Intelligent Document Classifier

Developer Guide

Version: 2.1

Disclaimer

Newgen Software, Registered Office, New Delhi
E-44/13
Okhla Phase - II
New Delhi 110020
India
Phone: +91 1146 533 200
info@newgensoft.com

Contents

Preface

This chapter provides information about the purpose of this guide, details on the intended audience, revision history, and related documentation for the Intelligent Document Classifier (IDC) Developer Guide.

Revision history

Revision date	Description
August 2023	Initial publication

About this guide

This developer guide provides information about the integration of the Intelligent Document Classifier.

Intended audience

This developer guide is intended for the developers from internal or external product implementation teams responsible for integrating IDC 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.

Related documents

The following documents are related to IDC Developer Guide:

- Intelligent Document Classifier Release Notes
- Intelligent Document Classifier Installation and Configuration Guide
- Intelligent Document Classifier Deployment Guide
- Intelligent Document Classifier Best Practices Guide

Documentation feedback

To provide feedback or any improvement suggestions on technical documentation, write an email to docs.feedback@newgen.co.in.

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

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

Introduction to IDC

IDC is a document classification engine that automatically identifies various structural and textual attributes within the image to perform classification. IDC does classification based on structural (Document Image Classification), template-based (Document Object Classification), textual (Document Content Classification) features, or a combination of all of these.

IDC is a Python-based Web API that takes a document as input and utilizes deep learning algorithms to extract structural, template, and textual features of the document for classification.

Document Image Classification analyzes the structural aspect of the document image and identifies individual document types within large packets or folders. It utilizes pre-processing, augmentation and primarily employs a deep-learning approach for image classification. The in-house augmentation module boosts the learning model's performance. For run-time categorization, the classification module is supported by a segmentation module that removes unnecessary backgrounds from document images.

Document Content Classification analyzes the textual pattern of document images and identifies important words to classify them. It is suitable for classifying documents that lack proper structure or format. The content classification engine accepts input in the form of either images or text and provides classification along with confidence. It also includes classification and verification modules to reduce false-positive cases.

Document Object Classification is designed specifically to work on fixed template-based identity cards such as Aadhaar, PAN, EIDA Card, and more. It analyzes important regions within the document image for classification. The object model incorporates advanced augmentation techniques and builds an object localization model to identify significant regions in the document. There are two available models: Small and Large. You can select them based on the complexity of your use case. The object model internally applies various verification techniques to avoid false positives and does not require segmentation techniques to function.

Prerequisites

The following prerequisites are required for using the IDC 2.1 Client component:

- Tesseract OCR 4.0 must be installed.
- Microsoft Visual C++ 2015-2019 Redistributable must be installed.

getModelConfiguration()

The *getModelConfiguration()* method enables you to obtain the model configuration of the IDC 2.1 model.

Request parameters

The following table describes the request parameters of this method:

Parameter	Input type	Mandatory	Description
host	String	Yes	The URL of the IDC 2.1 server.

Sample request

```
docclassifierClient.getModelConfiguration("host");
```

Where *docclassifierClient* is an object of the *DocumentClassifierClient* class.

For example:

```
docclassifierClient.getModelConfiguration("https://qanumbertheory.
newgensoftware.net");
```

Or

```
docclassifierClient.getModelConfiguration("http://127.0.0.1:8080");
```

Response parameters

The following table describes the response parameters of this method:

Return type	Description
String	A string containing the model configuration of the deployed model, with possible values being Smart, Content, or Image.

Sample response

```
Image
```

getPrediction()

The *getPrediction()* method enables you to obtain page-wise results for a multi-page TIFF file. When you call this method, it returns a Map where each page number serves as the key, and the corresponding value is a *PredictResponse* object. The *PredictResponse* object contains all the prediction-related results.

Request parameters

The following table describes the request parameters of this method:

Parameter	Input type	Mandatory	Description
image	File	Yes	A TIFF file containing multiple pages used for prediction.
host	String	Yes	The URL of the IDC 2.1 server.
threadCount	String	Yes	The number of threads used for parallel processing of multiple-page TIFF files.
null	String	Yes	If the OCR is in JSON format, it gets represented as a JSONObject; otherwise, as null.
segment	String	No	The possible values for the segment are "N" or "Y", with the default value being "N". If the segment value is set to Y, the image gets segmented to locate the document within the page.

Sample request

```
docclassifierClient.getPrediction(image, "host", threadCount, null);
```

Where *docclassifierClient* is an object of the *DocumentClassifierClient* class.

For example:

```
docclassifierClient.getPrediction(file, "https://qa-
numbertheory.newgensoftware.net", 4, null);
```

Or

```
docclassifierClient.getPrediction(file, "http://127.0.0.1:8010", 4, null);
```

Response parameters

A Map, where the page number serves as the key, and the *PredictResponse* object acts as the corresponding value is received in response

PredictResponse object parameters

The following table describes the parameters of the *PredictResponse* object:

Parameter	Return type	Description
getLicenseStatus	String	The current status of the license.
getTransactionId	String	The unique ID for each processed request.
getSessionId	String	The unique ID of the session.
getStatus	String	The status of the prediction.
getProtocol	String	It specifies the distinctive protocol value that determines the version on which the request is being processed.
getModelId	String	The unique ID of the model.
getPrediction	String	The prediction of the model.
getGroup	String	*For future use
getConfidence	String	The confidence score of the prediction.
getPageNumber	String	The number of the current page in the input document.

Parameter	Return type	Description
getStructuredPrediction	String	The value of the image model prediction.
getStructuredConfidence	String	The image model confidence score.
getStructuredClassifierPrediction	String	The image model prediction with the classifier.
getStructuredClassifierConfidence	String	The image model confidence with the classifier.
getStructuredVerifierPrediction	String	The image model prediction with the verifier.
getStructuredVerifierConfidence	String	The image model confidence with the verifier
getStructuredOrientation	String	The orientation of the image.
getUnstructuredPrediction	String	The content model prediction.
getUnstructuredConfidence	String	The content model confidence.
getUnstructuredLPrediction	String	The linear model prediction.
getUnstructuredPPrediction	String	The probabilistic model prediction.
getUnstructuredPConfidence	String	The probabilistic model confidence.
getUnstructuredAllConfidence	ArrayList<String>	The list of all confidence scores.
getUnstructuredDPrediction	String	The dictionary model prediction.
getUnstructuredDScore	String	The dictionary model score.
getUnstructuredCoordinates	String	The coordinates of the textual content.
getUnstructuredWords	String	It gives none.
getUnstructuredStatus	String	The status of the process.
getOutputStage	String	*For future use
getMessage	String	The status message of the final process.

Sample response

```
getCandidateInfo()::null

getPageInfo()::null
getLicenseStatus()::active
getTransactionId()::64d1e5bfd99bba5337fe1c8a
getSessionId()::null
getStatus()::200
getProtocol()::v2.0
getModelId()::image_v1.0
getPrediction()::Form 1040
getGroup()::
getConfidence()::90.82
getPageNumber()::0
getStructuredPrediction()::Form 1040
getStructuredConfidence()::90.82
getStructuredClassifierPrediction()::Form 1040
getStructuredClassifierConfidence()::90.82
getStructuredVerifierPrediction()::None
getStructuredVerifierConfidence()::0
getStructuredOrientation()::0
getUnstructuredPrediction()::
getUnstructuredConfidence()::0
getUnstructuredLPrediction()::
getUnstructuredPPrediction()::
getUnstructuredPConfidence()::
getUnstructuredAllConfidence()::[]
getUnstructuredDPrediction()::
getUnstructuredDScore()::
getUnstructuredCoordinates()::0
getUnstructuredWords()::0
getUnstructuredStatus()::0
getOutputStage()::
getMessage()::Success
getStructuredSegmentsPrediction()::None
getStructuredSegmentsConfidence()::None
getStructuredSegmentsCoordinates()::None
getTextualPatternResponse()::null
getContent():{}
```

getPredictionSinglePage()

The *getPredictionSinglePage()* method enables you to obtain prediction results for a single-page TIFF file. It returns a Map where the page number serves as the key, and the corresponding value is a *PredictResponse* object. The *PredictResponse* object contains all the prediction-related results.

Request parameters

The following table describes the request parameters of this method:

Parameter	Input type	Mandatory	Description
image	File	Yes	A TIFF file containing multiple pages used for prediction.
host	String	Yes	The URL of the IDC 2.1 server.
tessObject	Tesseract	Yes	An instance of tess4j Tesseract used for Optical Character Recognition (OCR) of an image.
configuration	String	No	The configuration of the IDC 2.1 model, that can be set as Smart, Content, or Image.
segment	String	No	The possible values for the segment are "N" or "Y", with the default value being "N". If the segment is set to Y, the image gets segmented to locate the document within the page.

Sample request

```
docclassifierClient.getPredictionSinglePage(image, host, tessObject);
```

Where *docclassifierClient* is an object of the *DocumentClassifierClient* class.

For example:

```
docclassifierClient.getPredictionSinglePage(file, https://qa-
numbertheory.newgensoftware.net, tessObject);
```

Or

```
docclassifierClient.getPredictionSinglePage(file, http://127.0.0.1:8080,
tessObject);
```

Response parameters

A Map, where the page number serves as the key, and the *PredictResponse* object acts as the corresponding value is received in response

PredictResponse object parameters

Refer to the PredictResponse object parameters section.

Sample response

Refer to the Sample response section.

submitFeedback()

The *submitFeedback()* method allows you to provide feedback when a document is incorrectly classified by the Document Classifier. If you need to retrain the image, you can send it back along with the feedback. To do so, the user must include the *transactionId* obtained from the predict API request, along with details of the incorrect classification, predicted category, and actual category in your feedback.

Request parameters

The following table describes the request parameters of this method:

Parameter	Input type	Mandatory	Description
imageFile	File	Yes	An image file representing a single page used for feedback.
host	String	Yes	The URL of the IDC 2.1 server.
actualCategory	String	Yes	The actual category to which the document belongs.
predictedCategory	String	Yes	The prediction provided by the IDC 2.1 model.
confidence	String	Yes	The confidence score of the prediction given by the IDC 2.1 model.
transactionId	String	Yes	The transaction ID acquired from the IDC 2.1 server for the misclassified image.

Sample request

```
docclassifierClient.submitFeedback(imageFile, host, "actualCategory",
"predictedCategory", "confidence", "transactionId"
```

Where *docclassifierClient* is an object of the *DocumentClassifierClient* class.

Response parameters

An object of type *SubmitResponse* is received from the IDC 2.1 engine.

SubmitResponse object parameters

The following table describes the parameters of the *SubmitResponse* object:

Parameter	Return type	Description
getStatus	Integer	The status of the submitted feedback.
getTransactionId	String	The unique ID for each submitted feedback.
getMessage	String	The status message of the final feedback.

Sample repsonse

```
getStatus()::200
getTransactionId()::64d0bea3f5d343aad5413b7d
getMessage()::success
```

Common codes

The table below lists the codes and their descriptions, that are common to all the method explained in this guide:

Code	Message	Description	Workaround (if any)
200	Success	Process completed successfully	Not applicable
300	Blank model	The provided model path is empty.	Provide the correct model path in the *config.ini* file.
301	Protocol mismatch	The value of the protocol does not match.	Ddd the correct protocol value.
302	Token mismatch	The given token does not match.	Verify the input token.
304	Page exhausted	The limit of the document classification license has been exhausted.	Obtain a new license key to continue.
305	License verification failed	The entered license key is incorrect.	Enter the correct license key.
401	Model internal server error occurred	The model did not work as expected.	Check the configuration parameters and input files.
402	Invalid image file	The provided image file is incorrect.	Check the image file format and ensure that the file is not corrupted.
403	No data received	Input data is missing.	Provide the required data.
404	Error decoding file	The input file failed to decode.	Check if the file is corrupted.
405	Incorrect image size	The input image file is incorrect.	Enter the correct input file.
406	Not enough text	The extracted OCR text is insufficient.	Use a different document.

Code	Message	Description	Workaround (if any)
407	Multipage image exception	The input document is a multi-page file.	Input a single-page file.
409	Requested model not available	The requested model type is not available.	Add the relevant model to the correct path.
500	Token missing	The token is missing.	Add the token.
501	Configuration missing	The configuration is missing.	Add the configuration.
502	Content-type missing	The content type is missing.	Provide the content type.
503	Configuration and content-type mismatch	The configuration and content-type values mismatch.	Enter the correct values.
504	Empty result image	The image file is missing.	Check if the image is missing in the input.
505	Classification error - content	The OCR content is missing.	Enter the OCR content in the input.
506	DB exception	There is an exception in the database.	Check if the database is running.
601	Content type missing in submit request	The content type is missing.	Add the content type.
602	Prediction missing in submit request	The predicted class is missing.	Add the predicted class.
603	Confidence missing in submit request	The confidence value is missing.	Add the confidence value.
604	True label missing in submit request	The modified label is missing.	Add the true label.
606	Internal server error in submit request	The feedback API is not working.	Check the feedback API endpoint.
5002	License key expired	The license has expired.	Get a renewed license key.

Code	Message	Description	Workaround (if any)
5003	License expired	You have exceeded the assigned number of pages, indicating that the number of page uses available with the license is over.	Get a renewed license key.
5004	Number of concurrent users limit exceeded	The API is receiving more concurrent requests than it can handle.	Decrease the number of requests to reduce the load from the server.
5005	Number of classes limit exceeded	Limit of the number of classes has exceeded.	Not applicable
5006	License key not provided	The license key is missing.	Provide the license key.
5007	License is already registered	The entered license key is already in use.	Enter a different license key.
-5000	Exception occurred in server	The server is not working as expected.	Not applicable
5009	Invalid source of request	Wrong input request.	Check the input request.
5010	Invalid source of request	Wrong input request.	Check the input request.
5011	Model not available	The model is missing from the designated path.	Check the model and its correct path.

Configuring IDC

This section lists the various configurations used in IDC 2.1 Client components.

The following table provides information on the different tags available for customization in the *config.ini file* available in the *DocumentClassifierClient.jar* file:

Parameter	Description
threadCount	This tag determines the number of threads used for parallel tasks when using the *getPrediction()* method for processing multiple-page TIFF files.
hostname	The tag holds the value of IDC 2.1 server URL. If the host is not explicitly provided in the *getPrediction()*, *getPredictionSinglePage()*, *getModelConfiguration()*, or *submitFeedback()* methods, the server URL specified in the *config.ini* file is used as the default.
generateLogs	This flag accepts either "true" or "false" values. When set to true, it triggers the generation of debugging and error logs.
tessdataPath	This flag holds the value of the path to the *tessdata* folder used by Tesseract OCR.
language	This flag determines OCR document language. The default value is "eng."
ocrTimeout	This flag determines the time duration after which the OCR processing of a document gets aborted.
fileSizeLimit	This flag determines the size limit of each file in megabyte (MB) in the context of multiple page TIFF files. It represents the average size of a page within the TIFF file.
singleDocument	This flag determines the processing of single document.
pageLimit	This flag determines the number of pages for processing in a document.
--pageLimit	This flag determines the range of pages for processing in a document. The "--" indicates that the number of pages value gets ignored while processing the document.

Parameter	Description
charLimit	This flag determines the condition for textual extraction. If the characters are less than 150, OCR gets executed. Otherwise, the text gets extracted form the textual document without using OCR.
ocrFlag	This flag returns OCR value in response body.
ocrEngine	The flag holds the name of the engine used for performing OCR.
--ocrEngine	The flag holds the name of the engine used for performing OCR. The "--" indicates that the engine value gets ignored while performing the OCR.
gv_key_path	This flag holds the Google Cloud Vision JSON key copied from the console.
gv_json_path	This flag holds the path of the JSON key to access Google Cloud Vision API, such that the client ID and private key ID can be extracted from the service account.
usePythonForFileSeparation	This flag determines the splitting of PDF into images using Python.
crop_x1_percent	This flag represents the percentage value for the designated region where OCR is executed. The value of x1 signifies the OCR execution starting point as a percentage measured from the top of the document.
crop_x2_percent	This flag represents the percentage value for the designated region where OCR is executed. The value of x2 signifies the OCR execution starting point as a percentage measured from the bottom of the document.
crop_y1_percent	This flag represents the percentage value for the designated region where OCR is executed. The value of y1 signifies the OCR execution starting point as a percentage measured from the left of the document.
crop_y2_percent	This flag represents the percentage value for the designated region where OCR is executed. The value of y2 signifies the OCR execution starting point as a percentage measured from the right of the document.
x_csrftoken	This flag holds the value of CSRF token.