Intelligent IDXtract

Developer Guide

Version: 1.1 SP1

Disclaimer

This document contains information proprietary to Newgen Software Technologies Limited. Users may not disclose or use any proprietary information or use any part of this document without written permission from Newgen Software Technologies Limited.

Newgen Software Technologies Limited makes no representations or warranties regarding any software or the contents or use of this manual. It also specifically disclaims any express or implied warranties of merchantability, title, or fitness for any particular purpose. Even though Newgen Software Technologies Limited has tested the hardware and software and reviewed the documentation, it does not guarantee or imply that this document is error-free or accurate regarding any particular specification. As a result, this product is sold as it is and the user, the purchaser, is assuming the entire risk as to its quality and performance. Further, Newgen Software Technologies Limited reserves the right to revise this publication and make changes in its content without any obligation to notify any person, of such revisions or changes. Newgen Software Technologies Limited authorizes no Newgen agent, dealer, or employee to make any modification, extension, or addition to the above statements.

Newgen Software Technologies Limited has attempted to supply trademark information about company names, products, and services mentioned in this document. Trademarks indicated below were derived from various sources.

Table of contents

1 Preface

This developer guide describes the development and integration of Intelligent IDXtract 1.1 SP1.

1.1 Revision history

Revision Date	Description
February 2024	Initial publication

1.2 Intended audience

This developer guide is intended for the developers from internal or external product implementation teams who integrate IDXtract APIs with another product or platform. The reader must be comfortable with API signature formats in JSON to understand API requests and responses. Administrative rights might be required to perform the integration operations.

1.3 Documentation feedback

To provide feedback or any improvement suggestions on technical documentation, you can write an email to docs.feedback@newgensoft.com.

To help capture your feedback effectively, share the following information in your email:

- Document name
- Version
- Chapter, topic, or section
- Feedback or suggestions

2 Introduction to Intelligent IDXtract

Intelligent IDXtract is the next-generation extraction solution. It combines the best of artificial intelligence (AI) and machine learning (ML) technologies with rules-based capabilities to efficiently extract data virtually from any identity document. It leverages AI and ML to identify specific data zones from ID cards and eliminates the need for unique labels or identifiers as required by conventional systems.

Intelligent IDXtract comes integrated with advanced AI and ML-based image processing and OCR capabilities, facilitating accurate data capturing from mobile-captured ID cards and improved data capture rates on scanned identity documents. The flexible and configurable post-processing capabilities of the application enhance performance.

It provides support for REST web services and includes the *Zone extraction* endpoint.

NOTE:
All the service URLs are examples only. You must change it as per the deployment of the Intelligent IDXtract.

3 Zone extraction

The *zoneextraction* endpoint takes Base64 encoded images and fields as a request for extraction and redaction. As requested, it provides the extracted field information based on the trained model as per the fields along with their zoned and redacted Base64 encoded image.

The *zoneextraction* endpoint provides an interface for ID card optical character recognition (OCR) extraction. For example, you can use this endpoint to extract Aadhar card details such as name, date of birth, and gender.

The following are the endpoint details:

- **Endpoint** – */zoneextraction*
- **Service URL** – *http://127.0.0.1:9100/zoneextraction*
- **Method type** – POST
- **Request content type** – *application/json*
- **Response content type** – *application/json*

3.1 Request parameters

The following table describes the request parameters of the *zoneextraction* endpoint:

Parameter	Data Type	Parameter Type	Mandatory	Default Value	Description
img (In)	Base64_string	Body	Yes	NA	An image in a base64_string format.
zoning fields(In)	List	Body	Yes	NA	A list of all the field names that require zone identification.
redaction fields(In)	List	Body	Yes	NA	A list of all the field names that require redaction of zone area. Call getredactionobjects() to get the list of redaction fields.
extraction fields(in)	List	Body	Yes	NA	A list of all the field names that require extraction. Call getextractionobjects() to get the list of extraction fields.
enable redaction(In)	Boolean	Body	Yes	true	It is a flag value to enable or disable redaction. Specify: • true to enable redaction. • false to disable redaction.

Parameter	Data Type	Parameter Type	Mandatory	Default Value	Description
enable extraction(In)	Boolean	Body	Yes	true	It is a flag value to enable or disable extraction. Specify: • true to enable extraction. • false to disable extraction.
image name(In)	String	Body	Yes	NA	Name of the image.
return zoned image	Boolean	Body	Yes	true	It is a flag value that returns the zone-marked image. Specify: • true to return the zone-marked image. • false to not get the zone-marked image.
return redacted image	Boolean	Body	Yes	true	It is a flag value that returns the redacted image. Specify: • true to return the zone-marked image. • false to not return the redacted image.

3.2 Sample request

```
{
"img" : "zoned image",
"zoning fields" :
"eida_front_nationality","face","eida_front_name","eida_front_chip","eida_front_idno"],
"redaction fields" : ["face","eida_front_idno"],
"extraction fields"   :[
"eida_front_nationality","face","eida_front_name","eida_front_chip","eida_front_idno"]
"enable redaction" : true,
"enable extraction" : true,
"return zoned image" : true,
"image name" : image_1
}
```

3.3 Response parameters

The following table describes the response parameters of the *zoneextraction* endpoint:

Parameter	Data Type	Description
objects	List of JSON data	Refer to the below table for details.
status	Integer	Status of the response returned by the API.
uuid	Alpha-numeric	The Universally Unique Identifier (UUID) number for the image.
zoned image	Base64_string	It specifies the image with zones marked.
redacted image	Base64_string	It specifies the image with redacted zones marked.

The following table describes the details of the *objects* parameter in the output:

Parameter	Data Type	Description
name	String	The field name of the identified zone.
confidence_yolo	Float	The confidence level of the identified zone.
coordinates	List	Coordinates of identified zones Format – [x1, y1, x2, y2] The values of x1, y1, x2, and y2 are in pixels.
data	String	The final extracted data of the field.
confidence	Float	The final confidence level of the field.
data_ocr	String	Extracted data from the model coordinates.
confidence_ocr	Float	The confidence level of extraction results.
data_qr	String	Data extracted from the QR code.
data_mrz	String	Data extracted from MRZ.

3.4 Sample response


```
{
  "objects": [
    {
      "name": "face",
      "confidence_yolo": 93,
      "coordinates": [
        695,
        509,
        798,
        601
      ]
    },
    {
      "name": "eida_front_chip",
      "confidence_yolo": 92.8,
```

```
      "coordinates": [
        76,
        517,
        285,
        670
      ]
    },
    {
      "name": "eida_front_name",
      "confidence_yolo": 93,
      "coordinates": [
        121,
        722,
        462,
        769
      ],
      "data": "Hassan Baloch Soban Baloch",
      "data_ocr": "Hassan Baloch Soban Baloch",
      "confidence": 87.2,
      "confidence_ocr": 87.2
    },
    {
      "name": "eida_front_idno",
      "confidence_yolo": 93,
      "coordinates": [
        321,
        571,
        557,
        606
      ],
      "data": "784-1986-7285262-5",
      "data_ocr": "784-1986-7285262-5",
      "confidence": 91,
      "confidence_ocr": 91
    },
    {
      "name": "eida_front_nationality",
      "confidence_yolo": 92.3,
      "coordinates": [
        168,
        808,
        283,
        856
      ],
      "data": "Pakistan",
      "data_ocr": "Pakistan",
      "confidence": 57,
      "confidence_ocr": 57
    }
```

```
  ],
  "status": 200,
  "uuid": "224391392061773-S8r4QPpK-138945415318758281",
  "zoned image": "/9j/4AAQSkZJRgABAQAAAQABAAD//Z",
  "redacted image": "/9j/4AAQSkZJRgABAQAAAQABAAD//9k="
}
```


3.5 Status codes

The following table describes the various status codes that can appear in the response:

Status Code	Meaning
200	Success
201	Tampered database
202	Exceeded extraction limit
204	License expired
205	Metering validation error
300	Missing configuration file
301	Missing or incorrect model details
302	Missing database details
303	Unsupported database
304	Missing extraction configuration file
305	Invalid parameter number 2.
306	Invalid parameter number 3
307	Invalid parameter number 4
401	Internal server error
402	Invalid image file
403	No data received
404	Error in the decoding file.
407	No support for multipage image processing
408	Unidentified fields from class mapping
409	No output returned by the mode
701	Unable to load redaction objects for the YOLO model
702	Unable to load extraction objects for the YOLO model

3.6 Get extraction objects

The *getextractionobjects()* function provides a list of fields available for extraction.
The following table describes the response parameters of the *getextractionobjects()* function:

Parameter	Data Type	Description
extraction_names	List	A list of fields for extraction.

Return value:

```
Return JSON type data output (extraction_names)
```

Sample of *getextractionobjects()* output JSON

```
{
    "extraction_names": [
        "eida_front_name",
        "eida_front_nationality",
        "eida_front_idno",
    ],
    "status": 200
}
```

3.7 Get redaction objects

The *getredactionobjects()* function provides a list of redaction fields.
The following table describes the response parameters of the *getredactionobjects()* function:

Parameter	Data Type	Description
name	List	A list of fields for redaction.

Return value:

```
Return JSON type data output (redaction_objects)
```

Sample of *getredactionobjects()* output JSON

```
{
    "names": [
        "1d_barcode",
        "2d_barcode",
        "face",
    ],
    "status": 200
}
```

4 Sample driver code

A sample driver code to call Intelligent IDXtract API for extraction and redaction:

```
from pickle import FALSE
import requests
import os
import time
import ast
import filetype
import io
import json
import base64
from urllib3.exceptions import InsecureRequestWarning

addr = 'http://127.0.0.1:9100'
test_url = addr + '/zoneextraction'
redaction_url = addr + '/get_objects'

# add path to the location of the folder containing images on which extraction is
to be done
folderpath=r'C:\Users\aanchal.a\Desktop\IDXtract\Samples'

def get_filetype(path):
        type = filetype.guess_extension(path)
        return type

files = os.listdir(folderpath)
total_count = len(files)

for i,file in enumerate(files):
        file_path = os.path.join(folderpath, file)
        type = get_filetype(file_path)

        if type != 'pdf':
                img = open(file_path, 'rb').read()

        headers = {'content-type' : 'application/json'}
        img = base64.b64encode(img)

        base64_string = img.decode('utf-8')
        extraction_fields_list
=['dl_front_license_no','dl_front_name','dl_front_nationality','dl_front_dob','dl_f
ront_doi']
        redaction_fields = requests.post(redaction_url,
headers=headers,verify=False)

        input_data = {
```

```
        "img" : base64_string,
        "zoning fields" :extraction_fields_list,
        "redaction fields" :extraction_fields_list,
        "extraction fields" : extraction_fields_list,
        "enable redaction" : True,
        "enable extraction" : True,
        "return zoned image" : True,
        "image name":file
        }
        response = requests.post(test_url, data=json.dumps(input_data),
headers=headers)

        print(response.json()['objects'])
```