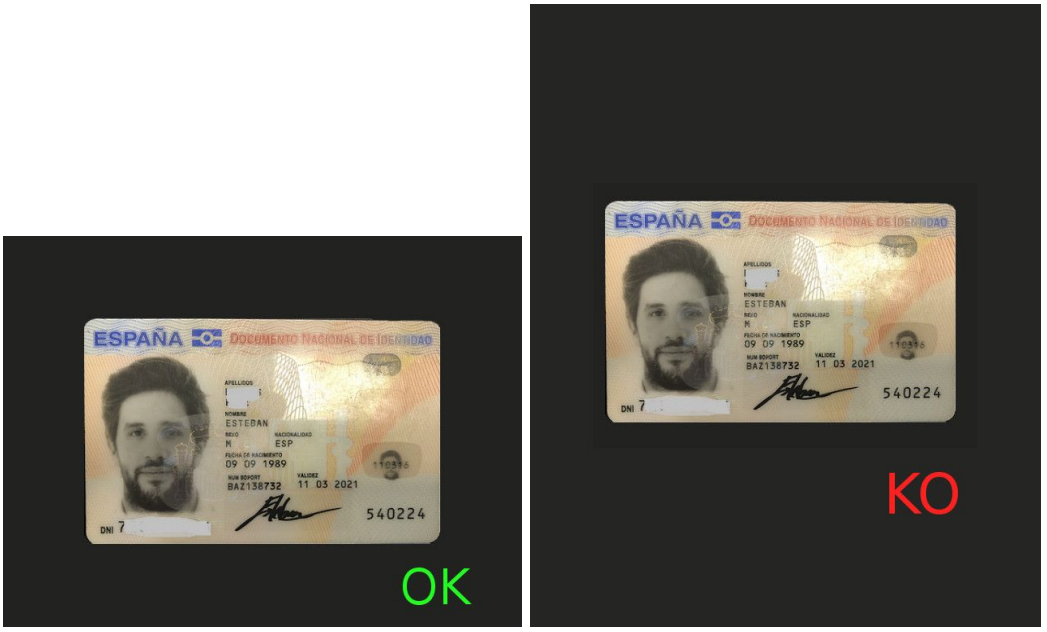
```
"createdAt": "2018-03-16 12:28:36.191418",
"updatedAt": "2018-03-16 12:54:47.918027",
"documentType": "DNI30",
"id": "7ac97077fb7541bab96045eb81004e7c",
"nodes": [
        {
                "fieldName": "Nombre / Name",
                "name": "PD_Name_Out",
                "text": "John"
        },
        {
                "fieldName": "Apellidos / Last Names",
                "name": "PD_LastName_Out",
                "text": "Lennon"
        },
        {
                "fieldName": "DNI / DNI",
                "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": "ValiDasMRZPaisExpedicionValue1",
                        "value": 1
                },
                {
                        "name": "ValiDasMRZFechaDeValidezRegular1",
                        "value": 1
                },
                {
                        "name": "ValiDasMRZFechaDeValidezDate2",
                        "value": 1
                },
                ...
        ]
}
```

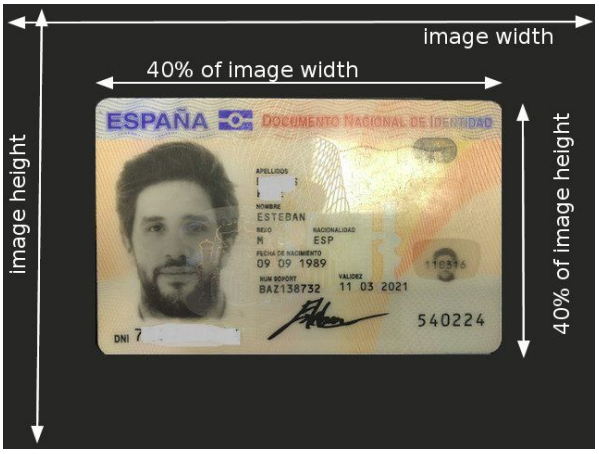
Annex 2: Input images requirements

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.