

# Document AI

---

## Introduction

This repository contains a collection of AWS Lambda APIs that uses OpenAI's GPT-4 chat models to extract information from documents using a predefined JSON schema. The API uses `gpt-4-1106-preview` model for `TXT` documents and `gpt-4-vision-preview` for `JPG/PNG/TIFF/BMP` documents. For `PDF` documents the API converts the PDF to images and uses the `gpt-4-vision-preview` model. For more information on the models, see [OpenAI's API documentation](#).

The application is deployed using [AWS CDK](#) and the following AWS services are used:

- `API Gateway` - manage API endpoints
- `Lambda` - manage schema and extract data
- `DynamoDB` - store schema, extracted data, and request status
- `S3` - store documents to be processed
- `SQS` - manage asynchronous processing of documents
- `Secrets Manager` - manage OpenAI API key
- `Systems Manager Parameter Store` - store application configuration and AWS resource names
- `IAM` - manage permissions for AWS resources

All the Lambda functions share a common layer contained in the `src/layers/docai` folder, and the Lambda functions are contained in the `src/functions` folder. While the CDK application is contained in the `resources` folder.

## Dependencies

The following

- `OSX`, `Linux` or `Windows`
- `Python 3.12`
- `Node 20.x`
- `AWS CDK 2.118.9`
- `Poetry 1.5.0`

### Python Dependencies

- `boto3`
- `aws-lambda-powertools[parser,tracer]`
- `titoken`
- `jsonschema`
- `openai`
- `pymupdf`
- `aws-cdk-lib`

### Python Development Dependencies

- `isort`

- [ruff](#)
- [pre-commit](#)
- [black](#)
- [mypy](#)

## Installation

The [docai](#) layer can be installed for development purposes using [poetry](#). Use the following command to install the layer for local development

```
cd src/layers/docai && poetry install
```

## Deployment

The command below synthesizes the CDK application and deploys all application resources to AWS

```
make deploy
```

## API Usage

To use the API, you will need to create an API key in the API Gateway console. The API key should be passed in the [x-api-key](#) header of the request. In addition, the [<api-id>](#) should be replaced with the API ID of the deployed API.

### Schema Management

A schema is a JSON schema that defines the structure of the data that to be extracted from a document. The description nodes of fields are used to provide prompt instructions, hints, and examples to the model on what and how to extract structured data from the document.

#### Create Schema

This endpoint creates a schema that can be used to extract information from a document.

```
curl -X POST \  
  -H "Content-Type: application/json" \  
  -H "x-api-key:<api-key>" \  
  -d @sample.json \  
  https://<api-id>.execute-api.ca-centra-1.amazonaws.com/prod/create-  
schema
```

#### Sample Request Data

```
{
  "schema_name": "test_loe",
  "schema_description": "Test extracts data from letter of employment",
  "schema_definition": {
    "$id": "https://example.com/test_loe.schema.json",
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "title": "Employment Letter",
    "description": "A schema that represents data extracted from an
employment letter",
    "type": "object",
    "properties": {
      "employer_name": {
        "description": "The name of the employer",
        "type": ["null", "string"]
      }
    },
    "required": ["employer_name"],
    "additionalProperties": false
  }
}
```

#### Sample Response Data

```
{
  "OK": true,
  "result": {
    "schema_name": "test_loe",
    "schema_version": "rvByviQNVb",
    "number_of_tokens": 102
  }
}
```

#### Get Schema

This endpoint returns the schema definition for a given schema name and version.

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "x-api-key:<api-key>" \
  -d @sample.json \
  https://<api-id>.execute-api.ca-centra-1.amazonaws.com/prod/get-schema
```

#### Sample Request Data

```
{
  "schema_name": "test_loe",
```

```
"schema_version": "rvByviQNVb"
}
```

#### Sample Response Data

```
{
  "OK": true,
  "result": {
    "schema_name": "test_loe",
    "schema_description": "Test extracts data from letter of employment",
    "schema_definition": {
      "$schema": "https://json-schema.org/draft/2020-12/schema",
      "description": "A schema that represents data extracted from an
employment letter",
      "additionalProperties": false,
      "title": "Employment Letter",
      "type": "object",
      "properties": {
        "employer_name": {
          "type": ["null", "string"],
          "description": "The name of the employer"
        }
      },
      "required": ["employer_name"],
      "$id": "https://example.com/test_loe.schema.json"
    },
    "schema_version": "rvByviQNVb",
    "schema_status": "ACTIVE",
    "number_of_tokens": 102,
    "created_at": "2024-01-06T02:09:28.036892"
  }
}
```

#### List Schemas

This endpoint returns a list of schemas that have been created for a given schema name.

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "x-api-key:<api-key>" \
  -d @sample.json \
  https://<api-id>.execute-api.ca-centra-1.amazonaws.com/prod/list-
schema
```

#### Sample Request Data

```
{
  "schema_name": "test_loe"
}
```

#### Sample Response Data

```
{
  "OK": true,
  "result": {
    "count": 2,
    "items": [
      {
        "schema_description": "Test extracts data from letter of
employment",
        "created_at": "2024-01-05T17:16:20.992628",
        "number_of_tokens": 234,
        "schema_name": "test_loe",
        "schema_version": "9jBVrISmXA",
        "schema_status": "ACTIVE",
        "schema_definition": {
          "description": "Schema to extract the employer name, employee
name, and employee salary from a letter of employment",
          "additionalProperties": false,
          "$schema": "http://json-schema.org/draft/2020-12/schema",
          "type": "object",
          "properties": {
            "employer_name": {
              "type": ["string", "null"],
              "description": "The name of the employer in the letter of
employment"
            },
            "employee_name": {
              "type": ["string", "null"],
              "description": "The name of the employee in the letter of
employment"
            },
            "employee_salary": {
              "description": "The salary information of the employee in
the letter of employment",
              "type": "object",
              "properties": {
                "amount": {
                  "type": ["number", "null"],
                  "description": "The amount of the salary"
                },
                "frequency": {
                  "type": ["string", "null"],
                  "description": "The frequency of the salary payment",
                  "enum": [
                    "annually",
                    "monthly",

```

```

        "biweekly",
        "weekly",
        "hourly"
    ]
}
},
"required": ["amount", "frequency"]
}
},
"required": ["employer_name", "employee_name",
"employee_salary"]
}
},
{
    "schema_description": "Test extracts data from letter of
employment",
    "created_at": "2024-01-05T17:18:35.254293",
    "number_of_tokens": 234,
    "schema_name": "test_loe",
    "schema_version": "Gcpmg1c79",
    "schema_status": "ACTIVE",
    "schema_definition": {
        "description": "Schema to extract the employer name, employee
name, and employee salary from a letter of employment",
        "additionalProperties": false,
        "$schema": "http://json-schema.org/draft/2020-12/schema",
        "type": "object",
        "properties": {
            "employer_name": {
                "type": ["string", "null"],
                "description": "The name of the employer in the letter of
employment"
            },
            "employee_name": {
                "type": ["string", "null"],
                "description": "The name of the employee in the letter of
employment"
            },
            "employee_salary": {
                "description": "The salary information of the employee in
the letter of employment",
                "type": "object",
                "properties": {
                    "amount": {
                        "type": ["number", "null"],
                        "description": "The amount of the salary"
                    },
                    "frequency": {
                        "type": ["string", "null"],
                        "description": "The frequency of the salary payment",
                        "enum": [
                            "annually",
                            "monthly",
                            "biweekly",

```

```

        "weekly",
        "hourly"
    ]
  },
  },
  "required": ["amount", "frequency"]
},
},
"required": ["employer_name", "employee_name",
"employee_salary"]
}
}
]
}
}

```

## Delete Schema

This endpoint deletes a schema for a given schema name and version.

```

curl -X POST \
  -H "Content-Type: application/json" \
  -H "x-api-key:<api-key>" \
  -d @sample.json \
  https://<api-id>.execute-api.ca-centra-1.amazonaws.com/prod/delete-
schema

```

### Sample Request Data

```

{
  "schema_name": "test_loe",
  "schema_version": "rvByviQNVb"
}

```

### Sample Response Data

```

{
  "OK": true,
  "result": {
    "message": "Schema deleted successfully"
  }
}

```

## Data Extraction

These endpoints are used to extract data from documents using a predefined schema. For `image/*` and `application/pdf` documents, the content is first converted to a `base64` string and then passed to the API. The API will then convert the `base64` content to an image and then extract the data from the image. For `text/plain` documents, the content is passed directly to the API. Check out `samples/b64string` folder for sample base64 data for testing the api

## Extract Data

This endpoint extracts data from a document using a predefined schema. It process document in an `ONLINE` manner, meaning that the document is processed when the API is called. The API will return the extracted data in the response.

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "x-api-key:<api-key>" \
  -d @sample.json \
  https://<api-id>.execute-api.ca-centra-1.amazonaws.com/prod/extract-
data
```

### Sample Request Data

```
{
  "schema_name": "test_loe",
  "schema_version": "rvByviQNVb",
  "content": "This is a letter of employment for John Smith. John Smith is
employed by Acme Inc. John Smith's salary is $100,000.00 per year.",
  "mime_type": "text/plain"
}
```

### Sample Response Data

```
{
  "OK": true,
  "result": {
    "request_id": "d4e0fbfb-ac56-4166-9f50-9ad0ba592f9f",
    "data": {
      "employer_name": "Acme Inc."
    }
  }
}
```

The `request_id` can be used to retrieve the extracted data using `/get-result` endpoint if the result is required at a later time.

## Extract Data Batch



This endpoint extracts data from documents using a predefined schema in a batch mode. The request is processed asynchronously and the extracted data can be retrieved using the `/get-result` endpoint.

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "x-api-key:<api-key>" \
  -d @sample.json \
  https://<api-id>.execute-api.ca-centra-1.amazonaws.com/prod/extract-
data-batch
```

#### *Sample Request Data*

```
{
  "schema_name": "test_loe",
  "schema_version": "rvByviQNVb",
  "content": "This is a letter of employment for John Smith. John Smith is
employed by Acme Inc. John Smith's salary is $100,000.00 per year.",
  "mime_type": "text/plain"
}
```

#### *Sample Response Data*

```
{
  "OK": true,
  "result": {
    "request_id": "97e5e420-17e1-4e12-bd8c-8d4771e6f3b5",
    "status": "QUEUED",
    "created_at": "2024-01-06T02:31:19.891472"
  }
}
```

### Get Result

This endpoint retrieves the extracted data for a given `request_id`. The `request_id` from an `/extract-data` or `/extract-data-batch` request can be used to retrieve the extracted data.

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "x-api-key:<api-key>" \
  -d @sample.json \
  https://<api-id>.execute-api.ca-centra-1.amazonaws.com/prod/get-result
```

#### *Sample Request Data*

```
{  
  "request_id": "97e5e420-17e1-4e12-bd8c-8d4771e6f3b5"  
}
```

#### *Sample Response Data*

```
{  
  "OK": true,  
  "result": {  
    "created_at": "2024-01-06T02:31:25.974506",  
    "request_id": "97e5e420-17e1-4e12-bd8c-8d4771e6f3b5",  
    "status": "COMPLETED",  
    "data": {  
      "employer_name": "Acme Inc."  
    }  
  }  
}
```