Processing Documents Synchronously

Amazon Textract can detect and analyze text in single-page documents that are provided as images in JPEG, PNG, PDF, and TIFF format. The operations are synchronous and return results in near real time. For more information about documents, see Text Detection and Document Analysis Response Objects.

This section covers how you can use Amazon Textract to detect and analyze text in a single-page document synchronously. To detect and analyze text in multipage documents, or to detect JPEG and PNG documents asynchronously, see Processing Documents Asynchronously.

You can use Amazon Textract synchronous operations for the following purposes:

- Text detection – You can detect lines and words on a single-page document image by using the DetectDocumentText operation. For more information, see Detecting Text.
- Text analysis – You can identify relationships between detected text on a single-page document by using the AnalyzeDocument operation. For more information, see Analyzing Documents.
- Invoice and receipt analysis – You can identify financial relationships between detected text on a single-page invoice or receipt using the AnalyzeExpense operation. For more information, see Analyzing Invoices and Receipts
- Identity document analysis – You can analyze identity documents issued by the US Government and extract information along with common types of information found on identity documents. For more information, see Analyzing Identity Documents.

Topics

Calling Amazon Textract Synchronous Operations

Amazon Textract operations process document images that are stored on a local file system, or document images stored in an Amazon S3 bucket. You specify where the input document is located by using the Document input parameter. The document image can be in either PNG, JPEG, PDF, or TIFF format. Results for synchronous operations are returned immediately and are not stored for retrieval.

For a complete example, see Detecting Document Text with Amazon Textract.

Request

The following describes how requests work in Amazon Textract.

Documents Passed as Image Bytes

You can pass a document image to an Amazon Textract operation by passing the image as a base64-encoded byte array. An example is a document image loaded from a local file system. Your code might not need to encode document file bytes if you're using an AWS SDK to call Amazon Textract API operations.

The image bytes are specified in the `Bytes` field of the `Document` input parameter. The following example shows the input JSON for an Amazon Textract operation that passes the image bytes in the `Bytes` input parameter.

```
{
    "Document": {
        "Bytes": "/9j/4AAQSk....."
    }
}
```

Note

If you're using the AWS CLI, you can't pass image bytes to Amazon Textract operations. Instead, you must reference an image stored in an Amazon S3 bucket.

The following Java code shows how to load an image from a local file system and call an Amazon Textract operation.

```
String document="input.png";

ByteBuffer imageBytes;
try (InputStream inputStream = new FileInputStream(new File(document))) {
    imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
}
AmazonTextract client = AmazonTextractClientBuilder.defaultClient();

DetectDocumentTextRequest request = new DetectDocumentTextRequest()
        .withDocument(new Document()
                .withBytes(imageBytes));


DetectDocumentTextResult result = client.detectDocumentText(request);
```

Documents Stored in an Amazon S3 Bucket

Amazon Textract can analyze document images that are stored in an Amazon S3 bucket. You specify the bucket and file name by using the S3Object field of the `Document` input parameter. The following example shows the input JSON for an Amazon Textract operation that processes a document stored in an Amazon S3 bucket.

```
{
    "Document": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.png"
        }
    }
}
```

The following example shows how to call an Amazon Textract operation using an image stored in an Amazon S3 bucket.

```
String document="input.png";
String bucket="bucket";

AmazonTextract client = AmazonTextractClientBuilder.defaultClient();

DetectDocumentTextRequest request = new DetectDocumentTextRequest()
        .withDocument(new Document()
                .withS3Object(new S3Object()
                        .withName(document)
                        .withBucket(bucket)));

DetectDocumentTextResult result = client.detectDocumentText(request);
```

Using an adapter

With Amazon Textract, you can customize the output or response of a call to AnalyzeDocument by using an AdapterId and adapter version. To use an adapter, you must first have created and trained an adapter using the Amazon Textract Console or the API. To apply your adapter, provide its ID when calling the AnalyzeDocument API. This enhances predictions on your documents. Note that when calling the Document, you can only use one adapter per page.

```
"AdaptersConfig": {
      "Adapters": [
         {
            "AdapterId": "2e9bf1c4aa31",
            "Version": "1"
         }
      ]
   }
```

The following Java example shows how to create Queries and a list of adapters, then provide these to AnalyzeDocument, using the AWS SDK for Java.

```
String document="input.png";

        ByteBuffer imageBytes;
        try (InputStream inputStream = new FileInputStream(new File(document))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
        }
        AmazonTextract client = AmazonTextractClientBuilder.defaultClient();

        List<Query> queries = new ArrayList<Query>();
        queries.add(new Query().withText("What is the employee name?")
                .withAlias("CUST_NAME")
                .withPages(Arrays.asList("*")));

        List<Adapter> adapters = new ArrayList<Adapter>();
        adapters.add(new Adapter()
                .withAdapterId("1111111111")
                .withVersion("1")
                .withPages(Arrays.asList("*")));
        AnalyzeDocumentRequest request = new AnalyzeDocumentRequest()
                .withFeatureTypes("QUERIES", "SIGNATURES")
                .withDocument(new Document()
                        .withBytes(imageBytes))
                .withAdaptersConfig(new AdaptersConfig()
                        .withAdapters(adapters));

        AnalyzeDocumentResult result = client.analyzeDocument(request);
```

Response

The following sample is the JSON response from a call to `DetectDocumentText`. For more information, see Detecting Text.

```
{
{
  "DocumentMetadata": {
    "Pages": 1
  },
  "Blocks": [
    {
      "BlockType": "PAGE",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.9995205998420715,
          "Height": 1.0,
          "Left": 0.0,
          "Top": 0.0
        },
        "Polygon": [
          {
            "X": 0.0,
            "Y": 0.0
          },
          {
            "X": 0.9995205998420715,
            "Y": 2.297314024515845E-16
          },
          {
            "X": 0.9995205998420715,
            "Y": 1.0
      "BlockType": "LINE",
      "Confidence": 99.93761444091797,
      "Text": "Employment Application",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3391372561454773,
          "Height": 0.06906412541866302,
          "Left": 0.29548385739326477,
          "Top": 0.027493247762322426
        },
        "Polygon": [
          {
            "X": 0.29548385739326477,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346210837364197,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346210837364197,
            "Y": 0.0965573713183403
          },
          {
            "X": 0.29548385739326477,
            "Y": 0.0965573713183403
          }
        ]
      },
      "Id": "26085884-d005-4144-b4c2-4d83dc50739b",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "ed48dacc-d089-498f-8e93-1cee1e5f39f3",
            "ac7370f3-cbb7-4cd9-a8f9-bdcb2252caaf"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.91246795654297,
      "Text": "Application Information",
        },
        "Polygon": [
          {
            "X": 0.8387037515640259,
            "Y": 0.8843405842781067
          },
          {
            "X": 0.9666182994842529,
            "Y": 0.8843405842781067
          },
          {
            "X": 0.9666182994842529,
            "Y": 0.9320254921913147
          },
          {
            "X": 0.8387037515640259,
            "Y": 0.9320254921913147
          }
        ]
      },
      "Id": "549ef3f9-3a13-4b77-bc25-fb2e0996839a"
    }
  ],
  "DetectDocumentTextModelVersion": "1.0",
  "ResponseMetadata": {
    "RequestId": "337129e6-3af7-4014-842b-f6484e82cbf6",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "337129e6-3af7-4014-842b-f6484e82cbf6",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "45675",
      "date": "Mon, 09 Nov 2020 23:54:38 GMT"
    },
    "RetryAttempts": 0
  }
}
}
```

Detecting Document Text with Amazon Textract

To detect text in a document, you use the DetectDocumentText operation, and pass a document file as input. `DetectDocumentText` returns a JSON structure that contains lines and words of detected text, the location of the text in the document, and the relationships between detected text. For more information, see Detecting Text.

You can provide an input document as an image byte array (base64-encoded image bytes), or as an Amazon S3 object. In this procedure, you upload an image file to your S3 bucket and specify the file name.

To detect text in a document (API)

1. If you haven't already:

   a. Give a user the `AmazonTextractFullAccess` and `AmazonS3ReadOnlyAccess` permissions. For more information, see Step 1: Set Up an AWS Account and Create a User.

   b. Install and configure the AWS CLI and the AWS SDKs. For more information, see Step 2: Set Up the AWS CLI and AWS SDKs.

2. Upload a document to your S3 bucket.

   For instructions, see Uploading Objects into Amazon S3 in the *Amazon Simple Storage Service User Guide*.

3. Use the following examples to call the `DetectDocumentText` operation.

   Java

   The following example code displays the document and boxes around lines of detected text.

   In the function `main`, replace the values of `bucket` and `document` with the names of the Amazon S3 bucket and document that you used in step 2. Replace the value of `credentialsProvider` with the name of your developer profile.

```
//Calls DetectDocumentText.
//Loads document from S3 bucket. Displays the document and bounding boxes around
 detected lines/words of text.
import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.BoundingBox;
import com.amazonaws.services.textract.model.DetectDocumentTextRequest;
import com.amazonaws.services.textract.model.DetectDocumentTextResult;
import com.amazonaws.services.textract.model.Document;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.Point;
import com.amazonaws.services.textract.model.Relationship;

public class DocumentText extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    DetectDocumentTextResult result;

    public DocumentText(DetectDocumentTextResult documentResult, BufferedImage
 bufImage) throws Exception {
        super();

        result = documentResult; // Results of text detection.
        image = bufImage; // The image containing the document.

    }

    // Draws the image and text bounding box.
    public void paintComponent(Graphics g) {

        int height = image.getHeight(this);
        int width = image.getWidth(this);

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, image.getWidth(this) , image.getHeight(this),
 this);

        // Iterate through blocks and display polygons around lines of detected
 text.
```

```
        List<Block> blocks = result.getBlocks();
        for (Block block : blocks) {
            DisplayBlockInfo(block);
            if ((block.getBlockType()).equals("LINE")) {
                ShowPolygon(height, width, block.getGeometry().getPolygon(),
g2d);
                /*
                  ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d);
                 */
            } else { // its a word, so just show vertical lines.
                ShowPolygonVerticals(height, width,
block.getGeometry().getPolygon(), g2d);
            }
        }
    }

    // Show bounding box at supplied location.
    private void ShowBoundingBox(int imageHeight, int imageWidth, BoundingBox
box, Graphics2D g2d) {

        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Display bounding box.
        g2d.setColor(new Color(0, 212, 0));
        g2d.drawRect(Math.round(left), Math.round(top),
                Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

    }

    // Shows polygon at supplied location
    private void ShowPolygon(int imageHeight, int imageWidth, List<Point>
points, Graphics2D g2d) {

        g2d.setColor(new Color(0, 0, 0));
        Polygon polygon = new Polygon();

        // Construct polygon and display
        for (Point point : points) {
            polygon.addPoint((Math.round(point.getX() * imageWidth)),
                    Math.round(point.getY() * imageHeight));
        }
        g2d.drawPolygon(polygon);
    }

    // Draws only the vertical lines in the supplied polygon.
    private void ShowPolygonVerticals(int imageHeight, int imageWidth,
 List<Point> points, Graphics2D g2d) {

        g2d.setColor(new Color(0, 212, 0));
        Object[] parry = points.toArray();
        g2d.setStroke(new BasicStroke(2));

        g2d.drawLine(Math.round(((Point) parry[0]).getX() * imageWidth),
                Math.round(((Point) parry[0]).getY() * imageHeight),
 Math.round(((Point) parry[3]).getX() * imageWidth),
                Math.round(((Point) parry[3]).getY() * imageHeight));

        g2d.setColor(new Color(255, 0, 0));
        g2d.drawLine(Math.round(((Point) parry[1]).getX() * imageWidth),
                Math.round(((Point) parry[1]).getY() * imageHeight),
 Math.round(((Point) parry[2]).getX() * imageWidth),
                Math.round(((Point) parry[2]).getY() * imageHeight));

    }
    //Displays information from a block returned by text detection and text
 analysis
    private void DisplayBlockInfo(Block block) {
        System.out.println("Block Id : " + block.getId());
        if (block.getText()!=null)
            System.out.println("    Detected text: " + block.getText());
        System.out.println("    Type: " + block.getBlockType());

        if (block.getBlockType().equals("PAGE") !=true) {
            System.out.println("    Confidence: " +
 block.getConfidence().toString());
        }
        if(block.getBlockType().equals("CELL"))
        {
            System.out.println("    Cell information:");
            System.out.println("        Column: " + block.getColumnIndex());
            System.out.println("        Row: " + block.getRowIndex());
            System.out.println("        Column span: " + block.getColumnSpan());
            System.out.println("        Row span: " + block.getRowSpan());

        }
        System.out.println("    Relationships");
        List<Relationship> relationships=block.getRelationships();
        if(relationships!=null) {
            for (Relationship relationship : relationships) {
                System.out.println("        Type: " + relationship.getType());
                System.out.println("        IDs: " +
relationship.getIds().toString());
            }
        } else {
            System.out.println("        No related Blocks");
        }

        System.out.println("    Geometry");
        System.out.println("        Bounding Box: " +
block.getGeometry().getBoundingBox().toString());
        System.out.println("        Polygon: " +
block.getGeometry().getPolygon().toString());

        List<String> entityTypes = block.getEntityTypes();

        System.out.println("    Entity Types");
        if(entityTypes!=null) {
            for (String entityType : entityTypes) {
                System.out.println("        Entity Type: " + entityType);
            }
        } else {
            System.out.println("        No entity type");
        }
        if(block.getPage()!=null)
            System.out.println("    Page: " + block.getPage());
        System.out.println();
    }

    public static void main(String arg[]) throws Exception {

        // The S3 bucket and document
        String document = "";
        String bucket = "";

        // set provider credentials
        AWSCredentialsProvider credentialsProvider = new
ProfileCredentialsProvider("default");
```

```
        AmazonS3 s3client =
 AmazonS3ClientBuilder.standard().withCredentials(credentialsProvider)
                .withEndpointConfiguration(
                        new EndpointConfiguration("https://
s3.amazonaws.com","us-east-1"))
                .build();


        // Get the document from S3
        com.amazonaws.services.s3.model.S3Object s3object =
 s3client.getObject(bucket, document);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        BufferedImage image = ImageIO.read(inputStream);

        // Call DetectDocumentText
        EndpointConfiguration endpoint = new EndpointConfiguration(
                "https://textract.us-east-1.amazonaws.com", "us-east-1");
        AmazonTextract client =
 AmazonTextractClientBuilder.standard().withCredentials(credentialsProvider)
                .withEndpointConfiguration(endpoint).build();


        DetectDocumentTextRequest request = new DetectDocumentTextRequest()
            .withDocument(new Document().withS3Object(new
 S3Object().withName(document).withBucket(bucket)));

        DetectDocumentTextResult result = client.detectDocumentText(request);

        // Create frame and panel.
        JFrame frame = new JFrame("RotateImage");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        DocumentText panel = new DocumentText(result, image);
        panel.setPreferredSize(new Dimension(image.getWidth() ,
 image.getHeight() ));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    }
}
```

Java V2

The following example code displays the document and boxes around lines of detected text.

In the function `main`, replace the values of `bucket` and `document` with the names of the Amazon S3 bucket and document that you used in step 2. Replace `profile-name` in the line that creates the `TextractClient` with the name of your developer profile.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import
 software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import
 software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.util.Iterator;
import java.util.List;
//snippet-end:[textract.java2._detect_s3_text.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class DetectText {

 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage:\n" +
         "    <bucketName> <docName> \n\n" +
         "Where:\n" +
         "    bucketName - The name of the Amazon S3 bucket that contains the
 document. \n\n" +
         "    docName - The document name (must be an image, i.e., book.png).
 \n";

     if (args.length != 2) {
         System.out.println(usage);
         System.exit(1);
     }

     String bucketName = args[0];
     String docName = args[1];
     Region region = Region.US_EAST_1;
     TextractClient textractClient = TextractClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
         .build();

     detectDocTextS3(textractClient, bucketName, docName);
     textractClient.close();
 }

 // snippet-start:[textract.java2._detect_s3_text.main]
 public static void detectDocTextS3 (TextractClient textractClient, String
 bucketName, String docName) {

     try {
         S3Object s3Object = S3Object.builder()
             .bucket(bucketName)
             .name(docName)
             .build();

         // Create a Document object and reference the s3Object instance
         Document myDoc = Document.builder()
             .s3Object(s3Object)
             .build();

         DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
             .document(myDoc)
             .build();
         DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
         for (Block block: textResponse.blocks()) {
             System.out.println("The block type is "
 +block.blockType().toString());
         }

         DocumentMetadata documentMetadata = textResponse.documentMetadata();
         System.out.println("The number of pages in the document is "
 +documentMetadata.pages());

     } catch (TextractException e) {

         System.err.println(e.getMessage());
         System.exit(1);
     }
 }
 // snippet-end:[textract.java2._detect_s3_text.main]
}
```

AWS CLI

This AWS CLI command displays the JSON output for the `detect-document-text` CLI operation.

Replace the values of `Bucket` and `Name` with the names of the Amazon S3 bucket and document that you used in step 2. Replace `profile-name` with the name of a profile that can assume the role and `region` with the region in which you want to run the code.

```
aws textract detect-document-text \
    --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
    --profile profile-name \
    --region region
```

Python

The following example code displays the document and boxes around detected lines of text.

In the function `main`, replace the values of `bucket` and `document` with the names of the Amazon S3 bucket and document that you used in step 2. Replace `profile-name` with the name of a profile that can assume the role and `region` with the region in which you want to run the code.

```
#Detects text in a document stored in an S3 bucket. Display polygon box around
 text and angled text
import boto3
import io
from PIL import Image, ImageDraw

def process_text_detection(s3_connection, client, bucket, document):

    #Get the document from S3
    s3_object = s3_connection.Object(bucket,document)
    s3_response = s3_object.get()

    stream = io.BytesIO(s3_response['Body'].read())
    image=Image.open(stream)

    #To process using image bytes:
    #image_binary = stream.getvalue()
    #response = client.detect_document_text(Document={'Bytes': image_binary})

    # Detect text in the document
    # Process using S3 object
    response = client.detect_document_text(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    # Get the text blocks
    blocks=response['Blocks']
    width, height =image.size
    print ('Detected Document Text')

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:
            # Display information about a block returned by text detection
            print('Type: ' + block['BlockType'])
            if block['BlockType'] != 'PAGE':
                print('Detected: ' + block['Text'])
                print('Confidence: ' + "{:.2f}".format(block['Confidence']) +
 "%")

            print('Id: {}'.format(block['Id']))
            if 'Relationships' in block:
                print('Relationships: {}'.format(block['Relationships']))
            print('Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
            print('Polygon: {}'.format(block['Geometry']['Polygon']))
            print()
            draw=ImageDraw.Draw(image)
            # Draw WORD - Green -  start of word, red - end of word
            if block['BlockType'] == "WORD":
                draw.line([(width * block['Geometry']['Polygon'][0]['X'],
                height * block['Geometry']['Polygon'][0]['Y']),
                (width * block['Geometry']['Polygon'][3]['X'],
                height * block['Geometry']['Polygon'][3]['Y'])],fill='green',
                width=2)

                draw.line([(width * block['Geometry']['Polygon'][1]['X'],
                height * block['Geometry']['Polygon'][1]['Y']),
                (width * block['Geometry']['Polygon'][2]['X'],
                height * block['Geometry']['Polygon'][2]['Y'])],
                fill='red',
                width=2)


            # Draw box around entire LINE
            if block['BlockType'] == "LINE":
                points=[]

                for polygon in block['Geometry']['Polygon']:
                    points.append((width * polygon['X'], height * polygon['Y']))

                draw.polygon((points), outline='black')

    # Display the image
    image.show()

    return len(blocks)

def main():
    session = boto3.Session(profile_name='profile-name')
    s3_connection = session.resource('s3')
    client = session.client('textract', region_name='region')
    bucket = ''
    document = ''
    block_count=process_text_detection(s3_connection,client,bucket,document)
    print("Blocks detected: " + str(block_count))
```

```
if __name__ == "__main__":
    main()
```

Node.js

The following Node.js example code displays the document and boxes around detected lines of text. It outputs an image of the results to the directory you run the code from. It makes use of the `image-size` and `images` packages.

In the function `main`, replace the values of `bucket` and `document` with the names of the Amazon S3 bucket and document that you used in step 2. Replace the value of `regionConfig` with the name of the region your account is in. Replace the value of `credentials`with the name of your developer profile.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

async function main(){
// Import AWS
const AWS = require("aws-sdk")
// Use Image-Size to get
const sizeOf = require('image-size');
// Image tool to draw buffers
const images = require("images");

// Set variables
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;
AWS.config.update({region:'region-name'});
const bucket = 'bucket-name' // the s3 bucket name
const photo  = 'photo-name' // the name of file

// Create a canvas and get the context
const { createCanvas } = require('canvas')
const canvas = createCanvas(200, 200)
const ctx = canvas.getContext('2d')

// Connect to Textract
const client = new AWS.Textract();
// Connect to S3 to display image
```

```
const s3 = new AWS.S3();

// Define paramaters
const params = {
  Document: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
}

// Function to display image
async function getImage(){
  const imageData =  s3.getObject(
    {
        Bucket: bucket,
        Key: photo
      }

  ).promise();
  return imageData;
}

// get image
var imageData = await getImage()

// Get the height, width of the image
const dimensions = sizeOf(imageData.Body)
const width = dimensions.width
const height = dimensions.height
console.log(imageData.Body)
console.log(width, height)

canvas.width = width;
canvas.height = height;

try{
  // Call API and log response
  const res = await client.detectDocumentText(params).promise();
  var image = images(imageData.Body).size(width, height)
  //console.log the type of block, text, text type, and confidence
  res.Blocks.forEach(block => {
    console.log(`Block Type: ${block.BlockType}`),
    console.log(`Text: ${block.Text}`)
    console.log(`TextType: ${block.TextType}`)
    console.log(`Confidence: ${block.Confidence}`)

    // Draw box around detected text using polygons
    ctx.strokeStyle = 'rgba(0,0,0,0.5)';
    ctx.beginPath();
    block.Geometry.Polygon.forEach(({X, Y}) =>
    ctx.lineTo(width * X - 10, height * Y - 10)
    );
    ctx.closePath();
    ctx.stroke();
    console.log("-----")
  })

  // render image
  var buffer = canvas.toBuffer("image/png");
  image.draw(images(buffer), 10, 10)
  image.save("output-image.jpg");

} catch (err){
console.error(err);}

}

main()
```

.NET

The following example provides detected text as a list. Replace the values of `bucket` and `document` with the names of the Amazon S3 bucket and document image that you used in step 2.

```

using System;
using System.Linq;
using Amazon.Textract;
using Amazon.Textract.Model;

namespace TextractAnalyzeID
{
    class Program
    {
        static async Task Main()
        {
            String document = "document";
            String bucket = "bucket";

            AmazonTextractClient textractClient = new AmazonTextractClient();

            DetectDocumentTextRequest detectDocumentTextRequest = new
 DetectDocumentTextRequest()
            {
                Document = new Document()
                {
                    S3Object = new S3Object()
                    {
                        Name = document,
                        Bucket = bucket
                    }
                }
            };

            try
            {
                var DocumentText = await
 textractClient.DetectDocumentTextAsync(detectDocumentTextRequest);
                foreach (Block block in DocumentText.Blocks)
                {
                    Console.WriteLine(block.BlockType);
                    if (block.BlockType != "PAGE") {
                        Console.WriteLine("Detected Text= " + block.Text);
                        Console.WriteLine("Confidence= " + block.Confidence);
                    }
                    Console.WriteLine("Id= " + block.Id);

                    foreach(Relationship relationship in block.Relationships) {
                        Console.WriteLine(relationship.Type);
                        relationship.Ids.ForEach(id => Console.WriteLine("Id= "
 + id));
                    }

                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

4. Run the example. The Python and Java examples display the document image. A black box surrounds each line of detected text. A green vertical line is the start of a detected word. A red vertical line is the end of a detected word. The AWS CLI example displays only the JSON output for the `DetectDocumentText` operation.

Analyzing Document Text with Amazon Textract

To analyze text in a document, you use the AnalyzeDocument operation, and pass a document file as input. `AnalyzeDocument` returns a JSON structure that contains the analyzed text. For more information, see Analyzing Documents.

You can provide an input document as an image byte array (base64-encoded image bytes), or as an Amazon S3 object. In this procedure, you upload an image file to your S3 bucket and specify the file name.

To analyze text in a document (API)

1. If you haven't already:

   a. Give a user the `AmazonTextractFullAccess` and `AmazonS3ReadOnlyAccess` permissions. For more information, see Step 1: Set Up an AWS Account and Create a User.

   b. Install and configure the AWS CLI and the AWS SDKs. For more information, see Step 2: Set Up the AWS CLI and AWS SDKs.

2. Upload an image that contains a document to your S3 bucket.

   For instructions, see Uploading Objects into Amazon S3 in the *Amazon Simple Storage Service User Guide*.

3. Use the following examples to call the `AnalyzeDocument` operation.

   Java

   The following example code displays the document and boxes around detected items.

In the function `main`, replace the values of `bucket` and `document` with the names of the Amazon S3 bucket and document image that you used in step 2. Replace the value of `credentialsProvider` with the name of your developer profile.

```
//Loads document from S3 bucket. Displays the document and polygon around
 detected lines of text.
import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.AnalyzeDocumentRequest;
import com.amazonaws.services.textract.model.AnalyzeDocumentResult;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.BoundingBox;
import com.amazonaws.services.textract.model.Document;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.Point;
import com.amazonaws.services.textract.model.Relationship;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;

public class AnalyzeDocument extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;

    AnalyzeDocumentResult result;

    public AnalyzeDocument(AnalyzeDocumentResult documentResult, BufferedImage
 bufImage) throws Exception {
        super();

        result = documentResult; // Results of text detection.
        image = bufImage; // The image containing the document.
    }

    // Draws the image and text bounding box.
    public void paintComponent(Graphics g) {

        int height = image.getHeight(this);
        int width = image.getWidth(this);

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, image.getWidth(this), image.getHeight(this),
 this);

        // Iterate through blocks and display bounding boxes around everything.

        List<Block> blocks = result.getBlocks();
        for (Block block : blocks) {
            DisplayBlockInfo(block);
            switch(block.getBlockType()) {

            case "KEY_VALUE_SET":
                if (block.getEntityTypes().contains("KEY")){
                    ShowBoundingBox(height, width,
 block.getGeometry().getBoundingBox(), g2d, new Color(255,0,0));
                }
                else {  //VALUE
                    ShowBoundingBox(height, width,
 block.getGeometry().getBoundingBox(), g2d, new Color(0,255,0));
                }
                break;
            case "TABLE":
                ShowBoundingBox(height, width,
 block.getGeometry().getBoundingBox(), g2d, new Color(0,0,255));
                break;
            case "CELL":
                ShowBoundingBox(height, width,
 block.getGeometry().getBoundingBox(), g2d, new Color(255,255,0));
                break;
            case "SELECTION_ELEMENT":
                if (block.getSelectionStatus().equals("SELECTED"))
                    ShowSelectedElement(height, width,
 block.getGeometry().getBoundingBox(), g2d, new Color(0,0,255));
```

```
                break;
            default:
                //PAGE, LINE & WORD
                //ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(200,200,0));
            }
        }

        // uncomment to show polygon around all blocks
        //ShowPolygon(height,width,block.getGeometry().getPolygon(),g2d);


    }

    // Show bounding box at supplied location.
    private void ShowBoundingBox(int imageHeight, int imageWidth, BoundingBox
box, Graphics2D g2d, Color color) {

        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Display bounding box.
        g2d.setColor(color);
        g2d.drawRect(Math.round(left), Math.round(top),
                Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

    }
    private void ShowSelectedElement(int imageHeight, int imageWidth,
BoundingBox box, Graphics2D g2d, Color color) {

        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Display bounding box.
        g2d.setColor(color);
        g2d.fillRect(Math.round(left), Math.round(top),
                Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

    }

    // Shows polygon at supplied location
```

```
   private void ShowPolygon(int imageHeight, int imageWidth, List<Point>
points, Graphics2D g2d) {

       g2d.setColor(new Color(0, 0, 0));
       Polygon polygon = new Polygon();

       // Construct polygon and display
       for (Point point : points) {
           polygon.addPoint((Math.round(point.getX() * imageWidth)),
                   Math.round(point.getY() * imageHeight));
       }
       g2d.drawPolygon(polygon);
   }
   //Displays information from a block returned by text detection and text
analysis
   private void DisplayBlockInfo(Block block) {
       System.out.println("Block Id : " + block.getId());
       if (block.getText()!=null)
           System.out.println("    Detected text: " + block.getText());
       System.out.println("    Type: " + block.getBlockType());

       if (block.getBlockType().equals("PAGE") !=true) {
           System.out.println("    Confidence: " +
block.getConfidence().toString());
       }
       if(block.getBlockType().equals("CELL"))
       {
           System.out.println("    Cell information:");
           System.out.println("        Column: " + block.getColumnIndex());
           System.out.println("        Row: " + block.getRowIndex());
           System.out.println("        Column span: " + block.getColumnSpan());
           System.out.println("        Row span: " + block.getRowSpan());

       }

       System.out.println("    Relationships");
       List<Relationship> relationships=block.getRelationships();
       if(relationships!=null) {
           for (Relationship relationship : relationships) {
               System.out.println("        Type: " + relationship.getType());
               System.out.println("        IDs: " +
relationship.getIds().toString());
           }
       } else {
```

```
                System.out.println("        No related Blocks");
            }

            System.out.println("    Geometry");
            System.out.println("        Bounding Box: " +
    block.getGeometry().getBoundingBox().toString());
            System.out.println("        Polygon: " +
    block.getGeometry().getPolygon().toString());

            List<String> entityTypes = block.getEntityTypes();

            System.out.println("    Entity Types");
            if(entityTypes!=null) {
                for (String entityType : entityTypes) {
                    System.out.println("        Entity Type: " + entityType);
                }
            } else {
                System.out.println("        No entity type");
            }

            if(block.getBlockType().equals("SELECTION_ELEMENT")) {
                System.out.print("    Selection element detected: ");
                if (block.getSelectionStatus().equals("SELECTED")){
                    System.out.println("Selected");
                }else {
                    System.out.println(" Not selected");
                }
            }

            if(block.getPage()!=null)
                System.out.println("    Page: " + block.getPage());
            System.out.println();
        }

        public static void main(String arg[]) throws Exception {

            // The S3 bucket and document
            String document = "";
            String bucket = "";

            // set provider credentials
            AWSCredentialsProvider credentialsProvider = new
    ProfileCredentialsProvider("default");
```

```
        AmazonS3 s3client =
 AmazonS3ClientBuilder.standard().withCredentials(credentialsProvider)
                .withEndpointConfiguration(
                        new EndpointConfiguration("https://
s3.amazonaws.com","us-east-1"))
                .build();


        // Get the document from S3
        com.amazonaws.services.s3.model.S3Object s3object =
 s3client.getObject(bucket, document);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        BufferedImage image = ImageIO.read(inputStream);

        // Call AnalyzeDocument
        EndpointConfiguration endpoint = new EndpointConfiguration(
                "https://textract.us-east-1.amazonaws.com", "us-east-1");
        AmazonTextract client =
 AmazonTextractClientBuilder.standard().withCredentials(credentialsProvider)
                .withEndpointConfiguration(endpoint).build();


        AnalyzeDocumentRequest request = new AnalyzeDocumentRequest()
                .withFeatureTypes("TABLES","FORMS","SIGNATURES")
                 .withDocument(new Document().
                        withS3Object(new
 S3Object().withName(document).withBucket(bucket)));


        AnalyzeDocumentResult result = client.analyzeDocument(request);

        // Create frame and panel.
        JFrame frame = new JFrame("RotateImage");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        AnalyzeDocument panel = new AnalyzeDocument(result, image);
        panel.setPreferredSize(new Dimension(image.getWidth(),
 image.getHeight()));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    }
}
```

Java V2

The following example code displays the document and boxes around lines of detected text.

In the function `main`, replace the values of `bucket` and `document` with the names of the Amazon S3 bucket and document that you used in step 2. Replace `profile-name` in the line that creates the `TextractClient` with the name of your developer profile.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
// snippet-end:[textract.java2._analyze_doc.import]

/**
 * Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AnalyzeDocument {

    public static void main(String[] args) {
```