aws

开发人员指南

Amazon Textract

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Textract: 开发人员指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其它商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

什么是 Amazon Textract？ 1
新 Amazon Textract 2
将 Amazon Textract 与AWS开发工具包 2
工作方式 4
检测文本 4
分析文档 5
分析发票和收据 7
分析身份证件 10
输入文档 11
Amazon Textract 响应对象 12
文本检测和文档分析响应对象 12
发票和收据回复对象 31
身份文档响应对象 34
文档页面上的物品位置 36
Bounding Box 37
面 39
入门 41
第 1 步：设置账户 41
注册 AWS 41
创建 IAM 用户 42
下一个步骤 43
第 2 步：设置AWS CLI和AWS软件开发工具包 43
下一个步骤 44
第 3 步：开始使用AWS CLI和AWSSDK API 45
设置 AWS CLI 示例的格式 45
使用同步操作处理文档 46
调用 Amazon Textract 同步操作 46
请求 46
响应 48
检测文本 120
分析文档文本 132
分析发票和收据单据 144
分析身份证文档 157
使用异步操作处理文档 162

调用异步操作 162
开始文本检测 163
获取 Amazon Textract 分析请求的完成状态 165
获取 Amazon Textract 文本检测结果 166
配置异步操作 176
授予 Amazon Textract 访问您的 Amazon SNS 主题 177
检测或分析多页文档中的文本 178
执行异步操作 179
Amazon Textract 结果通知 204
处理受限的呼叫和断开的连接 206
Amazon Textract 的最佳实践 211
提供最佳输入文档 211
使用置信度得分 211
考虑使用人工审核 212
教程 213
先决条件 213
从表单文档中提取键值对 213
将表导出到 CSV 文件 216
创建AWS Lambda函数 226
要从 Lambda 函数调用 DetectDocumentText 操作，请执行以下操作： 226
其他代码示例 229
代码示例 231
操作 231
分析文档 232
检测文档中的文本 234
获取有关文档分析作业的数据 238
开始对文档进行异步分析 239
开始异步文本检测 243
跨服务示例 244
创建 Amazon Textract 浏览器应用程序 245
检测从图像中提取的文本中的实体 246
Amazon A2I 和 248
Amazon A2I 的核心概念 248
人工审核激活条件 248
人工审查工作流程（流程定义） 249
人类循环 250

使用 Amazon A2I 251
创建人工审核工作流程 252
分析文档 257
监控人工循环 258
查看输出数据和工作人员指标 259
安全性 262
数据保护 262
Amazon Textract 中的加密 263
互联网络流量保密性 264
Identity and Access Management 264
Audience 264
使用身份进行身份验证 265
使用策略管理访问 267
Amazon Textract 如何与 IAM 配合使用 269
基于身份的策略示例 272
故障排除 275
日志记录和监控 277
监控 278
Amazon Textract 的 CloudWatch 指标 281
使用 记录 Amazon Textract API 调用AWS CloudTrail 283
CloudTrail 中的 Amazon Textract 信息 283
了解 Amazon Textract 日志文件条目 285
合规性验证 287
故障恢复能力 287
基础设施安全性 288
配置和漏洞分析 288
VPC 终端节点 (AWS PrivateLink) 288
Amazon Textract VPC 终端节点的注意事项 289
为 Amazon Textract 创建接口 VPC 终端节点 289
为 Amazon Textract 创建 VPC 终端节点策略 289
API 引用 291
操作 291
AnalyzeDocument 292
AnalyzeExpense 298
AnalyzeID 304
DetectDocumentText 308

GetDocumentAnalysis 313
GetDocumentTextDetection 319
GetExpenseAnalysis 325
StartDocumentAnalysis 333
StartDocumentTextDetection 339
StartExpenseAnalysis 345
数据类型 350
AnalyzeIDDetections 352
Block 353
BoundingBox 358
Document 360
DocumentLocation 362
DocumentMetadata 363
ExpenseDetection 364
ExpenseDocument 365
ExpenseField 366
ExpenseType 368
Geometry 369
HumanLoopActivationOutput 370
HumanLoopConfig 372
HumanLoopDataAttributes 374
IdentityDocument 375
IdentityDocumentField 376
LineItemFields 377
LineItemGroup 378
NormalizedValue 379
NotificationChannel 380
OutputConfig 381
Point 383
Relationship 384
S3Object 385
Warning 387
限制 388
Amazon Textract 388
文档历史记录 390
AWS词汇表 392

..... cccxciii

什么是 Amazon Textract？

Amazon Textract 让您可以向应用程序轻松添加文档文本检测和分析功能。使用 Amazon Textract 买家可以：

- 检测各种文档中的打字和手写文本，包括财务报告、医疗记录和税务表格。
- 使用 Amazon Textract 文档分析 API 从包含结构化数据的文档中提取文本、表单和表格。
- 使用分析费用 API 处理发票和收据。
- 使用 AnalyZeID API 处理美国政府颁发的驾驶执照和护照等身份证件。

Amazon Textract 基于同样由 Amazon 计算机视觉科学家开发的成熟且高度可扩展的深度学习技术，每天能够分析数十亿图像和视频。使用无需任何机器学习方面的专业技能。Amazon Textract 包括简单易用的 API，可以分析图像文件和 PDF 文件。Amazon Textract 始终从新数据进行学习，Amazon 会不断向此服务添加新功能。

以下是使用 Amazon Textract 的常见使用案例：

- 创建智能搜索索引— 使用 Amazon Textract，您可以创建图像和 PDF 文件中检测到的文本库。
- 使用智能文本提取功能进行自然语言处理 (NLP)— Amazon Textract 让您可以控制如何将文本分组为 NLP 应用程序的输入。它可以将文本提取为单词和行。如果启用了 Amazon Textract 文档表分析，它还会按表格单元格对文本进行分组。
- 加快来自不同来源的数据的捕获和标准化— Amazon Textract 支持从各种文档中提取文本和表格数据，例如财务文档、研究报告和医疗笔记。借助 Amazon Textract 分析文档 API，您可以轻松快速地从文档中提取非结构化和结构化数据。
- 自动从表单中捕获数据— Amazon Textract 允许从表单中提取结构化数据。借助 Amazon Textract Analysis API，您可以在现有业务工作流程中构建提取功能，以便通过表单提交的用户数据可以提取为可用的格式。

使用 Amazon Textract 的一些好处包括：

- 将文档文本检测集成到应用中— Amazon Textract 通过使用简单 API 提供强大而准确的分析，来消除在应用程序中内置文本检测功能的复杂性。无需计算机视觉或深度学习方面的专业技能，即可使用 Amazon Textract 来检测文档文本。利用 Amazon Textract 文本 API，您可以轻松地将文本检测功能内置到任何 Web、移动或互联设备应用程序中。

- 可扩展文档分析— Amazon Textract 使您能够快速分析和从数百万个文档中提取数据，从而加快决策过程。
- 低成本-使用 Amazon Textract，您只需为分析文档付费。没有最低费用或预付费用。利用的分级定价模式，您可以免费开始使用并在您发展业务时节省更多成本。

借助同步处理，Amazon Textract 可以分析延迟至关重要的应用程序的单页文档。Amazon Textract 还提供异步操作以将支持扩展到多页文档。

新 Amazon Textract

如果这是您首次使用 Amazon Textract，建议您按顺序阅读以下内容：

1. Amazon Textract 的工作原理-本节介绍 Amazon Textract 组件以及它们如何协同工作以提供端到端体验。
2. Amazon Textract 入门— 在本部分中，您将设置账户并测试 Amazon Textract API。

将 Amazon Textract 与AWS开发工具包

AWS 软件开发工具包 (SDK) 可用于很多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

软件开发工具包文档	代码示例
适用于 C++ 的 AWS SDK	适用于 C++ 的 AWS SDK代码示例
适用于 Go 的 AWS SDK	适用于 Go 的 AWS SDK代码示例
适用于 Java 的 AWS SDK	适用于 Java 的 AWS SDK代码示例
适用于 JavaScript 的 AWS SDK	适用于 JavaScript 的 AWS SDK代码示例
适用于 .NET 的 AWS SDK	适用于 .NET 的 AWS SDK代码示例
适用于 PHP 的 AWS SDK	适用于 PHP 的 AWS SDK代码示例
适用于 Python (Boto3) 的 AWS SDK	适用于 Python (Boto3) 的 AWS SDK代码示例
适用于 Ruby 的 AWS SDK	适用于 Ruby 的 AWS SDK代码示例

可用性示例

找不到所需的内容？ 通过使用此页面底部的 Provide feedback（提供反馈）链接请求代码示例。

Amazon Textract 的工作原理

借助 Amazon Textract，您可以检测和分析单页或多页输入文档中的文本（请参阅输入文档)。

Amazon Textract 为以下操作提供操作。

- 只能检测文本。有关更多信息，请参阅检测文本.
- 检测和分析文本之间的关系。有关更多信息，请参阅分析文档.
- 检测和分析发票和收据中的文本。有关更多信息，请参阅分析发票和收据.
- 检测和分析政府身份证件中的文本。有关更多信息，请参阅分析身份证件.

Amazon Textract 提供同步操作，用于处理小型、单页的文档以及接近实时的响应。有关更多信息，请参阅 使用同步操作处理文档。Amazon Textract 还提供异步操作，您可以使用这些操作来处理更大的多页文档。异步响应不是实时的。有关更多信息，请参阅 使用异步操作处理文档。

当 Amazon Textract 操作处理文档时，结果将返回在the section called “Block”对象或数组the section called “ExpenseDocument”对象。两个对象都包含检测到的有关项目的信息，包括它们在文档中的位置以及它们与文档中其他项目的关系。有关更多信息，请参阅 Amazon Textract 响应对象。有关演示如何使用的示例`Block`对象，请参阅教程.

主题

- 检测文本
- 分析文档
- 分析发票和收据
- 分析身份证件
- 输入文档
- Amazon Textract 响应对象
- 文档页面上的物品位置

检测文本

Amazon Textract 提供同步和异步操作，这些操作仅返回文档中检测到的文本。对于这两组操作，以下信息将以多个方式返回the section called “Block”对象。

- 检测到的文本的行和单词

- 检测到的文本的行和单词之间的关系
- 检测到的文本显示在其上的页面
- 文档页面上行和文本单词的位置

有关更多信息，请参阅 the section called “文本的行和单词”。

要同步检测文本，请使用DetectDocumentTextAPI 操作，并将文档文件作为输入传递。操作返回整组结果。有关更多信息以及示例，请参阅 使用同步操作处理文档。

> **Note**
>
> Amazon Rekognition API 操作`DetectText`不同于`DetectDocumentText`. 你使用`DetectText`以检测实时场景中的文字，例如海报或路标。

要异步检测文本，请使用StartDocumentTextDetection开始处理输入文档文件。要获得结果，请致电GetDocumentTextDetection. 结果将在来自的一个或多个回复中返回`GetDocumentTextDetection`. 有关更多信息以及示例，请参阅 使用异步操作处理文档。

分析文档

Amazon Textract 分析文档和表单以了解检测到的文本之间的关系 Amazon Textract 分析操作会返回 3 种文档提取类别 — 文本、表单和表格。对发票和收据的分析是通过不同的流程处理的，有关详细信息，请参阅分析发票和收据.

提取文本

从文档中提取的原始文本。有关更多信息，请参阅 。文本的行和单词.

提取表单

表单数据链接到从文档中提取的文本项目。Amazon Textract 将表单数据表示为键/值对。在以下示例中，Amazon Textract 检测到的文本行之一是名称：Jane Doe. Amazon Textract 还可以识别密钥（名称：）和一个值 (Jane Doe)。有关更多信息，请参阅 。表单数据（键值对）.

名称：Jane Doe

地址：123 任何街，美国 Anytown

出生日期：1980 年 12-26-

键值对还用于表示从表单中提取的复选框或选项按钮（单选按钮）。

男：☑

有关更多信息，请参阅 。选择元素.

提取表

Amazon Textract 可以提取表格、表格单元格和表格单元格中的项目，并可以编程以 JSON、.csv 或 .txt 文件返回结果。

名称	Address
Ana Carolina	123 Aany Stown

有关更多信息，请参阅表。也可以从表中提取选择元素。有关更多信息，请参阅 。选择元素.

对于已分析的商品，Amazon Textract 将以下商品返回多个the section called “Block”对象：

- 检测到的文本的行和单词
- 检测到的物品的内容
- 检测到的物品之间的关系
- 检测到商品的页面
- 项目在文档页面上的位置

您可以使用同步或异步操作来分析文档中的文本。要同步分析文本，请使用AnalyzeDocument操作，然后将文档作为输入传递。AnalyzeDocument返回整组结果。有关更多信息，请参阅 使用 Amazon Textract 分析文档文本。

要异步检测文本，请使用StartDocumentAnalysis开始处理。要获得结果，请致电GetDocumentAnalysis. 结果将在来自的一个或多个回复中返回GetDocumentAnalysis. 有关更多信息以及示例，请参阅 检测或分析多页文档中的文本。

要指定要执行哪种类型的分析，可以使用FeatureTypes列出输入参数。将 TABLES 添加到列表以返回有关输入文档中检测到的表的信息，例如，表格单元格、单元格文本和单元格中的选择元素。添加 FORMS 以返回单词关系，例如键值对和选择元素。要执行这两种类型的分析，请将 TABLES 和 FORMS 添加到FeatureTypes.

在文档中检测到的所有行和单词都包含在响应中（包括与FeatureTypes)。

分析发票和收据

Amazon Textract 从几乎任何发票或收据中提取相关数据，例如联系信息、购买的商品和供应商名称，而无需任何模板或配置。发票和收据通常使用各种布局，因此大规模手动提取数据变得困难而且耗时。Amazon Textract 使用 ML 了解发票和收据的上下文，并自动提取诸如发票或收据日期、发票或收据编号、商品价格、总金额和付款条件等数据，以满足您的业务需求。

Amazon Textract 还可以识别对您的工作流程至关重要但可能没有明确标记的供应商名称。例如，Amazon Textract 可以在收据上找到供应商名称，即使该名称仅在页面顶部的徽标中注明，没有明确的键值对组合。Amazon Textract 还可以让您轻松整合来自不同收据和发票的输入，这些收据和发票对同一概念使用不同词汇。例如，Amazon Textract 将不同文档（如客户编号、客户号码和账户 ID）中的字段名称之间的关系映射出来，将标准分类输出为`INVOICE_RECEIPT_ID`. 在这种情况下，Amazon Textract 一致地表示不同文档类型的数据。与标准分类不一致的字段被归类为`OTHER`.

以下是 AnalyzeExperet 目前支持的标准字段的列表：

- 供应商名称：`VENDOR_NAME`
- 总计：`TOTAL`
- 收件人地址：`RECEIVER_ADDRESS`
- 发票/收据日期：`INVOICE_RECEIPT_DATE`
- 发票/收据编号：`INVOICE_RECEIPT_ID`
- 付款条款：`PAYMENT_TERMS`
- 小计：`SUBTOTAL`
- 截止日期：`DUE_DATE`
- 税：`TAX`
- 发票纳税人 ID（SSN/ITIN 或 EIN）：`TAX_PAYER_ID`
- 项目名称：`ITEM_NAME`
- 商品价格：`PRICE`
- 商品数量：`QUANTITY`

分析费用 API 返回给定文档页面的以下元素：

- 页面中的收据或发票数量表示为`ExpenseIndex`
- 表示为的单个字段的标准化名称`Type`
- 显示在文档上的字段的实际名称，表示为`LabelDetection`

- 表示为的相应字段的值ValueDetection
- 提交的文档中的页数表示为Pages
- 检测到字段、值或行项目的页码，表示为PageNumber
- 几何图形，其中包括边界框和页面上各个字段、值或行项目的坐标位置，表示为Geometry
- 与文档中检测到的每条数据相关联的置信度评分，表示为Confidence
- 所购买的单个行商品的整行，表示为EXPENSE_ROW

以下是 AnalyzeExendal 处理的收据的 API 输出的一部分，其中显示了总额：作为标准字段提取的文档中 55.64 美元TOTAL，文档上的实际文本为 “总计”，置信度分数为 “97.1”，页码 “1”，总值为 “$55.64”，边界框和多边形坐标：


```
{
    "Type": {
        "Text": "TOTAL",
        "Confidence": 99.94717407226562
    },
    "LabelDetection": {
        "Text": "Total:",
        "Geometry": {
            "BoundingBox": {
                "Width": 0.09809663146734238,
                "Height": 0.0234375,
                "Left": 0.36822840571403503,
                "Top": 0.8017578125
            },
            "Polygon": [
                {
                    "X": 0.36822840571403503,
                    "Y": 0.8017578125
                },
                {
                    "X": 0.466325044631958,
                    "Y": 0.8017578125
                },
                {
                    "X": 0.466325044631958,
                    "Y": 0.8251953125
                },
                {
                    "X": 0.36822840571403503,
```

```
                    "Y": 0.8251953125
                }
            ]
        },
        "Confidence": 97.10792541503906
    },
        "ValueDetection": {
            "Text": "$55.64",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.10395314544439316,
                    "Height": 0.0244140625,
                    "Left": 0.66837477684021,
                    "Top": 0.802734375
                },
                "Polygon": [
                    {
                        "X": 0.66837477684021,
                        "Y": 0.802734375
                    },
                    {
                        "X": 0.7723279595375061,
                        "Y": 0.802734375
                    },
                    {
                        "X": 0.7723279595375061,
                        "Y": 0.8271484375
                    },
                    {
                        "X": 0.66837477684021,
                        "Y": 0.8271484375
                    }
                ]
            },
        "Confidence": 99.85165405273438
    },
    "PageNumber": 1
    }
```


您可以使用同步操作来分析发票或收据。要分析这些文档，您可以使用 AnalyzeExendal 操作并向其传递收据或发票。AnalyzeExpense返回整组结果。有关更多信息，请参阅 使用 Amazon Textract 分析发票和收据。

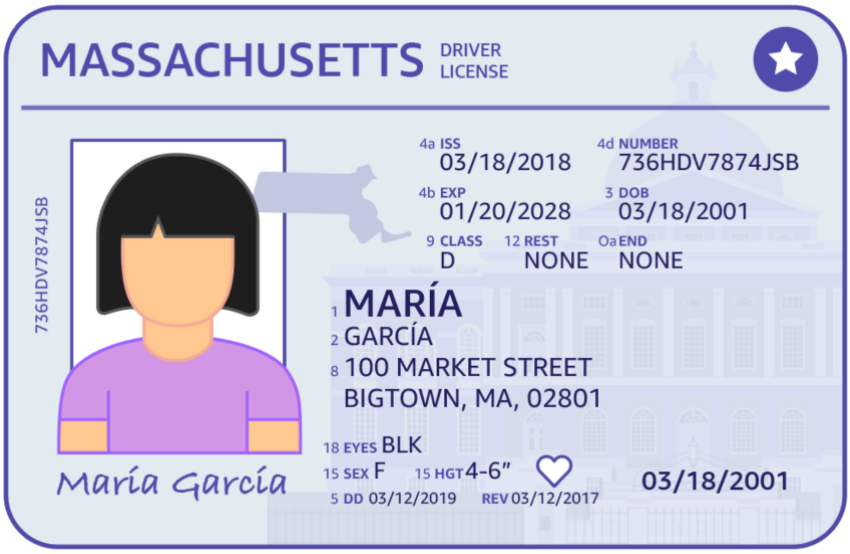
要异步分析发票和收据，请使用StartExpenseAnalysis开始处理输入文档文件。要获得结果，请致电GetExpenseAnalysis. 给定呼叫的结果StartExpenseAnalysis返回方GetExpenseAnalysis. 有关更多信息以及示例，请参阅 使用异步操作处理文档。

分析身份证件

Amazon Textract 可以使用 AnalyZeID API 从美国政府颁发的护照、驾驶执照和其他身份证件中提取相关信息。借助 Analysis ID，企业可以快速准确地从身份证中提取信息，例如具有不同模板或格式的美国驾照、州身份证和护照。AnalyzeID API 返回两类数据类型：

- ID 上可用的键值对，例如出生日期、发行日期、ID #、类别和限制。
- 文档中可能没有与其关联的显式密钥的隐含字段，例如姓名、地址和颁发者。

在响应中，关键名称进行了标准化。例如，如果驾驶执照显示 LIC#（执照号码），而护照显示护照否，那么 Analysis ID 响应将标准化密钥与原始密钥（例如 LIC#）一起返回标准化密钥作为 “文件 ID”。通过这种标准化，客户可以在许多 ID 中轻松组合信息，这些 ID 对同一概念使用不同术



分析 ID 返回名为的结构中的信息IdentityDocumentFields. 这些是JSON包含两条信息的结构：标准化的类型和与该类型关联的值。这两者都有信心分数。有关更多信息，请参阅 身份文档响应对象。

您可以使用同步操作来分析驾驶执照或护照。要分析这些文档，您可以使用 AnalyzeID 操作并向其传递身份证明文件。AnalyzeID返回整组结果。有关更多信息，请参阅 使用 Amazon Textract 分析身份文档。

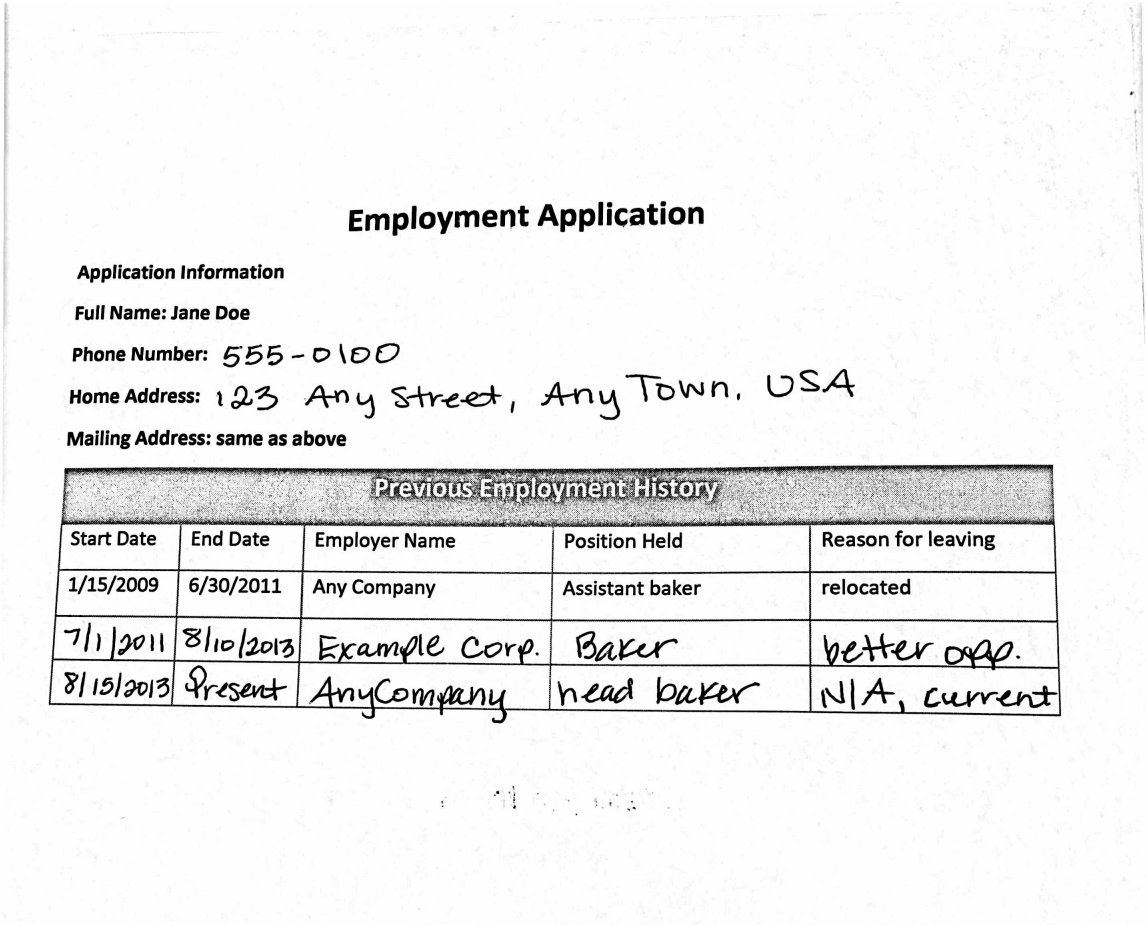
> **Note**
>
> 一些身份证件，例如驾驶执照，有两面。您可以在同一分析 ID API 请求中将驾驶执照的正面和背面图像作为单独的图像传递。

输入文档

Amazon Textract 操作的合适输入是单页或多页文档。一些例子是法律文件、表格、身份证或信件。表单是包含问题或提示的文档，以便用户提供答案。一些例子是患者登记表、纳税表或保险索赔表。

文档可以是 JPEG、PNG、PDF 或 TIFF 格式。使用 PDF 和 TIFF 格式文件，您可以处理多页文档。有关 Amazon Textract 如何将文档表示为的信息`Block`对象，请参阅文本检测和文档分析响应对象.

以下是可接受的输入文档示例。



Employment Application

Application Information

Full Name: Jane Doe

Phone Number: 555-0100

Home Address: 123 Any Street, Any Town, USA

Mailing Address: same as above

Previous Employment History

Start Date	End Date	Employer Name	Position Held	Reason for leaving
1/15/2009	6/30/2011	Any Company	Assistant baker	relocated
7/1/2011	8/10/2013	Example Corp.	Baker	better opp.
8/15/2013	Present	AnyCompany	head baker	N/A, current

有关文档限制的信息，请参阅Amazon Textract 中的硬性限制.

对于 Amazon Textract 同步操作，您可以使用存储在 Amazon S3 存储桶中的输入文档，也可以传递 base64 编码的图像字节。有关更多信息，请参阅 调用 Amazon Textract 同步操作。对于异步操作，您需要在 Amazon S3 存储桶中提供输入文档。有关更多信息，请参阅 调用 Amazon Textract 异步操作。

Amazon Textract 响应对象

Amazon Textract 操作根据运行的操作返回不同类型的对象。为了检测文本和分析通用文档，该操作会返回一个 Block 对象。为了分析发票或收据，该操作会返回 ExpenseDocuments 对象。为了分析身份文档，该操作会返回一个 Identity DocumentFields 对象。有关这些响应对象的更多信息，请参阅以下各部分：

主题

- 文本检测和文档分析响应对象
- 发票和收据回复对象
- 身份文档响应对象

文本检测和文档分析响应对象

当 Amazon Textract 处理文档时，它会创建一个Block检测到或分析的文本的对象。每个区块都包含有关检测到的物品、物品所在位置的信息，以及 Amazon Textract 对处理准确性的信心。

文档由以下类型组成`Block`对象。

- 页面
- 文本的行和单词
- 表单数据（键值对）
- 表格与单元格
- 选择元素

区块的内容取决于你调用的操作。如果调用其中一个文本检测操作，则返回检测到的文本的页面、行和单词。有关更多信息，请参阅 检测文本。如果调用其中一个文档分析操作，将返回有关检测到的页面、键值对、表格、选择元素和文本的信息。有关更多信息，请参阅 分析文档。

一段时间`Block`对象字段在这两种类型的处理中都是通用的。例如，每个区块都有一个唯一的标识符。

有关演示如何使用的示例Block对象，请参阅教程.

文档布局

Amazon Textract 将文档的表示形式作为不同类型的Block在父对子关系或键值对中链接的对象。还会返回提供文档中页数的元数据。以下是典型的 JSON。Block类型的对象PAGE.


```
{
    "Blocks": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Top": 0.0,
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
                ]
            },
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "2602b0a6-20e3-4e6e-9e46-3be57fd0844b",
                        "82aedd57-187f-43dd-9eb1-4f312ca30042",
                        "52be1777-53f7-42f6-a7cf-6d09bdc15a30",
```

```
                        "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"
        }.....

    ],
    "DocumentMetadata": {
        "Pages": 1
    }
}
```

文档由一个或多个创建PAGE数据块。每个页面都包含页面上检测到的主要项目的子块列表，例如文本行和表格行。有关更多信息，请参阅 页面。

你可以确定类型Block通过检查对象BlockType字段中返回的子位置类型。

一个Block对象包含相关列表Block中的对象Relationships字段，这是一个数组Relationship对象。一个Relationships数组是 CHIER 类型或 VALUE 类型。类型为 CHIRD 的数组用于列出当前区块的子项目。例如，如果当前块的类型为 LINE，Relationships包含构成文本行的 WORD 块的 ID 列表。VALUE 类型的数组用于包含键/值对。您可以通过检查Type字段中的Relationship对象。

子块没有关于其父 Block 对象的信息。

对于显示的示例Block请参阅信息使用同步操作处理文档.

信心

Amazon Textract 操作将返回 Amazon Textract 对检测到的商品准确性的置信度的百分比。为了获得信心，请使用Confidence字段中的Block对象。值越大，则置信度越高。视情况而定，信心低的检测可能需要人员的视觉确认。

Geometry

除身份分析外，Amazon Textract 操作会返回有关文档页面上检测到的商品位置的位置信息。要获得位置，请使用Geometry字段中的Block对象。有关更多信息，请参阅 。文档页面上的物品位置

页面

文档包含一个或多个页面。一个the section called “Block”类型的对象PAGE文档的每一页都存在。一个PAGEblock 对象包含在文档页面上检测到的文本行、键值对和表的子 ID 列表。

用于的 JSONPAGE块看上去类似以下内容。

```
{

    "Geometry": ....
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "2602b0a6-20e3-4e6e-9e46-3be57fd0844b", // Line - Hello, world.
                "82aedd57-187f-43dd-9eb1-4f312ca30042", // Line - How are you?
                "52be1777-53f7-42f6-a7cf-6d09bdc15a30",
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"  // Page identifier
},
```

如果对 PDF 格式的多页文档使用异步操作，则可以通过检查Page字段中的Block对象。扫描的图像（JPEG、PNG、PDF 或 TIFF 格式的图像）被视为单页文档，即使图像上有多个文档页面也是如此。异步操作始终返回Page扫描图像的值为 1。

返回的页面总数将在Pages的字段DocumentMetadata.DocumentMetadata与每个列表一起返回BlockAmazon Textract 操作返回的对象。

文本的行和单词

Amazon Textract 操作返回的检测到的文本将返回在the section called “Block”对象。这些对象表示在文档页面上检测到的文本行或文本单词。以下文本显示了由多个单词组成的两行文本。

这是文本。

在两行中。

检测到的文本将在Text字段中的Block对象。这些区域有：BlockType字段确定文本是文本行 (LINE) 还是单词 (WORD)。一个字是一个或多个 ISO 基本拉丁文字母字符，不用空格分隔。一个线是一串制表符分隔的连续单词。

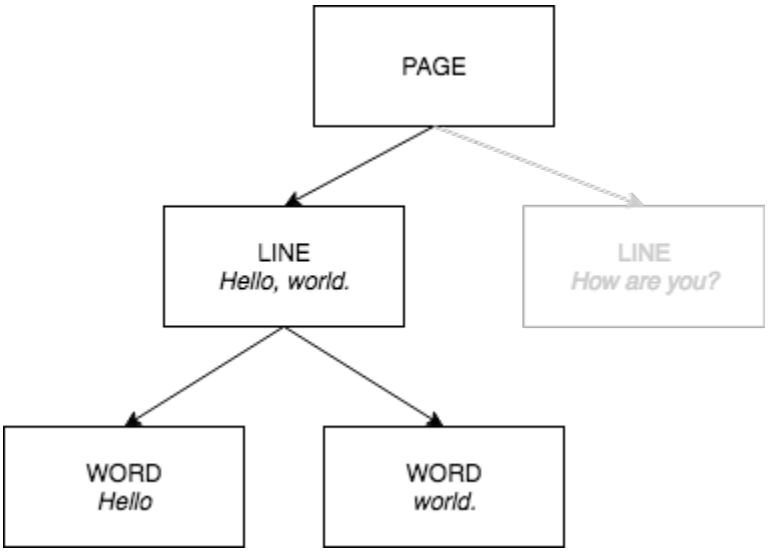
此外，Amazon Textract 将确定一段文本是使用手写还是使用`TextTypes`字段中返回的子位置类型。这些分别以手写和印刷的形式返回。

另一个`Block`属性对于所有区块类型都是共有的，例如 ID、置信度和几何信息。有关更多信息，请参阅 the section called “文本检测和文档分析响应对象”。

要只检测行和单词，你可以使用DetectDocumentText要么StartDocumentTextDetection. 有关更多信息，请参阅 检测文本。要获取检测到的文本（行和单词）以及有关其与文档其他部分（例如表格）的关系的信息，可以使用AnalyzeDocument要么StartDocumentAnalysis. 有关更多信息，请参阅 分析文档。

`PAGE`、`LINE`, 和`WORD`在父子关系中，区块彼此相关。一个`PAGE`方块是所有人的父项`LINE`阻止文档页面上的对象。因为 LINE 可以有一个或多个单词，因此`Relationships`LINE 块的数组存储构成文本行的子 WORD 块的 ID。

下图显示了该行的方式。Hello world。中的文本Hello world。你怎么样？代表为`Block`对象。



以下是来自的 JSON 输出`DetectDocumentText`当句子Hello world。你怎么样？检测到的对象。第一个例子是文档页面的 JSON。请注意孩子 ID 如何使您能够在文档中导航。

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "d7fbd604-d609-4d69-857d-247a3f591238", // Line - Hello, world.
                "b6c19a93-6493-4d8e-958f-853c8f7ca055" //  Line - How are you?
```

```
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "56ec1d77-171f-4881-9852-2b5b7e761608"
},
```

以下是构成 “你好，世界” 行的 LINE 块的 JSON：

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "7f97e2ca-063e-47a8-981c-8beee31afc01", // Word - Hello,
                "4b990aa0-af96-4369-b90f-dbe02538ed21"  // Word - world.
            ]
        }
    ],
    "Confidence": 99.63229370117188,
    "Geometry": {...},
    "Text": "Hello, world.",
    "BlockType": "LINE",
    "Id": "d7fbd604-d609-4d69-857d-247a3f591238"
},
```

以下是该词的 WORD 块的 JSONHello：

```
{
    "Geometry": {...},
    "Text": "Hello,",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.74746704101562,
    "Id": "7f97e2ca-063e-47a8-981c-8beee31afc01"
},
```

最后一个 JSON 是这个词的 WORD 块世界。：

```
{
    "Geometry": {...},
    "Text": "world.",
```

```
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.5171127319336,
    "Id": "4b990aa0-af96-4369-b90f-dbe02538ed21"
},
```

表单数据（键值对）

Amazon Textract 可以将表单数据从文档中提取为键值对。例如，在以下文本中，Amazon Textract 可以识别密钥 (名称：) 和一个值 (Ana Carolina)。

名称：Ana Carolina

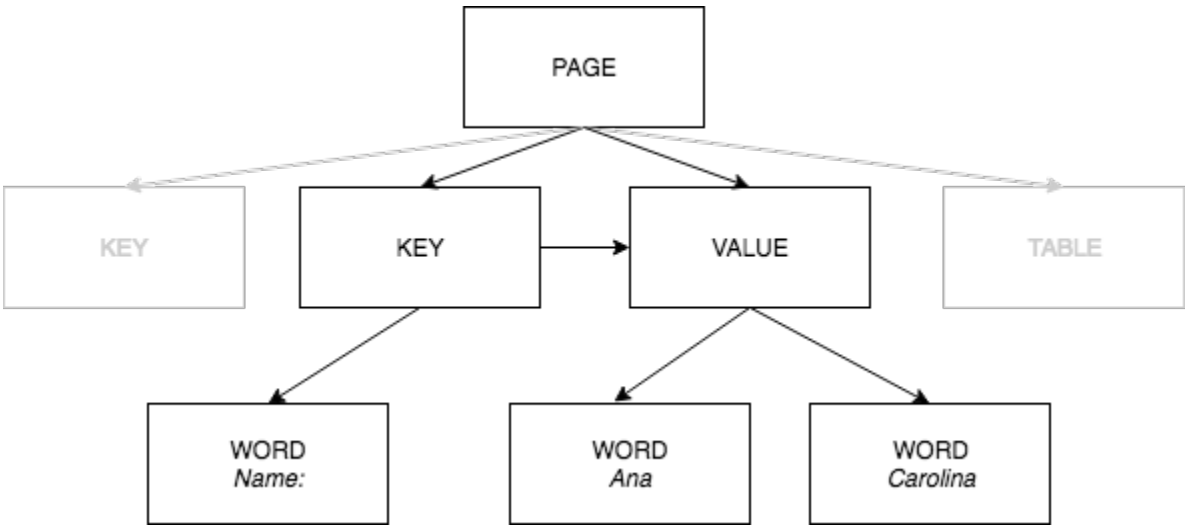
检测到的键值对将返回为Block来自的响应中的对象AnalyzeDocument和GetDocumentAnalysis. 您可以使用FeatureTypes输入参数来检索关于键值对、表或两者的信息。仅对于键值对，请使用值FORMS. 有关示例，请参阅 从表单文档中提取键值对。有关文档如何表示的一般信息，请参阅Block对象，请参阅文本检测和文档分析响应对象.

类型为 KEY_VALUE_SET 的块对象是 KEY_VALUE_SET 对象的容器，用于存储文档中检测到的链接文本项目的信息的 KEY 或 您可以使用EntityType属性来确定块是 KEY 还是 VALUE。

- 一个密钥对象包含有关链接文本键的信息。例如，名称：. KEY 区块有两个关系列表。VALUE 类型的关系是一个列表，其中包含与该密钥关联的 VALUE 块的 ID。CHIRD 类型的关系是组成密钥文本的 WORD 块的 ID 列表。
- 一个值对象包含有关键相关文本的信息。在上述示例中，Ana Carolina是键的值名称：. VALUE 块与识别 WORD 块的子块列表有关系。每个 WORD 块都包含组成该值文本的一个单词。一个VALUE对象还可以包含有关选定元素的信息。有关更多信息，请参阅 选择元素。

KEY_VALUE_SET 的每个实例Block对象是 PAGE 的子项Block对应于当前页面的对象。

下图显示键值对的方式。名称：Ana Carolina代表为Block对象。



以下示例演示键值对的方式。名称：Ana Carolina由 JSON 表示。

PAGE 块有类型的 CHILD 块KEY_VALUE_SET针对文档中检测到的每个 KEY 和 VALUE 块。

```
{
    "Geometry": ....
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "2602b0a6-20e3-4e6e-9e46-3be57fd0844b",
                "82aedd57-187f-43dd-9eb1-4f312ca30042",
                "52be1777-53f7-42f6-a7cf-6d09bdc15a30", // Key - Name:
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"  // Value - Ana Caroline
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"  // Page identifier
},
```

以下 JSON 显示了 KEY 区块（52be1777-53f7-42f6-a7cf-6d09bdc15a30）与价值区块（7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c）有关系。它还有一个字块的子块（c734fca6-c4c4-415c-b6c1-30f7510b72ee），其中包含密钥的文本（名称：)。

```
{
    "Relationships": [
        {
            "Type": "VALUE",
```

```
            "Ids": [
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"  // Value identifier
            ]
        },
        {
            "Type": "CHILD",
            "Ids": [
                "c734fca6-c4c4-415c-b6c1-30f7510b72ee"  // Name:
            ]
        }
    ],
    "Confidence": 51.55965805053711,
    "Geometry": ....,
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "KEY"
    ],
    "Id": "52be1777-53f7-42f6-a7cf-6d09bdc15a30"  //Key identifier
},
```

下面的 JSON 显示了 VALUE 块 7ca7ca6-00ef-4cda-b1aa-5571dfed1a7c 有一个组成该值文本的 WORD 块 ID 的子列表 (安娜和卡罗纳)。

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "db553509-64ef-4ecf-ad3c-bea62cc1cd8a", // Ana
                "e5d7646c-eaa2-413a-95ad-f4ae19f53ef3"  // Carolina
            ]
        }
    ],
    "Confidence": 51.55965805053711,
    "Geometry": ....,
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "VALUE"
    ],
    "Id": "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c" // Value identifier
}
```

下面的 JSON 显示了Block单词的对象名称：、安娜, 和卡罗纳.

```
{
    "Geometry": {...},
    "Text": "Name:",
    "TextType": "PRINTED".
    "BlockType": "WORD",
    "Confidence": 99.56285858154297,
    "Id": "c734fca6-c4c4-415c-b6c1-30f7510b72ee"
},
 {
    "Geometry": {...},
    "Text": "Ana",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.52057647705078,
    "Id": "db553509-64ef-4ecf-ad3c-bea62cc1cd8a"
},
{
    "Geometry": {...},
    "Text": "Carolina",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.84207916259766,
    "Id": "e5d7646c-eaa2-413a-95ad-f4ae19f53ef3"
},
```

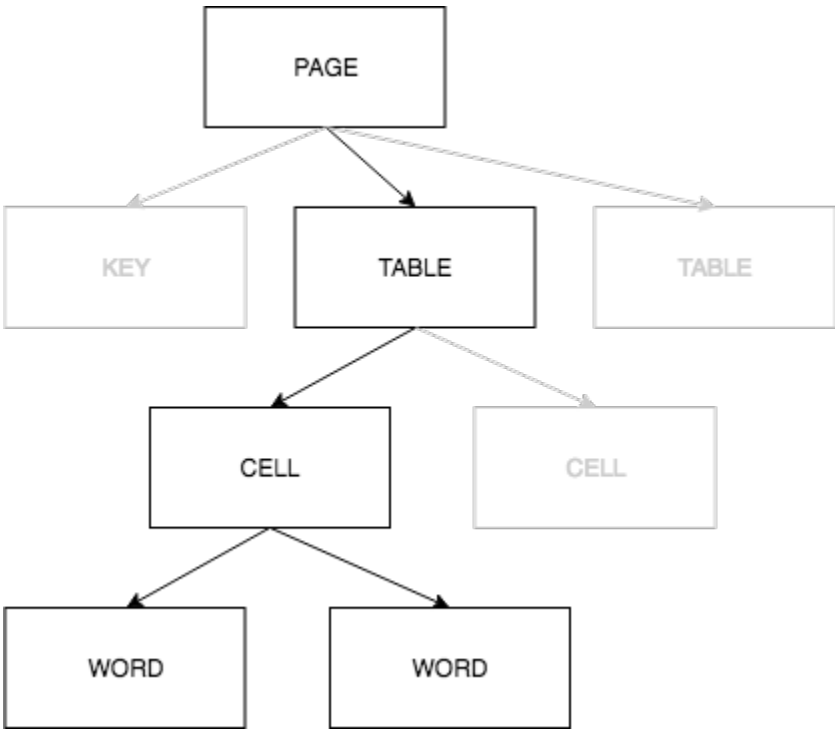

表

Amazon Textract 可以提取表格和表格中的单元格。例如，在表单上检测到下表时，Amazon Textract 会检测到包含四个单元格的表格。

名称	Address
Ana Carolina	123 Aany Stown

检测到的表将返回为Block来自的响应中的对象AnalyzeDocument和GetDocumentAnalysis. 您可以使用FeatureTypes输入参数来检索关于键值对、表或两者的信息。仅对于表格，请使用值TABLES. 有关示例，请参阅 将表导出到 CSV 文件。有关文档如何表示的一般信息，请参阅Block对象，请参阅文本检测和文档分析响应对象.

下图显示了表中的单个单元格的表示方式。Block对象。



包含单元格WORD阻止检测到的单词，以及SELECTION_ELEMENT选择元素（例如复选框）的块。

以下是上表的部分 JSON，该表有四个单元格。

PAGE Block 对象有一个 TABLE 块的子块 ID 列表以及检测到的每一行文本。

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "f2a4ad7b-f21d-4966-b548-c859b84f66a4",   // Line - Name
                "4dce3516-ffeb-45e0-92a2-60770e9cb744",   // Line  - Address
                "ee506578-768f-4696-8f4b-e4917e429f50",   // Line - Ana Carolina
                "33fc7223-411b-4399-8a90-ccd3c5a2c196",   // Line  - 123 Any Town
                "3f9665be-379d-4ae7-be44-d02f32b049c2"    // Table
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "78c3ce84-ae70-418e-add7-27058418adf6"
},
```

TABLE 模块包括表中单元格的子 ID 列表。TABLE 模块还包括文档中表位置的几何信息。下面的 JSON 显示该表包含四个单元格，这四个单元格列在Ids数组。

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "505e9581-0d1c-42fb-a214-6ff736822e8c",
                "6fca44d4-d3d3-46ab-b22f-7fca1fbaaf02",
                "9778bd78-f3fe-4ae1-9b78-e6d29b89e5e9",
                "55404b05-ae12-4159-9003-92b7c129532e"
            ]
        }
    ],
    "BlockType": "TABLE",
    "Confidence": 92.5705337524414,
    "Id": "3f9665be-379d-4ae7-be44-d02f32b049c2"
},
```

表格单元格的块类型为 CELL。这些区域有：Block每个单元格的对象包括与表中其他单元格相比单元格的单元格位置的信息。它还包括文档中单元格位置的几何信息。在上述示例中，505e9581-0d1c-42fb-a214-6ff736822e8c是包含该单词的单元格的子 ID名称. 以下示例是该单元格的信息。

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "e9108c8e-0167-4482-989e-8b6cd3c3653e"
            ]
        }
    ],
    "Confidence": 100.0,
    "RowSpan": 1,
    "RowIndex": 1,
    "ColumnIndex": 1,
    "ColumnSpan": 1,
    "BlockType": "CELL",
```

```
    "Id": "505e9581-0d1c-42fb-a214-6ff736822e8c"
},
```

每个单元格在桌子中都有一个位置，第一个单元格为 1,1。在上一示例中，具有值的单元格名称位于第 1 行，第 1 列。具有值的单元格123 Aany Stown位于第 2 行，第 2 列。单元格块对象包含此信息在RowIndex和ColumnIndex字段之间没有不同。子列表包含包含单元格中文本的 WORD Block 对象的 ID。列表中的单词按照检测到它们的顺序，从单元格的左上角到单元格的右下角。在前面的示例中，单元格具有一个值为 e9108c8e-0167-4482-989e-8b6cd3c3653e 的子 ID。以下输出针对 ID 值为 e9108c8e-0167-4482-989e-8e-8b6cd3c3653e 的字块：

```
"Geometry": {...},
"Text": "Name",
"TextType": "Printed",
"BlockType": "WORD",
"Confidence": 99.81139373779297,
"Id": "e9108c8e-0167-4482-989e-8b6cd3c3653e"
},
```

选择元素

Amazon Textract 可以检测选择元素，例如文档页面上的选项按钮（单选按钮）和复选框。可以在中检测到选择元素表单数据然后在桌子. 例如，在表单上检测到下表时，Amazon Textract 会检测到表格单元格中的复选框。

	同意	Neutral	不同意
服务不错	☑	☐	☐
易于使用	☐	☑	☐
公平的价格	☑	☐	☐

检测到的选择元素将返回为Block来自的响应中的对象AnalyzeDocument和GetDocumentAnalysis.

> **Note**
>
> 您可以使用FeatureTypes输入参数来检索关于键值对、表或两者的信息。例如，如果对表进行筛选，则响应将包括在表中检测到的选择元素。在键值对中检测到的选择元素不包括在响应中。

有关选择元素的信息包含在Block类型的对象SELECTION_ELEMENT. 要确定可选元素的状态，请使用SelectionStatus字段中的SELECTION_ELEMENT阻止。状态可以是已选中要么没有选择. 例如，的值SelectionStatus对于上一张图片是已选中.

一个SELECTION_ELEMENT Block对象与键值对或表格单元格关联。一个SELECTION_ELEMENT Block对象包含选择元素的边界框信息Geometry字段中返回的子位置类型。一个SELECTION_ELEMENT Block对象不是一个孩子PAGE Block对象。

表单数据（键值对）

键值对用于表示在表单上检测到的选择元素。这些区域有：KEY块包含选择元素的文本。这些区域有：VALUE块包含 SELECTION_ELEMENT 块。下图显示了选择元素的表示方式。the section called “Block”对象。

有关键值对的更多信息，请参阅表单数据（键值对）.

以下 JSON 代码段显示了包含选择元素的键值对的键值对的键（男 ☑）。子 ID（编号 bd14cfd5-9005-498b-a7f3-45ceb171f0ff）是包含选择元素文本的 WORD 块的 ID（男）。值 ID（编号 24aaac7f-FCC-49c7-a4f0-3688b05586d4）是VALUE包含SELECTION_ELEMENT阻止对象。

```
{
    "Relationships": [
        {
            "Type": "VALUE",
            "Ids": [
                "24aaac7f-fcce-49c7-a4f0-3688b05586d4"  // Value containing Selection
 Element
            ]
        },
        {
            "Type": "CHILD",
            "Ids": [
                "bd14cfd5-9005-498b-a7f3-45ceb171f0ff"  // WORD - male
```

```
            ]
        }
    ],
    "Confidence": 94.15619659423828,
    "Geometry": {
        "BoundingBox": {
            "Width": 0.022914813831448555,
            "Top": 0.08072036504745483,
            "Left": 0.18966935575008392,
            "Height": 0.014860388822853565
        },
        "Polygon": [
            {
                "Y": 0.08072036504745483,
                "X": 0.18966935575008392
            },
            {
                "Y": 0.08072036504745483,
                "X": 0.21258416771888733
            },
            {
                "Y": 0.09558075666427612,
                "X": 0.21258416771888733
            },
            {
                "Y": 0.09558075666427612,
                "X": 0.18966935575008392
            }
        ]
    },
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "KEY"
    ],
    "Id": "a118dc43-d5f7-49a2-a20a-5f876d9ffd79"
}
```

以下 JSON 片段是该词的 WORD 块男. WORD 模块还有一个父级 LINE 块。

```
{
    "Geometry": {
        "BoundingBox": {
            "Width": 0.022464623674750328,
```

```
            "Top": 0.07842985540628433,
            "Left": 0.18863198161125183,
            "Height": 0.01617223583161831
        },
        "Polygon": [
            {
                "Y": 0.07842985540628433,
                "X": 0.18863198161125183
            },
            {
                "Y": 0.07842985540628433,
                "X": 0.2110965996980667
            },
            {
                "Y": 0.09460209310054779,
                "X": 0.2110965996980667
            },
            {
                "Y": 0.09460209310054779,
                "X": 0.18863198161125183
            }
        ]
    },
    "Text": "Male",
    "BlockType": "WORD",
    "Confidence": 54.06439208984375,
    "Id": "bd14cfd5-9005-498b-a7f3-45ceb171f0ff"
},
```

VALUE 块有一个子项（编号为 f2f5e8d-e73a-4e99-a095-053acd3b6bfb），即选择 _ELECTION_E8d-e73a-4e99-a095-053acd3b6bfb）。

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "f2f5e8cd-e73a-4e99-a095-053acd3b6bfb"  // Selection element
            ]
        }
    ],
    "Confidence": 94.15619659423828,
    "Geometry": {
```

```
        "BoundingBox": {
            "Width": 0.017281491309404373,
            "Top": 0.07643391191959381,
            "Left": 0.2271782010793686,
            "Height": 0.026274094358086586
        },
        "Polygon": [
            {
                "Y": 0.07643391191959381,
                "X": 0.2271782010793686
            },
            {
                "Y": 0.07643391191959381,
                "X": 0.24445968866348267
            },
            {
                "Y": 0.10270800441503525,
                "X": 0.24445968866348267
            },
            {
                "Y": 0.10270800441503525,
                "X": 0.2271782010793686
            }
        ]
    },
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "VALUE"
    ],
    "Id": "24aaac7f-fcce-49c7-a4f0-3688b05586d4"
},
}
```

以下 JSON 是 SELECTION_ELEMENT 块。的价值`SelectionStatus`表示复选框处于选中状态。

```
{
    "Geometry": {
        "BoundingBox": {
            "Width": 0.020316146314144135,
            "Top": 0.07575977593660355,
            "Left": 0.22590067982673645,
            "Height": 0.027631107717752457
        },
```

```
        "Polygon": [
            {
                "Y": 0.07575977593660355,
                "X": 0.22590067982673645
            },
            {
                "Y": 0.07575977593660355,
                "X": 0.2462168186903
            },
            {
                "Y": 0.1033908873796463,
                "X": 0.2462168186903
            },
            {
                "Y": 0.1033908873796463,
                "X": 0.22590067982673645
            }
        ]
    },
    "BlockType": "SELECTION_ELEMENT",
    "SelectionStatus": "SELECTED",
    "Confidence": 74.14942932128906,
    "Id": "f2f5e8cd-e73a-4e99-a095-053acd3b6bfb"
}
```

表单元格

Amazon Textract 可以检测表格单元格中的选择元素。例如，下表中的单元格有复选框。

	同意	Neutral	不同意
服务不错	☑	☐	☐
易于使用	☐	☑	☐
公平的价格	☑	☐	☐

一个CELL可以包含子项SELECTION_ELEMENT用于选择元素的对象以及子元素WORD阻止检测到的文本。

有关表的更多信息，请参阅表.

TABLEBlock上一个表格的对象看起来类似于这个。

```
{
    "Geometry": {.....},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "652c09eb-8945-473d-b1be-fa03ac055928",
                "37efc5cc-946d-42cd-aa04-e68e5ed4741d",
                "4a44940a-435a-4c5c-8a6a-7fea341fa295",
                "2de20014-9a3b-4e26-b453-0de755144b1a",
                "8ed78aeb-5c9a-4980-b669-9e08b28671d2",
                "1f8e1c68-2c97-47b2-847c-a19619c02ca9",
                "9927e1d1-6018-4960-ac17-aadb0a94f4d9",
                "68f0ed8b-a887-42a5-b618-f68b494a6034",
                "fcba16e0-6bd7-4ea5-b86e-36e8330b68ea",
                "2250357c-ae34-4ed9-86da-45dac5a5e903",
                "c63ad40d-5a14-4646-a8df-2d4304213dbc",   // Cell
                "2b8417dc-e65f-4fcd-aa0f-61a23f1e8cb0",
                "26c62932-72f0-4dc2-9893-1ae27829c060",
                "27f291cc-abf4-4c23-aa24-676abe99cb1e",
                "7e5ce028-1bcd-4d9f-ad42-15ac181c5b47",
                "bf32e3d2-efa2-4fc1-b09b-ab9cc52ff734"
            ]
        }
    ],
    "BlockType": "TABLE",
    "Confidence": 99.99993896484375,
    "Id": "f66eac36-2e74-406e-8032-14d1c14e0b86"
}
```

单元格BLOCK包含复选框的单元格的对象（ID c63ad40d-5a14-4646-a8df-2d4304213dbc）服务不错看上去与下类似。它包括一个孩子Block（Id = 26d122fd-c5f4-4b53-92c4-0ae92730ee1e）这就是SELECTION_ELEMENT Block对象用于复选框。

```
{
    "Geometry": {.....},
    "Relationships": [
        {
            "Type": "CHILD",
```

```
            "Ids": [
                "26d122fd-c5f4-4b53-92c4-0ae92730ee1e"  // Selection Element
            ]
        }
    ],
    "Confidence": 79.741689682006836,
    "RowSpan": 1,
    "RowIndex": 3,
    "ColumnIndex": 3,
    "ColumnSpan": 1,
    "BlockType": "CELL",
    "Id": "c63ad40d-5a14-4646-a8df-2d4304213dbc"
}
```

Selection_ElementBlock对象如下所示。的价值SelectionStatus表示复选框处于选中状态。

```
{
    "Geometry": {.......},
    "BlockType": "SELECTION_ELEMENT",
    "SelectionStatus": "SELECTED",
    "Confidence": 88.79517364501953,
    "Id": "26d122fd-c5f4-4b53-92c4-0ae92730ee1e"
}
```

发票和收据回复对象

当您向分析费用 API 提交发票或收据时，它会返回一系列 ExpenseDocuments 对象。每个 ExpeenseDocument 都进一步分成LineItemGroups和SummaryFields. 大多数发票和收据都包含供应商名称、收据编号、收款日期或总金额等信息。分析费用将此信息返回SummaryFields. 收据和发票还包含有关购买物品的详细信息。分析费用 API 在下返回此信息LineItemGroups. 这些区域有：ExpenseIndex字段唯一标识费用，并将相应的SummaryFields和LineItemGroups在该费用中检测到。

分析费用响应中最精细的数据级别包括：Type、ValueDetection, 和LabelDetection(可选)。各个实体是：

- 类型：指在高层次上检测到什么类型的信息。
- 标签检测：指文档文本中关联值的标签。LabelDetection是可选的，只有在写入标签时才返回。
- 价值检测：指返回的标签或类型的值。

分析费用 API 还可以检测到ITEM、QUANTITY, 和PRICE在行项目中作为标准化字段。如果收据图片上的行项目中还有其他文本，例如 SKU 或详细描述，则该文本将包含在 JSON 中EXPENSE_ROW如下面的示例所示：

```
            {
                                "Type": {
                                    "Text": "EXPENSE_ROW",
                                    "Confidence": 99.95216369628906
                                },
                                "ValueDetection": {
                                    "Text": "Banana 5 $2.5",
                                    "Geometry": {
                                      …
                                    },
                                    "Confidence": 98.11214447021484
                                }
```

上面的示例显示了 AnalyzeSendal API 如何在收据上返回整行，该收据包含关于售价 2.5 美元的 5 只香蕉的订单项目信息。

类型

以下是键值对的标准类型或标准化类型的示例：

```
            {
                 "PageNumber": 1,
                 "Type": {
                     "Text": "VENDOR_NAME",
                     "Confidence": 70.0
                 },
                 "ValueDetection": {
                     "Geometry": { ... },
                     "Text": "AMAZON",
                     "Confidence": 87.89806365966797
                 }
             }
```

收据没有明确列出 “供应商名称”。但是，分析费用 API 将该文档识别为收据，并将值 “亚马逊” 归类为 “类型”VENDOR_NAME.

标签检测

以下是在客户文档页面上显示的文本示例：

```
            {
                 "PageNumber": 1,
                 "Type": {
                     "Text": "OTHER",
                     "Confidence": 70.0
                 },
                 "LabelDetection": {
                     "Geometry": { ... },
                     "Text": "CASHIER",
                     "Confidence": 88.19171142578125
                 },
                 "ValueDetection": {
                     "Geometry": { ... },
                     "Text": "Mina",
                     "Confidence": 87.89806365966797
                 }
             }
```

示例文档包含 “收银员米娜”。分析费用 API 提取了原样值并将其返回LabelDetection. 对于 “供应商名称” 之类的隐含值，其中 “钥匙” 没有在收据中明确显示，LabelDetection不会包含在分析费用元素中。在这种情况下，分析费用 API 不会返回LabelDetection.

价值检测

以下示例显示键值对的 “值”。

```
            {
                 "PageNumber": 1,
                 "Type": {
                     "Text": "OTHER",
                     "Confidence": 70.0
                 },
```

```
                    "LabelDetection": {
                        "Geometry": { ... },
                        "Text": "CASHIER",
                        "Confidence": 88.19171142578125
                    },
                    "ValueDetection": {
                        "Geometry": { ... },
                        "Text": "Mina",
                        "Confidence": 87.89806365966797
                    }
                }
```

在示例中，该文档包含 “收银员米娜”。分析费用 API 检测到出纳柜员价值为 Mina 并将其退回`ValueDetection`.

身份文档响应对象

当您向 AnalyZEID API 提交身份证件时，它会返回一系列`IdentityDocumentField`对象。这些对象中的每个都包含`Type`, 和`Value`.`Type`记录 Amazon Textract 检测到的标准化字段，并`Value`记录与标准化字段关联的文本。

以下是示例：`IdentityDocumentField`，为了简洁起见，缩短了。

```
{
    "DocumentMetadata": {
        "Pages": 1
    },
    "IdentityDocumentFields": [
        {
            "Type": {
                "Text": "first name"
            },
            "ValueDetection": {
                "Text": "jennifer",
                "Confidence": 99.99908447265625
            }
        },
        {
            "Type": {
                "Text": "last name"
            },
```

```
            "ValueDetection": {
                "Text": "sample",
                "Confidence": 99.99758911132812
            }
        },
```

这是从较长的响应中切断的身份文档字段的两个示例。检测到的类型和该类型的值之间存在分离。这里分别是名字和姓氏。这种结构与所有包含的信息重复。如果类型未被识别为标准化字段，它将被列为“其他”。

以下是驾驶执照的标准化字段的列表：

- 名
- 姓
- 中间名
- 后缀
- 地址中的城市
- 地址中的邮政编码
- 在地址中的状态
- County
- 文件编号
- 到期日期
- 出生日期
- 州名
- 发行日期
- class
- 限制
- 代言
- ID 类型
- 老兵
- 地址

以下是美国护照标准化字段的列表：

- 名
- 姓
- 中间名
- 文件编号
- 到期日期
- 出生日期
- 出生地
- 发行日期
- ID 类型

文档页面上的物品位置

Amazon Textract 操作会返回文档页面上找到的商品的位置和几何图形。DetectDocumentText和GetDocumentTextDetection返回线条和单词的位置和几何图形，而AnalyzeDocument和GetDocumentAnalysis返回键值对、表、单元格和选择元素的位置和几何。

要确定项目在文档页面上的位置，请使用边界框 (Geometry) Amazon Textract 操作在Block对象。这些区域有：Geometry对象包含两种类型的位置和几何信息，适用于检测到的项目

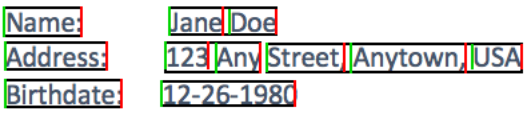
- 轴对齐BoundingBox对象，包含左上坐标以及项目的宽度和高度。
- 描述项目轮廓的多边形对象，指定为Point包含的对象X（水平轴）和Y（垂直轴）每个点的文档页面坐标。

用于的 JSONBlock对象看上去类似以下内容。请注意BoundingBox和Polygon字段之间没有不同。

```
{
    "Geometry": {
        "BoundingBox": {
            "Width": 0.053907789289951324,
            "Top": 0.08913730084896088,
            "Left": 0.11085548996925354,
            "Height": 0.013171200640499592
        },
        "Polygon": [
            {
                "Y": 0.08985357731580734,
                "X": 0.11085548996925354
```

```
            },
            {
                "Y": 0.08913730084896088,
                "X": 0.16447919607162476
            },
            {
                "Y": 0.10159222036600113,
                "X": 0.16476328670978546
            },
            {
                "Y": 0.10230850428342819,
                "X": 0.11113958805799484
            }
        ]
    },
    "Text": "Name:",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.56285858154297,
    "Id": "c734fca6-c4c4-415c-b6c1-30f7510b72ee"
},
```


您可以使用几何信息在检测到的物品周围绘制边界框。对于使用的示例BoundingBox和Polygon在每个单词的开头和结尾围绕线条和垂直线画框的信息，请参阅使用 Amazon Textract 检测文档文本. 示例输出类似以下内容。



Bounding Box

一个边界框（BoundingBox）具有以下属性：

- 高度 — 边界框的高度（以占整个文档页面高度的比例显示）。
- 左侧 — 边界框左上点的 X 坐标（以占整个文档页面宽度的比例显示）。
- 顶部 — 边界框左上点的 Y 坐标（以占整个文档页面高度的比例显示）。
- 宽度 — 边界框的宽度（以占整个文档页面宽度的比例显示）。

每个 BoundingBox 属性都有一个介于 0 和 1 之间的值。该值是占整个图像宽度的比例（适用于）。Left和Width) 或身高（适用于Height和Top)。例如，如果输入图像为 700 x 200 像

素，而边界框的左上坐标为 (350,50) 像素，则 API 将返回Left值为 0.5 (350/700) 和Top值为 0.25（50/200）。

下图显示了每个 BoundingBox 属性覆盖的文档页面的范围。

要显示位置和大小正确的边界框，您必须将 BoundingBox 值乘以文档页面宽度或高度（具体取决于所需的值）以获取像素值。使用像素值显示边界框。一个示例是使用 608 像素宽 x 588 像素高的文档页面，对分析的文本使用以下边界框值：

```
BoundingBox.Left: 0.3922065
BoundingBox.Top: 0.15567766
BoundingBox.Width: 0.284666
BoundingBox.Height: 0.2930403
```

以像素为单位的文本边界框的位置的计算方法如下：

```
Left coordinate = BoundingBox.Left (0.3922065) * document page width (608)
= 238
```

```
Top coordinate = BoundingBox.Top (0.15567766) * document page height (588)
= 91
```

```
Bounding box width = BoundingBox.Width (0.284666) * document page width
(608) = 173
```

```
Bounding box height = BoundingBox.Height (0.2930403) * document page height
(588) = 172
```

您可以使用这些值围绕分析的文本显示边界框。以下 Java 和 Python 示例演示如何显示边界框。

Java

```
   public 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 / scale), Math.round(top / scale),
```

```
                Math.round((imageWidth * box.getWidth()) / scale),
 Math.round((imageHeight * box.getHeight())) / scale);

    }
```

Python

这个 Python 示例采用`response`返回的DetectDocumentTextAPI 操作。

```
def process_text_detection(response):

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

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:

        draw = ImageDraw.Draw(image)

        if block['BlockType'] == "LINE":
            box=block['Geometry']['BoundingBox']
            left = width * box['Left']
            top = height * box['Top']
            draw.rectangle([left,top, left + (width * box['Width']), top +(height *
 box['Height'])],outline='black')

    # Display the image
    image.show()

    return len(blocks)
```

面

返回的多边形`AnalyzeDocument`是数组Point对象。EALE`Point`在文档页面上具有特定位置的 X 和 Y 坐标。与 BoundingBox 坐标一样，多边形坐标标准化为文档宽度和高度，并且介于 0 到 1 之间。

您可以使用多边形数组中的点在`Block`对象。您可以使用相同的方法来计算文档页面上每个多边形点的位置`BoundingBoxes`. 将 X 坐标乘以文档页面宽度，然后将 Y 坐标乘以文档页面高度。

以下示例演示如何显示多边形的垂直线。

```
   public 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));

   }
```

Amazon Textract 入门

本节提供了多个主题，可帮助您开始使用 Amazon Textract。如果您是首次使用 Amazon Textract，建议您先查看中的概念和术语。Amazon Textract 的工作原理.

您可以通过使用 Amazon Textract 控制台中的演示来尝试 API。有关更多信息，请参阅 。https://console.aws.amazon.com/textract/.

主题

- 第 1 步：设置 AWS 账户并创建 IAM 用户
- 第 2 步：设置AWS CLI和AWS软件开发工具包
- 第 3 步：开始使用AWS CLI和AWSSDK API

第 1 步：设置 AWS 账户并创建 IAM 用户

首次使用 Amazon Textract 前，请完成以下任务：

1. 注册 AWS.
2. 创建 IAM 用户.

注册 AWS

在注册 Amazon Web Services (AWS) 时，您的 AWS 账户会自动注册 AWS 中的所有发布服务。您只需为使用的服务付费。

借助 Amazon Textract，您仅需为实际使用的资源付费。有关 Amazon Textract 使用费率的更多信息，请参阅。Amazon Textract 定价. 如果您是 AWS 新客户，则可以免费试用 Amazon Textract。有关更多信息，请参阅 。AWS 免费使用套餐.

如果您已有一个 AWS 账户，请跳到下一个任务。如果您还没有 AWS 账户，请按照以下过程中的步骤创建。

创建亚马逊云科技账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

请记住您的 AWS 账户 ID，因为进行下一个任务时需要用到。

创建 IAM 用户

AWS 中的服务（如 Amazon Textract）要求您在访问时提供凭证，这样，服务才能确定您是否有权访问该服务所拥有的资源。控制台要求您的密码。您可以为您的 AWS 账户创建访问密钥以访问密钥以访问AWS CLI或者 API。但是，我们不建议使用您的 AWS 账户的凭证访问 AWS。而是建议您：

- 使用AWS Identity and Access Management(IAM) 以创建 IAM 用户。
- 将用户添加到具有管理权限的 IAM 组中.

您随后可以使用特殊 URL 和该 IAM 用户的凭证访问 AWS。

如果您已注册 AWS 但尚未为自己创建 IAM 用户，则可以使用 IAM 控制台自行创建。请按照以下过程在您的账户中创建 IAM 用户。

创建 IAM 用户和登录控制台

1. 在您的 AWS 账户中创建一个具有管理员权限的 IAM 用户。有关说明，请参阅创建您的第一个 IAM 用户和管理员组中的IAM 用户指南.
2. 以此 IAM 用户的身份，使用特殊 URL 登录 AWS 管理控制台。有关更多信息，请参阅 。用户如何登录您的账户中的IAM 用户指南.

> **Note**
>
> 具有管理员权限的 IAM 用户可以无限制地访问AWS账户中的服务。本指南中的代码示例假定您有一个具有`AmazonTextractFullAccess`权限。`AmazonS3ReadOnlyAccess`要访问 Amazon S3 存储桶中存储的文档的示例需要权限。根据您的安全要求，您可能希望使用限于这些权限的 IAM 组。有关更多信息，请参阅 。创建 IAM 组.

有关 IAM 的更多信息，请参阅以下文档：

- AWS Identity and Access Management (IAM)

- 入门
- IAM 用户指南

下一个步骤

第 2 步：设置AWS CLI和AWS软件开发工具包

第 2 步：设置AWS CLI和AWS软件开发工具包

以下步骤说明如何安装本文档中的示例使用的 AWS Command Line Interface (AWS CLI) 和 AWS 开发工具包。

有多种方法可对 AWS SDK 开发工具包调用进行身份验证。本指南中的示例假定您使用默认的凭证配置文件调用 AWS CLI 命令和 AWS 开发工具包 API 操作。您的默认凭据将跨服务运行，因此如果您已经配置了凭据，则无需再次进行配置。但是，如果您想为此服务创建另一组凭据，则可以创建名称配置文件。有关创建配置文件的更多信息，参阅命名配置文件。

有关可用的列表AWS地区，请参阅区域和终端节点中的Amazon Web Services 常规参考.

设置 AWS CLI 和 AWS 开发工具包

1. 下载并安装 AWS CLI 和您要使用的 AWS 开发工具包。本指南提供了示例，AWS CLI、Java 和 Python。有关其他 AWS 开发工具包的信息，请参阅用于 Amazon Web Services 的工具。
   - AWS CLI
   - 适用于 Java 的 AWS SDK
   - 适用于 Python (Boto3) 的 AWS SDK
2. 为您在中创建的用户创建访问密钥创建 IAM 用户.
   a. 登录 AWS 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 在导航窗格中，选择 Users (用户)。
   c. 选择您在中创建的用户的名称。创建 IAM 用户.
   d. 选择 Security Credentials（安全凭证）选项卡。
   e. 选择创建访问密钥。然后，选择 Download .csv file (下载 .csv 文件)，将访问密钥 ID 和秘密访问密钥保存至计算机上的 CSV 文件中。将文件存储在安全位置。关闭此对话框后，您将无法再次访问该秘密访问密钥。下载 CSV 文件之后，选择Close.
3. 在本地系统上的 AWS 凭证配置文件中设置凭证，该配置文件位于：

- ~/.aws/credentials(在 Linux、macOS 或 Unix) 上。
- C:\Users\USERNAME\.aws\credentialsWindows 上的。

这些区域有：.aws在首次初始配置 AWS 实例之前，文件夹不存在。首次使用 CLI 配置凭据时，将创建此文件夹。有关 AWS 凭证的更多信息，请参阅。配置和凭证文件设置。

此文件应包含以下格式的行：

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

将访问密钥 ID 和秘密访问密钥替换你的 _access_key_id和你的 _secret_access_key.

4. 在 AWS 中设置默认 AWS 区域config本地系统上的文件，该文件位于：

- ~/.aws/config(在 Linux、macOS 或 Unix) 上。
- C:\Users\USERNAME\.aws\configWindows 上的。

这些区域有：.aws在首次初始配置 AWS 实例之前，文件夹不存在。首次使用 CLI 配置凭据时，将创建此文件夹。有关 AWS 凭证的更多信息，请参阅。配置和凭证文件设置。

此文件应包含以下行：

```
[default]
region = your_aws_region
```

用所需的 AWS 区域 (例如 “us-west-2”) 替换 AWS 区域你的 _aws_ 地区.

> **Note**
>
> 如果您未选择区域，则默认情况下使用 us-east-1。

下一个步骤

第 3 步：开始使用AWS CLI和AWSSDK API

第 3 步：开始使用AWS CLI和AWSSDK API

在你设置完AWS CLI和AWS要使用的 SDK 开发工具包，您可以构建使用 Amazon Textract 的应用程序。以下主题将向您展示如何开始使用 Amazon Textract。

- 使用 Amazon Textract 分析文档文本

设置 AWS CLI 示例的格式

针对 Linux 操作系统设置本指南中 AWS CLI 示例的格式。要将示例用于 Microsoft Windows，您需要更改 `--document` 参数的 JSON 格式，并将换行符从反斜杠 (\) 更改为插字号 (^)。有关 JSON 格式的更多信息，请参阅。为 AWS 命令行界面指定参数值.

使用同步操作处理文档

Amazon Textract 可以检测和分析单页文档中以 JPEG、PNG、PDF 和 TIFF 格式提供的文本。操作是同步的，近实时返回结果。有关文档的更多信息，请参阅 文本检测和文档分析响应对象。

本节介绍了如何使用 Amazon Textract 同步检测和分析单页文档中的文本。要检测和分析多页文档中的文本，或者异步检测 JPEG 和 PNG 文档，请参阅使用异步操作处理文档.

您可以将 Amazon Textract 同步操作用于以下目的：

- 文本检测 — 您可以使用DetectDocumentTextoperation. 有关更多信息，请参阅 检测文本。
- 文本分析 — 您可以使用AnalyzeDocumentoperation. 有关更多信息，请参阅 分析文档。
- 发票和收据分析 — 您可以使用 AnalyzeEsend 操作识别单页发票上检测到的文本或收据之间的财务关系。有关更多信息，请参阅 。分析发票和收据
- 身份证件分析 — 您可以分析美国政府签发的身份证件，并提取信息以及身份证件上的常见信息类型。有关更多信息，请参阅分析身份证件.

主题

- 调用 Amazon Textract 同步操作
- 使用 Amazon Textract 检测文档文本
- 使用 Amazon Textract 分析文档文本
- 使用 Amazon Textract 分析发票和收据
- 使用 Amazon Textract 分析身份文档

调用 Amazon Textract 同步操作

Amazon Textract 操作处理存储在本地文件系统上的文档图像或存储在 Amazon S3 存储桶中的文档图像。您可以通过使用Document输入参数。文档图像可以是 PNG、JPEG、PDF 或 TIFF 格式。同步操作的结果将立即返回，不会存储以供检索。

有关完整的示例，请参阅使用 Amazon Textract 检测文档文本.

请求

下面介绍了请求在 Amazon Textract 中的工作方式。

作为图像字节传递的文档

您可以将文档图像传递给 Amazon Textract 操作，方法是将图像作为 base64 编码的字节数组传递该图像。例如，从本地文件系统加载的文档图像。如果您使用的是，代码可能无需对文档文件字节进行编码。AWS开发工具包调用 Amazon Textract API 操作。

图像字节在`Bytes`字段`Document`输入参数。以下示例显示了传递图像字节的 Amazon Textract 操作的输入 JSON`Bytes`输入参数。

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

> **Note**
>
> 如果您使用AWS CLI，您无法将图像字节传递给 Amazon Textract 操作。相反，您必须引用存储在 Amazon S3 存储桶中的映像。

以下 Java 代码显示如何从本地文件系统加载图像和调用 Amazon Textract 操作。

```
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);
```

Amazon S3 存储桶中存储的文档

Amazon Textract 可以分析存储在 Amazon S3 存储桶中的文档图像。您可以使用S3Object字段Document输入参数。以下示例显示针对处理存储在 Amazon S3 存储桶中的文档的 Amazon Textract 操作的输入 JSON。

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

以下示例显示如何使用存储在 Amazon S3 存储桶中的图像调用 Amazon Textract 操作。

```
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);
```

响应

以下示例是调用到的 JSON 响应。DetectDocumentText. 有关更多信息，请参阅 检测文本。

```
{
{
  "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
          },
          {
            "X": 0.0,
            "Y": 1.0
          }
        ]
      },
      "Id": "ca4b9171-7109-4adb-a811-e09bbe4834dd",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "26085884-d005-4144-b4c2-4d83dc50739b",
            "ee9d01bc-d91c-401d-8c0a-eec76f5f7862",
            "404bb3d3-d7ab-4008-a195-5dec87a08664",
            "8ae1b4ba-67c1-4486-bd20-54f461886ce9",
            "47aab5ab-be2c-4c73-97c7-d0a45454e843",
            "dd06bb49-6a56-4ea7-beec-a2aa09835c3c",
            "8837153d-81b8-4031-a49f-83a3d81803c2",
            "5dae3b74-9e95-4b62-99b7-93b88fe70648",
            "4508da80-64d8-42a8-8846-cfafe6eab10c",
            "e87be7a9-5519-42e1-b18e-ae10e2d3ed13",
            "f04bb223-d075-41c3-b328-7354611c826b",
            "a234f0e8-67de-46f4-a7c7-0bbe8d5159ce",
            "61b20e27-ff8a-450a-a8b1-bc0259f82fd6",
```

```
            "445f4fdd-c77b-4a7b-a2fc-6ca07cfe9ed7",
            "359f3870-7183-43f5-b638-970f5cefe4d5",
            "b9deea0a-244c-4d54-b774-cf03fbaaa8b1",
            "e2a43881-f620-44f2-b067-500ce7dc8d4d",
            "41756974-64ef-432d-b4b2-34702505975a",
            "93d96d32-8b4a-4a98-9578-8b4df4f227a6",
            "bc907357-63d6-43c0-ab87-80d7e76d377e",
            "2d727ca7-3acb-4bb9-a564-5885c90e9325",
            "f32a5989-cbfb-41e6-b0fc-ce1c77c014bd",
            "e0ba06d0-dbb6-4962-8047-8cac3adfe45a",
            "b6ed204d-ae01-4b75-bb91-c85d4147a37e",
            "ac4b9ee0-c9b2-4239-a741-5753e5282033",
            "ebc18885-48d7-45b8-90e3-d172b4357802",
            "babf6360-789e-49c1-9c78-0784acc14a0c"
          ]
        }
      ]
    },
    {
      "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",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.19878505170345306,
          "Height": 0.03754019737243652,
          "Left": 0.03988289833068848,
          "Top": 0.14050349593162537
        },
        "Polygon": [
          {
            "X": 0.03988289833068848,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.1780436933040619
          },
          {
            "X": 0.03988289833068848,
            "Y": 0.1780436933040619
          }
        ]
```

```
      },
      "Id": "ee9d01bc-d91c-401d-8c0a-eec76f5f7862",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "efe3fc6d-becb-4520-80ee-49a329386aee",
            "c2260852-6cfd-4a71-9fc6-62b2f9b02355"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.88693237304688,
      "Text": "Full Name: Jane Doe",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.16733919084072113,
          "Height": 0.031106337904930115,
          "Left": 0.03899926319718361,
          "Top": 0.21361036598682404
        },
        "Polygon": [
          {
            "X": 0.03899926319718361,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.24471670389175415
          },
          {
            "X": 0.03899926319718361,
            "Y": 0.24471670389175415
          }
        ]
      },
      "Id": "404bb3d3-d7ab-4008-a195-5dec87a08664",
      "Relationships": [
```

```
        {
          "Type": "CHILD",
          "Ids": [
            "e94eb587-9545-4215-b0fc-8e8cb1172958",
            "090aeba5-8428-4b7a-a54b-7a95a774120e",
            "64ff0abb-736b-4a6b-aa8d-ad2c0086ae1d",
            "565ffc30-89d6-4295-b8c6-d22b4ed76584"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9206314086914,
      "Text": "Phone Number: 555-0100",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3115004599094391,
          "Height": 0.047169625759124756,
          "Left": 0.03604753687977791,
          "Top": 0.2812676727771759
        },
        "Polygon": [
          {
            "X": 0.03604753687977791,
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.32843729853630066
          },
          {
            "X": 0.03604753687977791,
            "Y": 0.32843729853630066
          }
        ]
      },
      "Id": "8ae1b4ba-67c1-4486-bd20-54f461886ce9",
      "Relationships": [
        {
```

```
          "Type": "CHILD",
          "Ids": [
            "d782f847-225b-4a1b-b52d-f252f8221b1f",
            "fa69c5cd-c80d-4fac-81df-569edae8d259",
            "d4bbc0f1-ae02-41cf-a26f-8a1e899968cc"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.48902893066406,
      "Text": "Home Address: 123 Any Street, Any Town. USA",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.7431139945983887,
          "Height": 0.09577702730894089,
          "Left": 0.03359385207295418,
          "Top": 0.3258342146873474
        },
        "Polygon": [
          {
            "X": 0.03359385207295418,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.4216112196445465
          },
          {
            "X": 0.03359385207295418,
            "Y": 0.4216112196445465
          }
        ]
      },
      "Id": "47aab5ab-be2c-4c73-97c7-d0a45454e843",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
```

```
            "acfbed90-4a00-42c6-8a90-d0a0756eea36",
            "046c8a40-bb0e-4718-9c71-954d3630e1dd",
            "82b838bc-4591-4287-8dea-60c94a4925e4",
            "5cdcde7a-f5a6-4231-a941-b6396e42e7ba",
            "beafd497-185f-487e-b070-db4df5803e94",
            "ef1b77fb-8ba6-41fe-ba53-dce039af22ed",
            "7b555310-e7f8-4cd2-bb3d-dcec37f3d90e",
            "b479c24d-448d-40ef-9ed5-36a6ef08e5c7"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.89382934570312,
      "Text": "Mailing Address: same as above",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.26575741171836853,
          "Height": 0.039571404457092285,
          "Left": 0.03068041242659092,
          "Top": 0.43351811170578003
        },
        "Polygon": [
          {
            "X": 0.03068041242659092,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4730895161628723
          },
          {
            "X": 0.03068041242659092,
            "Y": 0.4730895161628723
          }
        ]
      },
      "Id": "dd06bb49-6a56-4ea7-beec-a2aa09835c3c",
      "Relationships": [
```

```
        {
          "Type": "CHILD",
          "Ids": [
            "d7261cdc-6ac5-4711-903c-4598fe94952d",
            "287f80c3-6db2-4dd7-90ec-5f017c80aa31",
            "ce31c3ad-b51e-4068-be64-5fc9794bc1bc",
            "e96eb92c-6774-4d6f-8f4a-68a7618d4c66",
            "88b85c05-427a-4d4f-8cc4-3667234e8364"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 94.67343139648438,
      "Text": "Previous Employment History",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3309842050075531,
          "Height": 0.051920413970947266,
          "Left": 0.3194798231124878,
          "Top": 0.5172380208969116
        },
        "Polygon": [
          {
            "X": 0.3194798231124878,
            "Y": 0.5172380208969116
          },
          {
            "X": 0.6504639983177185,
            "Y": 0.5172380208969116
          },
          {
            "X": 0.6504639983177185,
            "Y": 0.5691584348678589
          },
          {
            "X": 0.3194798231124878,
            "Y": 0.5691584348678589
          }
        ]
      },
      "Id": "8837153d-81b8-4031-a49f-83a3d81803c2",
      "Relationships": [
```

```
        {
          "Type": "CHILD",
          "Ids": [
            "8b324501-bf38-4ce9-9777-6514b7ade760",
            "b0cea99a-5045-464d-ac8a-a63ab0470995",
            "b92a6ee5-ca59-44dc-9c47-534c133b11e7"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.66949462890625,
      "Text": "Start Date",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08310240507125854,
          "Height": 0.030944595113396645,
          "Left": 0.034429505467414856,
          "Top": 0.6123942136764526
        },
        "Polygon": [
          {
            "X": 0.034429505467414856,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319030880928,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319030880928,
            "Y": 0.6433387994766235
          },
          {
            "X": 0.034429505467414856,
            "Y": 0.6433387994766235
          }
        ]
      },
      "Id": "5dae3b74-9e95-4b62-99b7-93b88fe70648",
      "Relationships": [
        {
          "Type": "CHILD",
```

```
          "Ids": [
            "ffe8b8e0-df59-4ac5-9aba-6b54b7c51b45",
            "91e582cd-9871-4e9c-93cc-848baa426338"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.86717224121094,
      "Text": "End Date",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07581500709056854,
          "Height": 0.03223184868693352,
          "Left": 0.14846202731132507,
          "Top": 0.6120467782020569
        },
        "Polygon": [
          {
            "X": 0.14846202731132507,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6442786455154419
          },
          {
            "X": 0.14846202731132507,
            "Y": 0.6442786455154419
          }
        ]
      },
      "Id": "4508da80-64d8-42a8-8846-cfafe6eab10c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "7c97b56b-699f-49b0-93f4-98e6d90b107c",
            "7af04e27-0c15-447e-a569-b30edb99a133"
```

```
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9539794921875,
      "Text": "Employer Name",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1347292959690094,
          "Height": 0.0392492413520813,
          "Left": 0.2647075653076172,
          "Top": 0.6140711903572083
        },
        "Polygon": [
          {
            "X": 0.2647075653076172,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.3994368314743042,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.3994368314743042,
            "Y": 0.6533204317092896
          },
          {
            "X": 0.2647075653076172,
            "Y": 0.6533204317092896
          }
        ]
      },
      "Id": "e87be7a9-5519-42e1-b18e-ae10e2d3ed13",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a9bfeb55-75cd-47cd-b953-728e602a3564",
            "9f0f9c06-d02c-4b07-bb39-7ade70be2c1b"
          ]
        }
      ]
```

```
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.35584259033203,
      "Text": "Position Held",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11393272876739502,
          "Height": 0.03415105864405632,
          "Left": 0.49973347783088684,
          "Top": 0.614840030670166
        },
        "Polygon": [
          {
            "X": 0.49973347783088684,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136661767959595,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136661767959595,
            "Y": 0.6489911079406738
          },
          {
            "X": 0.49973347783088684,
            "Y": 0.6489911079406738
          }
        ]
      },
      "Id": "f04bb223-d075-41c3-b328-7354611c826b",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "6d5edf02-845c-40e0-9514-e56d0d652ae0",
            "3297ab59-b237-45fb-ae60-a108f0c95ac2"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
```

```
      "Confidence": 99.9817886352539,
      "Text": "Reason for leaving",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.16511960327625275,
          "Height": 0.04062700271606445,
          "Left": 0.7430596351623535,
          "Top": 0.6116235852241516
        },
        "Polygon": [
          {
            "X": 0.7430596351623535,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6522505879402161
          },
          {
            "X": 0.7430596351623535,
            "Y": 0.6522505879402161
          }
        ]
      },
      "Id": "a234f0e8-67de-46f4-a7c7-0bbe8d5159ce",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "f4b8cf26-d2da-4a76-8345-69562de3cc11",
            "386d4a63-1194-4c0e-a18d-4d074a0b1f93",
            "a8622541-1896-4d54-8d10-7da2c800ec5c"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.77413177490234,
      "Text": "1/15/2009",
```

```
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08799663186073303,
          "Height": 0.03832906484603882,
          "Left": 0.03175082430243492,
          "Top": 0.691371738910675
        },
        "Polygon": [
          {
            "X": 0.03175082430243492,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.7297008037567139
          },
          {
            "X": 0.03175082430243492,
            "Y": 0.7297008037567139
          }
        ]
      },
      "Id": "61b20e27-ff8a-450a-a8b1-bc0259f82fd6",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "da7a6482-0964-49a4-bc7d-56942ff3b4e1"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.72286224365234,
      "Text": "6/30/2011",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08843101561069489,
          "Height": 0.03991425037384033,
```

```
          "Left": 0.14642837643623352,
          "Top": 0.6919752955436707
        },
        "Polygon": [
          {
            "X": 0.14642837643623352,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.731889545917511
          },
          {
            "X": 0.14642837643623352,
            "Y": 0.731889545917511
          }
        ]
      },
      "Id": "445f4fdd-c77b-4a7b-a2fc-6ca07cfe9ed7",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "5a8da66a-ecce-4ee9-a765-a46d6cdc6cde"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.86936950683594,
      "Text": "Any Company",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11800950765609741,
          "Height": 0.03943679481744766,
          "Left": 0.2626699209213257,
          "Top": 0.6972727179527283
        },
        "Polygon": [
```

```
          {
            "X": 0.2626699209213257,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.3806794285774231,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.3806794285774231,
            "Y": 0.736709475517273
          },
          {
            "X": 0.2626699209213257,
            "Y": 0.736709475517273
          }
        ]
      },
      "Id": "359f3870-7183-43f5-b638-970f5cefe4d5",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "77749c2b-aa7f-450e-8dd2-62bcaf253ba2",
            "713bad19-158d-4e3e-b01f-f5707ddb04e5"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.582275390625,
      "Text": "Assistant baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.13280922174453735,
          "Height": 0.032666124403476715,
          "Left": 0.49814170598983765,
          "Top": 0.699238657951355
        },
        "Polygon": [
          {
            "X": 0.49814170598983765,
            "Y": 0.699238657951355
```

```
          },
          {
            "X": 0.630950927734375,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.7319048047065735
          },
          {
            "X": 0.49814170598983765,
            "Y": 0.7319048047065735
          }
        ]
      },
      "Id": "b9deea0a-244c-4d54-b774-cf03fbaaa8b1",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "989944f9-f684-4714-87d8-9ad9a321d65c",
            "ae82e2aa-1601-4e0c-8340-1db7ad0c9a31"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.96180725097656,
      "Text": "relocated",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08668994903564453,
          "Height": 0.033302485942840576,
          "Left": 0.7426905632019043,
          "Top": 0.6974037289619446
        },
        "Polygon": [
          {
            "X": 0.7426905632019043,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
```

```
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.7307062149047852
          },
          {
            "X": 0.7426905632019043,
            "Y": 0.7307062149047852
          }
        ]
      },
      "Id": "e2a43881-f620-44f2-b067-500ce7dc8d4d",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a9cf9a8c-fdaa-413e-9346-5a28a98aebdb"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98190307617188,
      "Text": "7/1/2011",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09747002273797989,
          "Height": 0.07067441940307617,
          "Left": 0.028500309213995934,
          "Top": 0.7745237946510315
        },
        "Polygon": [
          {
            "X": 0.028500309213995934,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
```

```
            "Y": 0.8451982140541077
          },
          {
            "X": 0.028500309213995934,
            "Y": 0.8451982140541077
          }
        ]
      },
      "Id": "41756974-64ef-432d-b4b2-34702505975a",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0f711065-1872-442a-ba6d-8fababaa452a"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98418426513672,
      "Text": "8/10/2013",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10664612054824829,
          "Height": 0.06439518928527832,
          "Left": 0.14159755408763885,
          "Top": 0.7791688442230225
        },
        "Polygon": [
          {
            "X": 0.14159755408763885,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.8435640335083008
          },
          {
            "X": 0.14159755408763885,
```

```
            "Y": 0.8435640335083008
          }
        ]
      },
      "Id": "93d96d32-8b4a-4a98-9578-8b4df4f227a6",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a92d8eef-db28-45ba-801a-5da0f589d277"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98075866699219,
      "Text": "Example Corp.",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.2114926278591156,
          "Height": 0.058415766805410385,
          "Left": 0.26764172315597534,
          "Top": 0.794414758682251
        },
        "Polygon": [
          {
            "X": 0.26764172315597534,
            "Y": 0.794414758682251
          },
          {
            "X": 0.47913435101509094,
            "Y": 0.794414758682251
          },
          {
            "X": 0.47913435101509094,
            "Y": 0.8528305292129517
          },
          {
            "X": 0.26764172315597534,
            "Y": 0.8528305292129517
          }
        ]
      },
```

```
      "Id": "bc907357-63d6-43c0-ab87-80d7e76d377e",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "d6962efb-34ab-4ffb-9f2f-5f263e813558",
            "1876c8ea-d3e8-4c39-870e-47512b3b5080"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.91166687011719,
      "Text": "Baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09931200742721558,
          "Height": 0.06008726358413696,
          "Left": 0.5098910331726074,
          "Top": 0.787897527217865
        },
        "Polygon": [
          {
            "X": 0.5098910331726074,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.847984790802002
          },
          {
            "X": 0.5098910331726074,
            "Y": 0.847984790802002
          }
        ]
      },
      "Id": "2d727ca7-3acb-4bb9-a564-5885c90e9325",
      "Relationships": [
        {
```

```
          "Type": "CHILD",
          "Ids": [
            "00adeaef-ed57-44eb-b8a9-503575236d62"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.93852233886719,
      "Text": "better opp.",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18919607996940613,
          "Height": 0.06994765996932983,
          "Left": 0.7428008317947388,
          "Top": 0.7928366661071777
        },
        "Polygon": [
          {
            "X": 0.7428008317947388,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.9319968819618225,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.9319968819618225,
            "Y": 0.8627843260765076
          },
          {
            "X": 0.7428008317947388,
            "Y": 0.8627843260765076
          }
        ]
      },
      "Id": "f32a5989-cbfb-41e6-b0fc-ce1c77c014bd",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "c0fc9a58-7a4b-4f69-bafd-2cff32be2665",
            "bf6dc8ee-2fb3-4b6c-aee4-31e96912a2d8"
```

```
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.92573547363281,
      "Text": "8/15/2013",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10257463902235031,
          "Height": 0.05412459373474121,
          "Left": 0.027909137308597565,
          "Top": 0.8608770370483398
        },
        "Polygon": [
          {
            "X": 0.027909137308597565,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.915001630783081
          },
          {
            "X": 0.027909137308597565,
            "Y": 0.915001630783081
          }
        ]
      },
      "Id": "e0ba06d0-dbb6-4962-8047-8cac3adfe45a",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "5384f860-f857-4a94-9438-9dfa20eed1c6"
          ]
        }
      ]
    },
```

```
    {
      "BlockType": "LINE",
      "Confidence": 99.99625396728516,
      "Text": "Present",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09982697665691376,
          "Height": 0.06888341903686523,
          "Left": 0.1420602649450302,
          "Top": 0.8511748909950256
        },
        "Polygon": [
          {
            "X": 0.1420602649450302,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.9200583100318909
          },
          {
            "X": 0.1420602649450302,
            "Y": 0.9200583100318909
          }
        ]
      },
      "Id": "b6ed204d-ae01-4b75-bb91-c85d4147a37e",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0bb96ed6-b2e6-4da4-90b3-b85561bbd89d"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9826431274414,
      "Text": "AnyCompany",
```

```
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18611276149749756,
          "Height": 0.08581399917602539,
          "Left": 0.2615866959095001,
          "Top": 0.869536280632019
        },
        "Polygon": [
          {
            "X": 0.2615866959095001,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994574069977,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994574069977,
            "Y": 0.9553502798080444
          },
          {
            "X": 0.2615866959095001,
            "Y": 0.9553502798080444
          }
        ]
      },
      "Id": "ac4b9ee0-c9b2-4239-a741-5753e5282033",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "25343360-d906-440a-88b7-92eb89e95949"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.99549102783203,
      "Text": "head baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1937451809644699,
          "Height": 0.056156039237976074,
```

```
          "Left": 0.49359121918678284,
          "Top": 0.8702592849731445
        },
        "Polygon": [
          {
            "X": 0.49359121918678284,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873363852500916,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873363852500916,
            "Y": 0.9264153242111206
          },
          {
            "X": 0.49359121918678284,
            "Y": 0.9264153242111206
          }
        ]
      },
      "Id": "ebc18885-48d7-45b8-90e3-d172b4357802",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0ef3c194-8322-4575-94f1-82819ee57e3a",
            "d296acd9-3e9a-4985-95f8-f863614f2c46"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98360443115234,
      "Text": "N/A, current",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.22544169425964355,
          "Height": 0.06588292121887207,
          "Left": 0.7411766648292542,
          "Top": 0.8722732067108154
        },
```

```
        "Polygon": [
          {
            "X": 0.7411766648292542,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.9666183590888977,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.9666183590888977,
            "Y": 0.9381561279296875
          },
          {
            "X": 0.7411766648292542,
            "Y": 0.9381561279296875
          }
        ]
      },
      "Id": "babf6360-789e-49c1-9c78-0784acc14a0c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "195cfb5b-ae06-4203-8520-4e4b0a73b5ce",
            "549ef3f9-3a13-4b77-bc25-fb2e0996839a"
          ]
        }
      ]
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94815826416016,
      "Text": "Employment",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.17462396621704102,
          "Height": 0.06266549974679947,
          "Left": 0.29548385739326477,
          "Top": 0.03389188274741173
        },
        "Polygon": [
          {
```

```
            "X": 0.29548385739326477,
            "Y": 0.03389188274741173
          },
          {
            "X": 0.4701078236103058,
            "Y": 0.03389188274741173
          },
          {
            "X": 0.4701078236103058,
            "Y": 0.0965573862195015
          },
          {
            "X": 0.29548385739326477,
            "Y": 0.0965573862195015
          }
        ]
      },
      "Id": "ed48dacc-d089-498f-8e93-1cee1e5f39f3"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92706298828125,
      "Text": "Application",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.15933875739574432,
          "Height": 0.062391020357608795,
          "Left": 0.47528234124183655,
          "Top": 0.027493247762322426
        },
        "Polygon": [
          {
            "X": 0.47528234124183655,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346211433410645,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346211433410645,
            "Y": 0.08988427370786667
          },
```

```
          {
            "X": 0.47528234124183655,
            "Y": 0.08988427370786667
          }
        ]
      },
      "Id": "ac7370f3-cbb7-4cd9-a8f9-bdcb2252caaf"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9821548461914,
      "Text": "Application",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09610454738140106,
          "Height": 0.03656719997525215,
          "Left": 0.03988289833068848,
          "Top": 0.14147649705410004
        },
        "Polygon": [
          {
            "X": 0.03988289833068848,
            "Y": 0.14147649705410004
          },
          {
            "X": 0.13598744571208954,
            "Y": 0.14147649705410004
          },
          {
            "X": 0.13598744571208954,
            "Y": 0.1780436933040619
          },
          {
            "X": 0.03988289833068848,
            "Y": 0.1780436933040619
          }
        ]
      },
      "Id": "efe3fc6d-becb-4520-80ee-49a329386aee"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.84278106689453,
```

```
      "Text": "Information",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10029315203428268,
          "Height": 0.03209415823221207,
          "Left": 0.13837480545043945,
          "Top": 0.14050349593162537
        },
        "Polygon": [
          {
            "X": 0.13837480545043945,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.17259766161441803
          },
          {
            "X": 0.13837480545043945,
            "Y": 0.17259766161441803
          }
        ]
      },
      "Id": "c2260852-6cfd-4a71-9fc6-62b2f9b02355"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.83993530273438,
      "Text": "Full",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03039788082242012,
          "Height": 0.031106330454349518,
          "Left": 0.03899926319718361,
          "Top": 0.21361036598682404
        },
        "Polygon": [
          {
```

```
            "X": 0.03899926319718361,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.06939714401960373,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.06939714401960373,
            "Y": 0.24471670389175415
          },
          {
            "X": 0.03899926319718361,
            "Y": 0.24471670389175415
          }
        ]
      },
      "Id": "e94eb587-9545-4215-b0fc-8e8cb1172958"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.93611907958984,
      "Text": "Name:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05555811896920204,
          "Height": 0.030184319242835045,
          "Left": 0.07123806327581406,
          "Top": 0.2137702852487564
        },
        "Polygon": [
          {
            "X": 0.07123806327581406,
            "Y": 0.2137702852487564
          },
          {
            "X": 0.1267961859703064,
            "Y": 0.2137702852487564
          },
          {
            "X": 0.1267961859703064,
            "Y": 0.2439546138048172
          },
```

```
        {
          "X": 0.07123806327581406,
          "Y": 0.2439546138048172
        }
      ]
    },
    "Id": "090aeba5-8428-4b7a-a54b-7a95a774120e"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.91043853759766,
    "Text": "Jane",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.03905024006962776,
        "Height": 0.02941947989165783,
        "Left": 0.12933772802352905,
        "Top": 0.214289128780365
      },
      "Polygon": [
        {
          "X": 0.12933772802352905,
          "Y": 0.214289128780365
        },
        {
          "X": 0.16838796436786652,
          "Y": 0.214289128780365
        },
        {
          "X": 0.16838796436786652,
          "Y": 0.24370861053466797
        },
        {
          "X": 0.12933772802352905,
          "Y": 0.24370861053466797
        }
      ]
    },
    "Id": "64ff0abb-736b-4a6b-aa8d-ad2c0086ae1d"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.86123657226562,
```

```
      "Text": "Doe",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.035229459404945374,
          "Height": 0.030427640303969383,
          "Left": 0.17110899090766907,
          "Top": 0.21377210319042206
        },
        "Polygon": [
          {
            "X": 0.17110899090766907,
            "Y": 0.21377210319042206
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.21377210319042206
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.244199737906456
          },
          {
            "X": 0.17110899090766907,
            "Y": 0.244199737906456
          }
        ]
      },
      "Id": "565ffc30-89d6-4295-b8c6-d22b4ed76584"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92633056640625,
      "Text": "Phone",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.052783288061618805,
          "Height": 0.03104414977133274,
          "Left": 0.03604753687977791,
          "Top": 0.28701552748680115
        },
        "Polygon": [
          {
```

```
            "X": 0.03604753687977791,
            "Y": 0.28701552748680115
          },
          {
            "X": 0.08883082121610641,
            "Y": 0.28701552748680115
          },
          {
            "X": 0.08883082121610641,
            "Y": 0.31805968284606934
          },
          {
            "X": 0.03604753687977791,
            "Y": 0.31805968284606934
          }
        ]
      },
      "Id": "d782f847-225b-4a1b-b52d-f252f8221b1f"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86275482177734,
      "Text": "Number:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07424934208393097,
          "Height": 0.030300479382276535,
          "Left": 0.0915418416261673,
          "Top": 0.28639692068099976
        },
        "Polygon": [
          {
            "X": 0.0915418416261673,
            "Y": 0.28639692068099976
          },
          {
            "X": 0.16579118371009827,
            "Y": 0.28639692068099976
          },
          {
            "X": 0.16579118371009827,
            "Y": 0.3166973888874054
          },
```

```
        {
          "X": 0.0915418416261673,
          "Y": 0.3166973888874054
        }
      ]
    },
    "Id": "fa69c5cd-c80d-4fac-81df-569edae8d259"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.97282409667969,
    "Text": "555-0100",
    "TextType": "HANDWRITING",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.17021971940994263,
        "Height": 0.047169629484415054,
        "Left": 0.17732827365398407,
        "Top": 0.2812676727771759
      },
      "Polygon": [
        {
          "X": 0.17732827365398407,
          "Y": 0.2812676727771759
        },
        {
          "X": 0.3475480079650879,
          "Y": 0.2812676727771759
        },
        {
          "X": 0.3475480079650879,
          "Y": 0.32843729853630066
        },
        {
          "X": 0.17732827365398407,
          "Y": 0.32843729853630066
        }
      ]
    },
    "Id": "d4bbc0f1-ae02-41cf-a26f-8a1e899968cc"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.66238403320312,
```

```
      "Text": "Home",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.049357783049345016,
          "Height": 0.03134990110993385,
          "Left": 0.03359385207295418,
          "Top": 0.36172014474868774
        },
        "Polygon": [
          {
            "X": 0.03359385207295418,
            "Y": 0.36172014474868774
          },
          {
            "X": 0.0829516351222992,
            "Y": 0.36172014474868774
          },
          {
            "X": 0.0829516351222992,
            "Y": 0.3930700421333313
          },
          {
            "X": 0.03359385207295418,
            "Y": 0.3930700421333313
          }
        ]
      },
      "Id": "acfbed90-4a00-42c6-8a90-d0a0756eea36"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.6871109008789,
      "Text": "Address:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07411003112792969,
          "Height": 0.0314042791724205,
          "Left": 0.08516156673431396,
          "Top": 0.3600046932697296
        },
        "Polygon": [
          {
```

```
            "X": 0.08516156673431396,
            "Y": 0.3600046932697296
          },
          {
            "X": 0.15927159786224365,
            "Y": 0.3600046932697296
          },
          {
            "X": 0.15927159786224365,
            "Y": 0.3914089798927307
          },
          {
            "X": 0.08516156673431396,
            "Y": 0.3914089798927307
          }
        ]
      },
      "Id": "046c8a40-bb0e-4718-9c71-954d3630e1dd"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.93781280517578,
      "Text": "123",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05761868134140968,
          "Height": 0.05008566007018089,
          "Left": 0.1750781387090683,
          "Top": 0.35484206676483154
        },
        "Polygon": [
          {
            "X": 0.1750781387090683,
            "Y": 0.35484206676483154
          },
          {
            "X": 0.23269681632518768,
            "Y": 0.35484206676483154
          },
          {
            "X": 0.23269681632518768,
            "Y": 0.40492773056030273
          },
```

```
        {
          "X": 0.1750781387090683,
          "Y": 0.40492773056030273
        }
      ]
    },
    "Id": "82b838bc-4591-4287-8dea-60c94a4925e4"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.96530151367188,
    "Text": "Any",
    "TextType": "HANDWRITING",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.06814215332269669,
        "Height": 0.06354366987943649,
        "Left": 0.2550157308578491,
        "Top": 0.35471394658088684
      },
      "Polygon": [
        {
          "X": 0.2550157308578491,
          "Y": 0.35471394658088684
        },
        {
          "X": 0.3231579065322876,
          "Y": 0.35471394658088684
        },
        {
          "X": 0.3231579065322876,
          "Y": 0.41825762391090393
        },
        {
          "X": 0.2550157308578491,
          "Y": 0.41825762391090393
        }
      ]
    },
    "Id": "5cdcde7a-f5a6-4231-a941-b6396e42e7ba"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.87527465820312,
```

```
      "Text": "Street,",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12156613171100616,
          "Height": 0.05449587106704712,
          "Left": 0.3357025980949402,
          "Top": 0.3550415635108948
        },
        "Polygon": [
          {
            "X": 0.3357025980949402,
            "Y": 0.3550415635108948
          },
          {
            "X": 0.45726871490478516,
            "Y": 0.3550415635108948
          },
          {
            "X": 0.45726871490478516,
            "Y": 0.4095374345779419
          },
          {
            "X": 0.3357025980949402,
            "Y": 0.4095374345779419
          }
        ]
      },
      "Id": "beafd497-185f-487e-b070-db4df5803e94"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99514770507812,
      "Text": "Any",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07748188823461533,
          "Height": 0.07339789718389511,
          "Left": 0.47723668813705444,
          "Top": 0.3482133150100708
        },
        "Polygon": [
          {
```

```
            "X": 0.47723668813705444,
            "Y": 0.3482133150100708
          },
          {
            "X": 0.554718554019928,
            "Y": 0.3482133150100708
          },
          {
            "X": 0.554718554019928,
            "Y": 0.4216112196445465
          },
          {
            "X": 0.47723668813705444,
            "Y": 0.4216112196445465
          }
        ]
      },
      "Id": "ef1b77fb-8ba6-41fe-ba53-dce039af22ed"
    },
    {
      "BlockType": "WORD",
      "Confidence": 96.80656433105469,
      "Text": "Town.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11213835328817368,
          "Height": 0.057233039289712906,
          "Left": 0.5563329458236694,
          "Top": 0.3331930637359619
        },
        "Polygon": [
          {
            "X": 0.5563329458236694,
            "Y": 0.3331930637359619
          },
          {
            "X": 0.6684713363647461,
            "Y": 0.3331930637359619
          },
          {
            "X": 0.6684713363647461,
            "Y": 0.3904260993003845
          },
```

```
        {
          "X": 0.5563329458236694,
          "Y": 0.3904260993003845
        }
      ]
    },
    "Id": "7b555310-e7f8-4cd2-bb3d-dcec37f3d90e"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.98260498046875,
    "Text": "USA",
    "TextType": "HANDWRITING",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.08771833777427673,
        "Height": 0.05706935003399849,
        "Left": 0.6889894604682922,
        "Top": 0.3258342146873474
      },
      "Polygon": [
        {
          "X": 0.6889894604682922,
          "Y": 0.3258342146873474
        },
        {
          "X": 0.7767078280448914,
          "Y": 0.3258342146873474
        },
        {
          "X": 0.7767078280448914,
          "Y": 0.3829035460948944
        },
        {
          "X": 0.6889894604682922,
          "Y": 0.3829035460948944
        }
      ]
    },
    "Id": "b479c24d-448d-40ef-9ed5-36a6ef08e5c7"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.9583969116211,
```

```
      "Text": "Mailing",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06291338801383972,
          "Height": 0.03957144916057587,
          "Left": 0.03068041242659092,
          "Top": 0.43351811170578003
        },
        "Polygon": [
          {
            "X": 0.03068041242659092,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.09359379857778549,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.09359379857778549,
            "Y": 0.4730895459651947
          },
          {
            "X": 0.03068041242659092,
            "Y": 0.4730895459651947
          }
        ]
      },
      "Id": "d7261cdc-6ac5-4711-903c-4598fe94952d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.87476348876953,
      "Text": "Address:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07364854216575623,
          "Height": 0.03147412836551666,
          "Left": 0.0954652726650238,
          "Top": 0.43450701236724854
        },
        "Polygon": [
          {
```

```
            "X": 0.0954652726650238,
            "Y": 0.43450701236724854
          },
          {
            "X": 0.16911381483078003,
            "Y": 0.43450701236724854
          },
          {
            "X": 0.16911381483078003,
            "Y": 0.465981125831604
          },
          {
            "X": 0.0954652726650238,
            "Y": 0.465981125831604
          }
        ]
      },
      "Id": "287f80c3-6db2-4dd7-90ec-5f017c80aa31"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94071960449219,
      "Text": "same",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.04640670120716095,
          "Height": 0.026415130123496056,
          "Left": 0.17156922817230225,
          "Top": 0.44010937213897705
        },
        "Polygon": [
          {
            "X": 0.17156922817230225,
            "Y": 0.44010937213897705
          },
          {
            "X": 0.2179759293794632,
            "Y": 0.44010937213897705
          },
          {
            "X": 0.2179759293794632,
            "Y": 0.46652451157569885
          },
```

```
        {
          "X": 0.17156922817230225,
          "Y": 0.46652451157569885
        }
      ]
    },
    "Id": "ce31c3ad-b51e-4068-be64-5fc9794bc1bc"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.76510620117188,
    "Text": "as",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.02041218988597393,
        "Height": 0.025104399770498276,
        "Left": 0.2207803726196289,
        "Top": 0.44124215841293335
      },
      "Polygon": [
        {
          "X": 0.2207803726196289,
          "Y": 0.44124215841293335
        },
        {
          "X": 0.24119256436824799,
          "Y": 0.44124215841293335
        },
        {
          "X": 0.24119256436824799,
          "Y": 0.4663465619087219
        },
        {
          "X": 0.2207803726196289,
          "Y": 0.4663465619087219
        }
      ]
    },
    "Id": "e96eb92c-6774-4d6f-8f4a-68a7618d4c66"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.9301528930664,
```

```
      "Text": "above",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05268359184265137,
          "Height": 0.03216424956917763,
          "Left": 0.24375422298908234,
          "Top": 0.4354657828807831
        },
        "Polygon": [
          {
            "X": 0.24375422298908234,
            "Y": 0.4354657828807831
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4354657828807831
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4676300287246704
          },
          {
            "X": 0.24375422298908234,
            "Y": 0.4676300287246704
          }
        ]
      },
      "Id": "88b85c05-427a-4d4f-8cc4-3667234e8364"
    },
    {
      "BlockType": "WORD",
      "Confidence": 85.3905029296875,
      "Text": "Previous",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09860499948263168,
          "Height": 0.04000622034072876,
          "Left": 0.3194798231124878,
          "Top": 0.5194430351257324
        },
        "Polygon": [
          {
```

```
            "X": 0.3194798231124878,
            "Y": 0.5194430351257324
          },
          {
            "X": 0.4180848002433777,
            "Y": 0.5194430351257324
          },
          {
            "X": 0.4180848002433777,
            "Y": 0.5594492554664612
          },
          {
            "X": 0.3194798231124878,
            "Y": 0.5594492554664612
          }
        ]
      },
      "Id": "8b324501-bf38-4ce9-9777-6514b7ade760"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.14524841308594,
      "Text": "Employment",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.14039960503578186,
          "Height": 0.04645847901701927,
          "Left": 0.4214291274547577,
          "Top": 0.5219109654426575
        },
        "Polygon": [
          {
            "X": 0.4214291274547577,
            "Y": 0.5219109654426575
          },
          {
            "X": 0.5618287324905396,
            "Y": 0.5219109654426575
          },
          {
            "X": 0.5618287324905396,
            "Y": 0.568369448184967
          },
```

```
        {
          "X": 0.4214291274547577,
          "Y": 0.568369448184967
        }
      ]
    },
    "Id": "b0cea99a-5045-464d-ac8a-a63ab0470995"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.48454284667969,
    "Text": "History",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.08361124992370605,
        "Height": 0.05192042887210846,
        "Left": 0.5668527483940125,
        "Top": 0.5172380208969116
      },
      "Polygon": [
        {
          "X": 0.5668527483940125,
          "Y": 0.5172380208969116
        },
        {
          "X": 0.6504639983177185,
          "Y": 0.5172380208969116
        },
        {
          "X": 0.6504639983177185,
          "Y": 0.5691584348678589
        },
        {
          "X": 0.5668527483940125,
          "Y": 0.5691584348678589
        }
      ]
    },
    "Id": "b92a6ee5-ca59-44dc-9c47-534c133b11e7"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.78699493408203,
```

```
      "Text": "Start",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.041341401636600494,
          "Height": 0.030926469713449478,
          "Left": 0.034429505467414856,
          "Top": 0.6124123334884644
        },
        "Polygon": [
          {
            "X": 0.034429505467414856,
            "Y": 0.6124123334884644
          },
          {
            "X": 0.07577090710401535,
            "Y": 0.6124123334884644
          },
          {
            "X": 0.07577090710401535,
            "Y": 0.6433387994766235
          },
          {
            "X": 0.034429505467414856,
            "Y": 0.6433387994766235
          }
        ]
      },
      "Id": "ffe8b8e0-df59-4ac5-9aba-6b54b7c51b45"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.55198669433594,
      "Text": "Date",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03923053666949272,
          "Height": 0.03072454035282135,
          "Left": 0.07830137014389038,
          "Top": 0.6123942136764526
        },
        "Polygon": [
          {
```

```
            "X": 0.07830137014389038,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319105386734,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319105386734,
            "Y": 0.6431187391281128
          },
          {
            "X": 0.07830137014389038,
            "Y": 0.6431187391281128
          }
        ]
      },
      "Id": "91e582cd-9871-4e9c-93cc-848baa426338"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8897705078125,
      "Text": "End",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03212086856365204,
          "Height": 0.03193363919854164,
          "Left": 0.14846202731132507,
          "Top": 0.6120467782020569
        },
        "Polygon": [
          {
            "X": 0.14846202731132507,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.1805828958749771,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.1805828958749771,
            "Y": 0.6439804434776306
          },
```

```
          {
            "X": 0.14846202731132507,
            "Y": 0.6439804434776306
          }
        ]
      },
      "Id": "7c97b56b-699f-49b0-93f4-98e6d90b107c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8445816040039,
      "Text": "Date",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03987143933773041,
          "Height": 0.03142518177628517,
          "Left": 0.1844055950641632,
          "Top": 0.612853467464447
        },
        "Polygon": [
          {
            "X": 0.1844055950641632,
            "Y": 0.612853467464447
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.612853467464447
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6442786455154419
          },
          {
            "X": 0.1844055950641632,
            "Y": 0.6442786455154419
          }
        ]
      },
      "Id": "7af04e27-0c15-447e-a569-b30edb99a133"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9652328491211,
```

```
      "Text": "Employer",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08150768280029297,
          "Height": 0.0392492301762104,
          "Left": 0.2647075653076172,
          "Top": 0.6140711903572083
        },
        "Polygon": [
          {
            "X": 0.2647075653076172,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.34621524810791016,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.34621524810791016,
            "Y": 0.6533204317092896
          },
          {
            "X": 0.2647075653076172,
            "Y": 0.6533204317092896
          }
        ]
      },
      "Id": "a9bfeb55-75cd-47cd-b953-728e602a3564"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94273376464844,
      "Text": "Name",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05018233880400658,
          "Height": 0.03248906135559082,
          "Left": 0.34925445914268494,
          "Top": 0.6162016987800598
        },
        "Polygon": [
          {
```

```
            "X": 0.34925445914268494,
            "Y": 0.6162016987800598
          },
          {
            "X": 0.3994368016719818,
            "Y": 0.6162016987800598
          },
          {
            "X": 0.3994368016719818,
            "Y": 0.6486907601356506
          },
          {
            "X": 0.34925445914268494,
            "Y": 0.6486907601356506
          }
        ]
      },
      "Id": "9f0f9c06-d02c-4b07-bb39-7ade70be2c1b"
    },
    {
      "BlockType": "WORD",
      "Confidence": 98.85071563720703,
      "Text": "Position",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07007700204849243,
          "Height": 0.03255689889192581,
          "Left": 0.49973347783088684,
          "Top": 0.6164342164993286
        },
        "Polygon": [
          {
            "X": 0.49973347783088684,
            "Y": 0.6164342164993286
          },
          {
            "X": 0.5698104500770569,
            "Y": 0.6164342164993286
          },
          {
            "X": 0.5698104500770569,
            "Y": 0.6489911079406738
          },
```

```
        {
          "X": 0.49973347783088684,
          "Y": 0.6489911079406738
        }
      ]
    },
    "Id": "6d5edf02-845c-40e0-9514-e56d0d652ae0"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.86096954345703,
    "Text": "Held",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.04017873853445053,
        "Height": 0.03292537108063698,
        "Left": 0.5734874606132507,
        "Top": 0.614840030670166
      },
      "Polygon": [
        {
          "X": 0.5734874606132507,
          "Y": 0.614840030670166
        },
        {
          "X": 0.6136662364006042,
          "Y": 0.614840030670166
        },
        {
          "X": 0.6136662364006042,
          "Y": 0.6477653980255127
        },
        {
          "X": 0.5734874606132507,
          "Y": 0.6477653980255127
        }
      ]
    },
    "Id": "3297ab59-b237-45fb-ae60-a108f0c95ac2"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.97740936279297,
```

```
      "Text": "Reason",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06497219949960709,
          "Height": 0.03248770162463188,
          "Left": 0.7430596351623535,
          "Top": 0.6136704087257385
        },
        "Polygon": [
          {
            "X": 0.7430596351623535,
            "Y": 0.6136704087257385
          },
          {
            "X": 0.8080317974090576,
            "Y": 0.6136704087257385
          },
          {
            "X": 0.8080317974090576,
            "Y": 0.6461580991744995
          },
          {
            "X": 0.7430596351623535,
            "Y": 0.6461580991744995
          }
        ]
      },
      "Id": "f4b8cf26-d2da-4a76-8345-69562de3cc11"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98371887207031,
      "Text": "for",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.029645200818777084,
          "Height": 0.03462234139442444,
          "Left": 0.8108851909637451,
          "Top": 0.6117717623710632
        },
        "Polygon": [
          {
```

```
            "X": 0.8108851909637451,
            "Y": 0.6117717623710632
          },
          {
            "X": 0.8405303955078125,
            "Y": 0.6117717623710632
          },
          {
            "X": 0.8405303955078125,
            "Y": 0.6463940739631653
          },
          {
            "X": 0.8108851909637451,
            "Y": 0.6463940739631653
          }
        ]
      },
      "Id": "386d4a63-1194-4c0e-a18d-4d074a0b1f93"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98424530029297,
      "Text": "leaving",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06517849862575531,
          "Height": 0.040626998990774155,
          "Left": 0.8430007100105286,
          "Top": 0.6116235852241516
        },
        "Polygon": [
          {
            "X": 0.8430007100105286,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6522505879402161
          },
```

```
        {
          "X": 0.8430007100105286,
          "Y": 0.6522505879402161
        }
      ]
    },
    "Id": "a8622541-1896-4d54-8d10-7da2c800ec5c"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.77413177490234,
    "Text": "1/15/2009",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.08799663186073303,
        "Height": 0.03832906112074852,
        "Left": 0.03175082430243492,
        "Top": 0.691371738910675
      },
      "Polygon": [
        {
          "X": 0.03175082430243492,
          "Y": 0.691371738910675
        },
        {
          "X": 0.11974745243787766,
          "Y": 0.691371738910675
        },
        {
          "X": 0.11974745243787766,
          "Y": 0.7297008037567139
        },
        {
          "X": 0.03175082430243492,
          "Y": 0.7297008037567139
        }
      ]
    },
    "Id": "da7a6482-0964-49a4-bc7d-56942ff3b4e1"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.72286224365234,
```

```
      "Text": "6/30/2011",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08843102306127548,
          "Height": 0.03991425037384033,
          "Left": 0.14642837643623352,
          "Top": 0.6919752955436707
        },
        "Polygon": [
          {
            "X": 0.14642837643623352,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.731889545917511
          },
          {
            "X": 0.14642837643623352,
            "Y": 0.731889545917511
          }
        ]
      },
      "Id": "5a8da66a-ecce-4ee9-a765-a46d6cdc6cde"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92295837402344,
      "Text": "Any",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.034067559987306595,
          "Height": 0.037968240678310394,
          "Left": 0.2626699209213257,
          "Top": 0.6972727179527283
        },
        "Polygon": [
          {
```

```
            "X": 0.2626699209213257,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.2967374622821808,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.2967374622821808,
            "Y": 0.7352409362792969
          },
          {
            "X": 0.2626699209213257,
            "Y": 0.7352409362792969
          }
        ]
      },
      "Id": "77749c2b-aa7f-450e-8dd2-62bcaf253ba2"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.81578063964844,
      "Text": "Company",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08160992711782455,
          "Height": 0.03890080004930496,
          "Left": 0.29906952381134033,
          "Top": 0.6978086829185486
        },
        "Polygon": [
          {
            "X": 0.29906952381134033,
            "Y": 0.6978086829185486
          },
          {
            "X": 0.3806794583797455,
            "Y": 0.6978086829185486
          },
          {
            "X": 0.3806794583797455,
            "Y": 0.736709475517273
          },
```

```
        {
          "X": 0.29906952381134033,
          "Y": 0.736709475517273
        }
      ]
    },
    "Id": "713bad19-158d-4e3e-b01f-f5707ddb04e5"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.37964630126953,
    "Text": "Assistant",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.0789310410618782,
        "Height": 0.03139699995517731,
        "Left": 0.49814170598983765,
        "Top": 0.7005078196525574
      },
      "Polygon": [
        {
          "X": 0.49814170598983765,
          "Y": 0.7005078196525574
        },
        {
          "X": 0.5770727396011353,
          "Y": 0.7005078196525574
        },
        {
          "X": 0.5770727396011353,
          "Y": 0.7319048047065735
        },
        {
          "X": 0.49814170598983765,
          "Y": 0.7319048047065735
        }
      ]
    },
    "Id": "989944f9-f684-4714-87d8-9ad9a321d65c"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.784912109375,
```

```
      "Text": "baker",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.050264399498701096,
          "Height": 0.03237773850560188,
          "Left": 0.5806865096092224,
          "Top": 0.699238657951355
        },
        "Polygon": [
          {
            "X": 0.5806865096092224,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.7316163778305054
          },
          {
            "X": 0.5806865096092224,
            "Y": 0.7316163778305054
          }
        ]
      },
      "Id": "ae82e2aa-1601-4e0c-8340-1db7ad0c9a31"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.96180725097656,
      "Text": "relocated",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08668994158506393,
          "Height": 0.03330250084400177,
          "Left": 0.7426905632019043,
          "Top": 0.6974037289619446
        },
        "Polygon": [
          {
```

```
            "X": 0.7426905632019043,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.7307062149047852
          },
          {
            "X": 0.7426905632019043,
            "Y": 0.7307062149047852
          }
        ]
      },
      "Id": "a9cf9a8c-fdaa-413e-9346-5a28a98aebdb"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98190307617188,
      "Text": "7/1/2011",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09747002273797989,
          "Height": 0.07067439705133438,
          "Left": 0.028500309213995934,
          "Top": 0.7745237946510315
        },
        "Polygon": [
          {
            "X": 0.028500309213995934,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.8451982140541077
          },
```

```
        {
          "X": 0.028500309213995934,
          "Y": 0.8451982140541077
        }
      ]
    },
    "Id": "0f711065-1872-442a-ba6d-8fababaa452a"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.98418426513672,
    "Text": "8/10/2013",
    "TextType": "HANDWRITING",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.10664612054824829,
        "Height": 0.06439515948295593,
        "Left": 0.14159755408763885,
        "Top": 0.7791688442230225
      },
      "Polygon": [
        {
          "X": 0.14159755408763885,
          "Y": 0.7791688442230225
        },
        {
          "X": 0.24824367463588715,
          "Y": 0.7791688442230225
        },
        {
          "X": 0.24824367463588715,
          "Y": 0.843563973903656
        },
        {
          "X": 0.14159755408763885,
          "Y": 0.843563973903656
        }
      ]
    },
    "Id": "a92d8eef-db28-45ba-801a-5da0f589d277"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.97722625732422,
```

```
      "Text": "Example",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12127546221017838,
          "Height": 0.05682983994483948,
          "Left": 0.26764172315597534,
          "Top": 0.794414758682251
        },
        "Polygon": [
          {
            "X": 0.26764172315597534,
            "Y": 0.794414758682251
          },
          {
            "X": 0.3889172077178955,
            "Y": 0.794414758682251
          },
          {
            "X": 0.3889172077178955,
            "Y": 0.8512446284294128
          },
          {
            "X": 0.26764172315597534,
            "Y": 0.8512446284294128
          }
        ]
      },
      "Id": "d6962efb-34ab-4ffb-9f2f-5f263e813558"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98429870605469,
      "Text": "Corp.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07650306820869446,
          "Height": 0.05481306090950966,
          "Left": 0.4026312530040741,
          "Top": 0.7980174422264099
        },
        "Polygon": [
          {
```

```
            "X": 0.4026312530040741,
            "Y": 0.7980174422264099
          },
          {
            "X": 0.47913432121276855,
            "Y": 0.7980174422264099
          },
          {
            "X": 0.47913432121276855,
            "Y": 0.8528305292129517
          },
          {
            "X": 0.4026312530040741,
            "Y": 0.8528305292129517
          }
        ]
      },
      "Id": "1876c8ea-d3e8-4c39-870e-47512b3b5080"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.91166687011719,
      "Text": "Baker",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09931197017431259,
          "Height": 0.06008723005652428,
          "Left": 0.5098910331726074,
          "Top": 0.787897527217865
        },
        "Polygon": [
          {
            "X": 0.5098910331726074,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.8479847311973572
          },
```

```
        {
          "X": 0.5098910331726074,
          "Y": 0.8479847311973572
        }
      ]
    },
    "Id": "00adeaef-ed57-44eb-b8a9-503575236d62"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.98870849609375,
    "Text": "better",
    "TextType": "HANDWRITING",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.10782185196876526,
        "Height": 0.06207133084535599,
        "Left": 0.7428008317947388,
        "Top": 0.7928366661071777
      },
      "Polygon": [
        {
          "X": 0.7428008317947388,
          "Y": 0.7928366661071777
        },
        {
          "X": 0.8506226539611816,
          "Y": 0.7928366661071777
        },
        {
          "X": 0.8506226539611816,
          "Y": 0.8549079895019531
        },
        {
          "X": 0.7428008317947388,
          "Y": 0.8549079895019531
        }
      ]
    },
    "Id": "c0fc9a58-7a4b-4f69-bafd-2cff32be2665"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.8883285522461,
```

```
      "Text": "opp.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07421936094760895,
          "Height": 0.058906231075525284,
          "Left": 0.8577775359153748,
          "Top": 0.8038780689239502
        },
        "Polygon": [
          {
            "X": 0.8577775359153748,
            "Y": 0.8038780689239502
          },
          {
            "X": 0.9319969415664673,
            "Y": 0.8038780689239502
          },
          {
            "X": 0.9319969415664673,
            "Y": 0.8627843260765076
          },
          {
            "X": 0.8577775359153748,
            "Y": 0.8627843260765076
          }
        ]
      },
      "Id": "bf6dc8ee-2fb3-4b6c-aee4-31e96912a2d8"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92573547363281,
      "Text": "8/15/2013",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10257463902235031,
          "Height": 0.05412459000945091,
          "Left": 0.027909137308597565,
          "Top": 0.8608770370483398
        },
        "Polygon": [
          {
```

```
            "X": 0.027909137308597565,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.915001630783081
          },
          {
            "X": 0.027909137308597565,
            "Y": 0.915001630783081
          }
        ]
      },
      "Id": "5384f860-f857-4a94-9438-9dfa20eed1c6"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99625396728516,
      "Text": "Present",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09982697665691376,
          "Height": 0.06888339668512344,
          "Left": 0.1420602649450302,
          "Top": 0.8511748909950256
        },
        "Polygon": [
          {
            "X": 0.1420602649450302,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.9200583100318909
          },
```

```
          {
            "X": 0.1420602649450302,
            "Y": 0.9200583100318909
          }
        ]
      },
      "Id": "0bb96ed6-b2e6-4da4-90b3-b85561bbd89d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9826431274414,
      "Text": "AnyCompany",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18611273169517517,
          "Height": 0.08581399917602539,
          "Left": 0.2615866959095001,
          "Top": 0.869536280632019
        },
        "Polygon": [
          {
            "X": 0.2615866959095001,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994276046753,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994276046753,
            "Y": 0.9553502798080444
          },
          {
            "X": 0.2615866959095001,
            "Y": 0.9553502798080444
          }
        ]
      },
      "Id": "25343360-d906-440a-88b7-92eb89e95949"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99523162841797,
```

```
      "Text": "head",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07429949939250946,
          "Height": 0.05485520139336586,
          "Left": 0.49359121918678284,
          "Top": 0.8714361190795898
        },
        "Polygon": [
          {
            "X": 0.49359121918678284,
            "Y": 0.8714361190795898
          },
          {
            "X": 0.5678907036781311,
            "Y": 0.8714361190795898
          },
          {
            "X": 0.5678907036781311,
            "Y": 0.926291286945343
          },
          {
            "X": 0.49359121918678284,
            "Y": 0.926291286945343
          }
        ]
      },
      "Id": "0ef3c194-8322-4575-94f1-82819ee57e3a"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99574279785156,
      "Text": "baker",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1019822508096695,
          "Height": 0.05615599825978279,
          "Left": 0.585354208946228,
          "Top": 0.8702592849731445
        },
        "Polygon": [
          {
```

```
            "X": 0.585354208946228,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873364448547363,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873364448547363,
            "Y": 0.9264153242111206
          },
          {
            "X": 0.585354208946228,
            "Y": 0.9264153242111206
          }
        ]
      },
      "Id": "d296acd9-3e9a-4985-95f8-f863614f2c46"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9880599975586,
      "Text": "N/A,",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08230073750019073,
          "Height": 0.06588289886713028,
          "Left": 0.7411766648292542,
          "Top": 0.8722732067108154
        },
        "Polygon": [
          {
            "X": 0.7411766648292542,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.8234773874282837,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.8234773874282837,
            "Y": 0.9381561279296875
          },
```

```
          {
            "X": 0.7411766648292542,
            "Y": 0.9381561279296875
          }
        ]
      },
      "Id": "195cfb5b-ae06-4203-8520-4e4b0a73b5ce"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97914123535156,
      "Text": "current",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12791454792022705,
          "Height": 0.04768490046262741,
          "Left": 0.8387037515640259,
          "Top": 0.8843405842781067
        },
        "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
  }
}
}
```

使用 Amazon Textract 检测文档文本

要检测文档中的文本，请使用DetectDocumentText操作，然后将文档文件作为输入传递。DetectDocumentText返回 JSON 结构，其中包含检测到的文本的行和单词、文档中文本的位置以及检测到的文本之间的关系。有关更多信息，请参阅 检测文本。

您可以提供输入文档作为图像字节数组 (base64 编码的图像字节) 或 Amazon S3 对象。在此过程中，您将图像文件上传到您的 S3 存储桶并指定文件名称。

检测文档中的文本 (API)

1. 如果您尚未执行以下操作，请：

   a. 使用创建或更新 IAM 用户AmazonTextractFullAccess和AmazonS3ReadOnlyAccess权限。有关更多信息，请参阅 第 1 步：设置 AWS 账户并创建 IAM 用户。

   b. 安装和配置 AWS CLI 和 AWS 开发工具包。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。

2. 将文档上传到 S3 存储桶。

   有关说明，请参阅将对象上传到 Amazon S3中的Amazon Simple Service 用户指南.

3. 使用以下示例调用 DetectDocumentText 操作。

   Java

   以下示例代码在检测到的文本行周围显示文档和框。

在函数main，替换的值bucket和document将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档的名称。

```
//Calls DetectDocumentText.
//Loads document from S3 bucket. Displays the document and bounding boxes around
 detected lines/words of text.
package com.amazonaws.samples;

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.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 = "";


        AmazonS3 s3client = AmazonS3ClientBuilder.standard()
                .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()
                .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);

    }
}
```

AWS CLI

此 AWS CLI 命令显示 detect-document-text CLI 操作的 JSON 输出。

替换的值Bucket和Name将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档的名称。

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

Python

以下示例代码显示检测到的文本行周围的文档和框。

在函数main，替换的值bucket和document将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档的名称。

```
#Detects text in a document stored in an S3 bucket. Display polygon box around
 text and angled text
import boto3
import io
from io import BytesIO
import sys

import psutil
import time

import math
from PIL import Image, ImageDraw, ImageFont


# Displays information about a block returned by text detection and text
 analysis
def DisplayBlockInformation(block):
```

```
    print('Id: {}'.format(block['Id']))
    if 'Text' in block:
        print('    Detected: ' + block['Text'])
    print('    Type: ' + block['BlockType'])

    if 'Confidence' in block:
        print('    Confidence: ' + "{:.2f}".format(block['Confidence']) + "%")

    if block['BlockType'] == 'CELL':
        print("    Cell information")
        print("        Column: " + str(block['ColumnIndex']))
        print("        Row: " + str(block['RowIndex']))
        print("        ColumnSpan: " + str(block['ColumnSpan']))
        print("        RowSpan: " + str(block['RowSpan']))

    if 'Relationships' in block:
        print('    Relationships: {}'.format(block['Relationships']))
    print('    Geometry: ')
    print('        Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
    print('        Polygon: {}'.format(block['Geometry']['Polygon']))

    if block['BlockType'] == "KEY_VALUE_SET":
        print ('    Entity Type: ' + block['EntityTypes'][0])
    if 'Page' in block:
        print('Page: ' + block['Page'])
    print()

def process_text_detection(bucket, document):


    #Get the document from S3
    s3_connection = boto3.resource('s3')

    s3_object = s3_connection.Object(bucket,document)
    s3_response = s3_object.get()

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


    # Detect text in the document

    client = boto3.client('textract')
    #process using image bytes
```

```
    #image_binary = stream.getvalue()
    #response = client.detect_document_text(Document={'Bytes': image_binary})

    #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
    draw = ImageDraw.Draw(image)
    print ('Detected Document Text')

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:
            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')

                # Uncomment to draw bounding box
                #box=block['Geometry']['BoundingBox']
                #left = width * box['Left']
                #top = height * box['Top']
                #draw.rectangle([left,top, left + (width * box['Width']), top
 +(height * box['Height'])],outline='black')


    # Display the image
    image.show()
    # display image for 10 seconds


    return len(blocks)

def main():

    bucket = ''
    document = ''
    block_count=process_text_detection(bucket,document)
    print("Blocks detected: " + str(block_count))

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

Node.js

以下 Node.js 示例代码显示了检测到的文本行周围的文档和框，将结果的图像输出到运行代码的目录中。它利用`image-size`和`images`软件包。

在函数`main`，替换的值`bucket`和`document`将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档的名称。替换的值`regionConfig`使用您的账户所在地区的名称。

```
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");

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

// Set variables
const bucket = 'bucket-name' // the s3 bucket name
const photo  = 'image-name' // the name of file
const regionConfig = 'region'

// Set region if needed
AWS.config.update({region:regionConfig});

// 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()
```

4. 运行示例。Python 和 Java 示例显示了文档图像。每行检测到的文本都有一个黑框。绿色垂直线是检测到的单词的开头。红色垂直线是检测到的单词的末尾。这些区域有：AWS CLI示例仅显示的 JSON 输出DetectDocumentTextoperation.

使用 Amazon Textract 分析文档文本

要分析文档中的文本，请使用AnalyzeDocument操作，然后将文档文件作为输入传递。AnalyzeDocument返回一个 JSON 结构，其中包含分析的文本。有关更多信息，请参阅 分析文档。

您可以提供输入文档作为图像字节数组 (base64 编码的图像字节) 或 Amazon S3 对象。在此过程中，您将图像文件上传到您的 S3 存储桶并指定文件名称。

分析文档中的文本 (API)

1. 如果您尚未执行以下操作，请：

   a. 使用创建或更新 IAM 用户AmazonTextractFullAccess和AmazonS3ReadOnlyAccess权限。有关更多信息，请参阅 第 1 步：设置 AWS 账户并创建 IAM 用户。

   b. 安装和配置 AWS CLI 和 AWS 开发工具包。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。

2. 将包含文档的图像上传到您的 S3 存储桶。

   有关说明，请参阅将对象上传到 Amazon S3中的Amazon Simple Service 用户指南.

3. 使用以下示例调用 AnalyzeDocument 操作。

   Java

   以下示例代码显示检测到的项目周围的文档和框。

在函数main，替换的值bucket和document将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档图像的名称。

```
//Loads document from S3 bucket. Displays the document and polygon around
 detected lines of text.
package com.amazonaws.samples;

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.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 = "";

        AmazonS3 s3client = AmazonS3ClientBuilder.standard()
                .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()
                .withEndpointConfiguration(endpoint).build();


        AnalyzeDocumentRequest request = new AnalyzeDocumentRequest()
                .withFeatureTypes("TABLES","FORMS")
                 .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);

    }
}
```

AWS CLI

此 AWS CLI 命令显示 `detect-document-text` CLI 操作的 JSON 输出。

替换的值`Bucket`和`Name`将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档的名称。

```
aws textract analyze-document \
 --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
 --feature-types '["TABLES","FORMS"]'
```

Python

以下示例代码显示检测到的项目周围的文档和框。

在函数main，替换的值bucket和document将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档的名称。

```
#Analyzes text in a document stored in an S3 bucket. Display polygon box around
 text and angled text
import boto3
import io
from io import BytesIO
import sys

import math
from PIL import Image, ImageDraw, ImageFont

def ShowBoundingBox(draw,box,width,height,boxColor):

    left = width * box['Left']
    top = height * box['Top']
    draw.rectangle([left,top, left + (width * box['Width']), top +(height *
 box['Height'])],outline=boxColor)

def ShowSelectedElement(draw,box,width,height,boxColor):

    left = width * box['Left']
    top = height * box['Top']
    draw.rectangle([left,top, left + (width * box['Width']), top +(height *
 box['Height'])],fill=boxColor)

# Displays information about a block returned by text detection and text
 analysis
def DisplayBlockInformation(block):
    print('Id: {}'.format(block['Id']))
    if 'Text' in block:
        print('    Detected: ' + block['Text'])
```

```
    print('    Type: ' + block['BlockType'])

    if 'Confidence' in block:
        print('    Confidence: ' + "{:.2f}".format(block['Confidence']) + "%")

    if block['BlockType'] == 'CELL':
        print("    Cell information")
        print("        Column:" + str(block['ColumnIndex']))
        print("        Row:" + str(block['RowIndex']))
        print("        Column Span:" + str(block['ColumnSpan']))
        print("        RowSpan:" + str(block['ColumnSpan']))

    if 'Relationships' in block:
        print('    Relationships: {}'.format(block['Relationships']))
    print('    Geometry: ')
    print('        Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
    print('        Polygon: {}'.format(block['Geometry']['Polygon']))

    if block['BlockType'] == "KEY_VALUE_SET":
        print ('    Entity Type: ' + block['EntityTypes'][0])

    if block['BlockType'] == 'SELECTION_ELEMENT':
        print('    Selection element detected: ', end='')

        if block['SelectionStatus'] =='SELECTED':
            print('Selected')
        else:
            print('Not selected')

    if 'Page' in block:
        print('Page: ' + block['Page'])
    print()

def process_text_analysis(bucket, document):

    #Get the document from S3
    s3_connection = boto3.resource('s3')

    s3_object = s3_connection.Object(bucket,document)
    s3_response = s3_object.get()

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

```
    # Analyze the document
    client = boto3.client('textract')

    image_binary = stream.getvalue()
    response = client.analyze_document(Document={'Bytes': image_binary},
        FeatureTypes=["TABLES", "FORMS"])

    ### Alternatively, process using S3 object ###
    #response = client.analyze_document(
    #    Document={'S3Object': {'Bucket': bucket, 'Name': document}},
    #    FeatureTypes=["TABLES", "FORMS"])

    ### To use a local file ###
    # with open("pathToFile", 'rb') as img_file:
        ### To display image using PIL ###
    #    image = Image.open()
        ### Read bytes ###
    #    img_bytes = img_file.read()
    #    response = client.analyze_document(Document={'Bytes': img_bytes},
 FeatureTypes=["TABLES", "FORMS"])


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

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:

        DisplayBlockInformation(block)

        draw=ImageDraw.Draw(image)
        if block['BlockType'] == "KEY_VALUE_SET":
            if block['EntityTypes'][0] == "KEY":
                ShowBoundingBox(draw, block['Geometry']
['BoundingBox'],width,height,'red')
            else:
                ShowBoundingBox(draw, block['Geometry']
['BoundingBox'],width,height,'green')

        if block['BlockType'] == 'TABLE':
```

```
            ShowBoundingBox(draw, block['Geometry']['BoundingBox'],width,height,
 'blue')

        if block['BlockType'] == 'CELL':
            ShowBoundingBox(draw, block['Geometry']['BoundingBox'],width,height,
 'yellow')
        if block['BlockType'] == 'SELECTION_ELEMENT':
            if block['SelectionStatus'] =='SELECTED':
                ShowSelectedElement(draw, block['Geometry']
['BoundingBox'],width,height, 'blue')

            #uncomment to draw polygon for all Blocks
            #points=[]
            #for polygon in block['Geometry']['Polygon']:
            #    points.append((width * polygon['X'], height * polygon['Y']))
            #draw.polygon((points), outline='blue')

    # Display the image
    image.show()
    return len(blocks)


def main():

    bucket = ''
    document = ''
    block_count=process_text_analysis(bucket,document)
    print("Blocks detected: " + str(block_count))

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

Node.js

以下示例代码显示检测到的项目周围的文档和框。

在下面的代码中，替换bucket和photo将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档的名称。替换的值region使用与您的账户关联的区域。

```
// Import required AWS SDK clients and commands for Node.js
import { AnalyzeDocumentCommand } from  "@aws-sdk/client-textract";
import  { TextractClient } from "@aws-sdk/client-textract";
```

```
// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
// Create SNS service object.
const textractClient = new TextractClient({ region: REGION });

const bucket = 'buckets'
const photo = 'photo'

// Set params
const params = {
    Document: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
    FeatureTypes: ['TABLES', 'FORMS'],
  }

const displayBlockInfo = async (response) => {
    try {
        response.Blocks.forEach(block => {
            console.log(`ID: ${block.Id}`)
            console.log(`Block Type: ${block.BlockType}`)
            if ("Text" in block && block.Text !== undefined){
                console.log(`Text: ${block.Text}`)
            }
            else{}
            if ("Confidence" in block && block.Confidence !== undefined){
                console.log(`Confidence: ${block.Confidence}`)
            }
            else{}
            if (block.BlockType == 'CELL'){
                console.log("Cell info:")
                console.log(`   Column Index - ${block.ColumnIndex}`)
                console.log(`   Row - ${block.RowIndex}`)
                console.log(`   Column Span - ${block.ColumnSpan}`)
                console.log(`   Row Span - ${block.RowSpan}`)
            }
            if ("Relationships" in block && block.Relationships !== undefined){
                console.log(block.Relationships)
                console.log("Geometry:")
                console.log(`   Bounding Box -
 ${JSON.stringify(block.Geometry.BoundingBox)}`)
```

```
                console.log(`   Polygon -
 ${JSON.stringify(block.Geometry.Polygon)}`)
            }
            console.log("-----")
        });
      } catch (err) {
        console.log("Error", err);
      }
}

const analyze_document_text = async () => {
    try {
        const analyzeDoc = new AnalyzeDocumentCommand(params);
        const response = await textractClient.send(analyzeDoc);
        //console.log(response)
        displayBlockInfo(response)
        return response; // For unit tests.
      } catch (err) {
        console.log("Error", err);
      }
}

analyze_document_text()
```

4. 运行示例。Python 和 Java 示例使用以下彩色边界框显示文档图像：

   - 红色 — KEY Block 对象
   - 绿色 — VALUE Block 对象
   - 蓝色 — TABLE Block 对象
   - 黄色 — CELL Block 对象

   选定的选择元素用蓝色填充。

   这些区域有：AWS CLI示例仅显示的 JSON 输出`AnalyzeDocument`operation.

使用 Amazon Textract 分析发票和收据

要分析发票和收据单据，您可以使用 AnalyzeEsend API，然后将文档文件作为输入进行传递。`AnalyzeExpense`是一种同步操作，它返回包含分析过的文本的 JSON 结构。有关更多信息，请参阅 分析发票和收据。

要异步分析发票和收据，请使用StartExpenseAnalysis开始处理输入文档文件并使用GetExpenseAnalysis以获得结果。

您可以提供输入文档作为图像字节数组 (base64 编码的图像字节) 或 Amazon S3 对象。在此过程中，您将图像文件上传到您的 S3 存储桶并指定文件名称。

分析发票或收据 (API)

1. 如果您尚未执行以下操作，请：

   a. 使用创建或更新 IAM 用户AmazonTextractFullAccess和AmazonS3ReadOnlyAccess权限。有关更多信息，请参阅 第 1 步：设置 AWS 账户并创建 IAM 用户。

   b. 安装和配置 AWS CLI 和 AWS 开发工具包。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。

2. 将包含文档的图像上传到您的 S3 存储桶。

   有关说明，请参阅将对象上传到 Amazon S3中的Amazon Simple Service 用户指南.

3. 使用以下示例调用 AnalyzeExpense 操作。

   CLI

```
  aws textract analyze-expense --document '{"S3Object": {"Bucket": "bucket
 name","Name":
"object name"}}'
```

   Python

```
import boto3
import io
from PIL import Image, ImageDraw

def draw_bounding_box(key, val, width, height, draw):
    # If a key is Geometry, draw the bounding box info in it
    if "Geometry" in key:
        # Draw bounding box information
        box = val["BoundingBox"]
        left = width * box['Left']
        top = height * box['Top']
```

```
        draw.rectangle([left, top, left + (width * box['Width']), top + (height
 * box['Height'])],
                       outline='black')

# Takes a field as an argument and prints out the detected labels and values
def print_labels_and_values(field):
    # Only if labels are detected and returned
    if "LabelDetection" in field:
        print("Summary Label Detection - Confidence: {}".format(
            str(field.get("LabelDetection")["Confidence"])) + ", "
              + "Summary Values: {}".format(str(field.get("LabelDetection")
["Text"])))
        print(field.get("LabelDetection")["Geometry"])
    else:
        print("Label Detection - No labels returned.")
    if "ValueDetection" in field:
        print("Summary Value Detection - Confidence: {}".format(
            str(field.get("ValueDetection")["Confidence"])) + ", "
              + "Summary Values: {}".format(str(field.get("ValueDetection")
["Text"])))
        print(field.get("ValueDetection")["Geometry"])
    else:
        print("Value Detection - No values returned")

def process_text_detection(bucket, document):
    # Get the document from S3
    s3_connection = boto3.resource('s3')
    s3_object = s3_connection.Object(bucket, document)
    s3_response = s3_object.get()

    # opening binary stream using an in-memory bytes buffer
    stream = io.BytesIO(s3_response['Body'].read())

    # loading stream into image
    image = Image.open(stream)

    # Detect text in the document
    client = boto3.client('textract', region_name="us-east-1")

    # process using S3 object
    response = client.analyze_expense(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    # Set width and height to display image and draw bounding boxes
```

```
    # Create drawing object
    width, height = image.size
    draw = ImageDraw.Draw(image)

    for expense_doc in response["ExpenseDocuments"]:
        for line_item_group in expense_doc["LineItemGroups"]:
            for line_items in line_item_group["LineItems"]:
                for expense_fields in line_items["LineItemExpenseFields"]:
                    print_labels_and_values(expense_fields)
                    print()

        print("Summary:")
        for summary_field in expense_doc["SummaryFields"]:
            print_labels_and_values(summary_field)
            print()

        #For draw bounding boxes
        for line_item_group in expense_doc["LineItemGroups"]:
            for line_items in line_item_group["LineItems"]:
                for expense_fields in line_items["LineItemExpenseFields"]:
                    for key, val in expense_fields["ValueDetection"].items():
                        if "Geometry" in key:
                            draw_bounding_box(key, val, width, height, draw)

        for label in expense_doc["SummaryFields"]:
            if "LabelDetection" in label:
                for key, val in label["LabelDetection"].items():
                    draw_bounding_box(key, val, width, height, draw)

    # Display the image
    image.show()

def main():
    bucket = 'Bucket-Name'
    document = 'Document-Name'
    process_text_detection(bucket, document)

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

Java

```
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.*;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeExpenseRequest;
import software.amazon.awssdk.services.textract.model.AnalyzeExpenseResponse;
import software.amazon.awssdk.services.textract.model.BoundingBox;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.ExpenseDocument;
import software.amazon.awssdk.services.textract.model.ExpenseField;
import software.amazon.awssdk.services.textract.model.LineItemFields;
import software.amazon.awssdk.services.textract.model.LineItemGroup;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.model.Point;

/**
 *
 * Demo code to parse Textract AnalyzeExpense API
 *
 */
public class TextractAnalyzeExpenseSample extends JPanel {

 private static final long serialVersionUID = 1L;
```

```
BufferedImage image;
static AnalyzeExpenseResponse result;

public TextractAnalyzeExpenseSample(AnalyzeExpenseResponse documentResult,
BufferedImage bufImage) throws Exception {
 super();

 result = documentResult; // Results of analyzeexpense summaryfields and
lineitemgroups detection.
 image = bufImage; // The image containing the document.

}

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

 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 summaryfields and lineitemgroups and display boundedboxes
around lines of detected label and value.
 List<ExpenseDocument> expenseDocuments = result.expenseDocuments();
 for (ExpenseDocument expenseDocument : expenseDocuments) {

  if (expenseDocument.hasSummaryFields()) {
   DisplayAnalyzeExpenseSummaryInfo(expenseDocument);
   List<ExpenseField> summaryfields = expenseDocument.summaryFields();
   for (ExpenseField summaryfield : summaryfields) {

    if (summaryfield.valueDetection() != null) {
     ShowBoundingBox(image.getHeight(this), image.getWidth(this),
       summaryfield.valueDetection().geometry().boundingBox(), g2d, new
Color(0, 0, 0));
    }

    if (summaryfield.labelDetection() != null) {

     ShowBoundingBox(image.getHeight(this), image.getWidth(this),
       summaryfield.labelDetection().geometry().boundingBox(), g2d, new
Color(0, 0, 0));

    }
```

```
    }

   }

   if (expenseDocument.hasLineItemGroups()) {
    DisplayAnalyzeExpenseLineItemGroupsInfo(expenseDocument);

    List<LineItemGroup> lineitemgroups = expenseDocument.lineItemGroups();

    for (LineItemGroup lineitemgroup : lineitemgroups) {

     if (lineitemgroup.hasLineItems()) {

      List<LineItemFields> lineItems = lineitemgroup.lineItems();
      for (LineItemFields lineitemfield : lineItems) {

       if (lineitemfield.hasLineItemExpenseFields()) {

        List<ExpenseField> expensefields =
lineitemfield.lineItemExpenseFields();
        for (ExpenseField expensefield : expensefields) {

         if (expensefield.valueDetection() != null) {
          ShowBoundingBox(image.getHeight(this), image.getWidth(this),
            expensefield.valueDetection().geometry().boundingBox(), g2d,
            new Color(0, 0, 0));
         }

         if (expensefield.labelDetection() != null) {
          ShowBoundingBox(image.getHeight(this), image.getWidth(this),
            expensefield.labelDetection().geometry().boundingBox(), g2d,
            new Color(0, 0, 0));
         }

        }
       }

      }

     }

    }
```

```
  }
 }

}

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

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

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

}

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

 float left = (float) imageWidth * (float) box.left();
 float top = (float) imageHeight * (float) box.top();
 System.out.println(left);
 System.out.println(top);

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

}

// 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.x() * imageWidth)), Math.round(point.y() *
imageHeight));
 }
 g2d.drawPolygon(polygon);
}

private void DisplayAnalyzeExpenseSummaryInfo(ExpenseDocument expensedocument)
{
 System.out.println(" ExpenseId : " + expensedocument.expenseIndex());
 System.out.println("    Expense Summary information:");
 if (expensedocument.hasSummaryFields()) {

  List<ExpenseField> summaryfields = expensedocument.summaryFields();

  for (ExpenseField summaryfield : summaryfields) {

   System.out.println("    Page: " + summaryfield.pageNumber());
   if (summaryfield.type() != null) {

    System.out.println("    Expense Summary Field Type:" +
summaryfield.type().text());

   }
   if (summaryfield.labelDetection() != null) {

    System.out.println("    Expense Summary Field Label:" +
summaryfield.labelDetection().text());
    System.out.println("    Geometry");
    System.out.println("        Bounding Box: "
      + summaryfield.labelDetection().geometry().boundingBox().toString());
    System.out.println(
      "        Polygon: " +
summaryfield.labelDetection().geometry().polygon().toString());

   }
   if (summaryfield.valueDetection() != null) {
    System.out.println("    Expense Summary Field Value:" +
summaryfield.valueDetection().text());
    System.out.println("    Geometry");
    System.out.println("        Bounding Box: "
      + summaryfield.valueDetection().geometry().boundingBox().toString());
    System.out.println(
```

```
      "        Polygon: " +
summaryfield.valueDetection().geometry().polygon().toString());

   }

  }

 }

}

private void DisplayAnalyzeExpenseLineItemGroupsInfo(ExpenseDocument
expensedocument) {

 System.out.println(" ExpenseId : " + expensedocument.expenseIndex());
 System.out.println("    Expense LineItemGroups information:");

 if (expensedocument.hasLineItemGroups()) {

  List<LineItemGroup> lineitemgroups = expensedocument.lineItemGroups();

  for (LineItemGroup lineitemgroup : lineitemgroups) {

   System.out.println("    Expense LineItemGroupsIndexID :" +
lineitemgroup.lineItemGroupIndex());

   if (lineitemgroup.hasLineItems()) {

    List<LineItemFields> lineItems = lineitemgroup.lineItems();

    for (LineItemFields lineitemfield : lineItems) {

     if (lineitemfield.hasLineItemExpenseFields()) {

      List<ExpenseField> expensefields = lineitemfield.lineItemExpenseFields();
      for (ExpenseField expensefield : expensefields) {

       if (expensefield.type() != null) {
        System.out.println("    Expense LineItem Field Type:" +
expensefield.type().text());

       }

       if (expensefield.valueDetection() != null) {
```

```
        System.out.println(
          "    Expense Summary Field Value:" +
expensefield.valueDetection().text());
        System.out.println("    Geometry");
        System.out.println("        Bounding Box: "
          + expensefield.valueDetection().geometry().boundingBox().toString());
        System.out.println("        Polygon: "
          + expensefield.valueDetection().geometry().polygon().toString());

       }

       if (expensefield.labelDetection() != null) {
        System.out.println(
          "    Expense LineItem Field Label:" +
expensefield.labelDetection().text());
        System.out.println("    Geometry");
        System.out.println("        Bounding Box: "
          + expensefield.labelDetection().geometry().boundingBox().toString());
        System.out.println("        Polygon: "
          + expensefield.labelDetection().geometry().polygon().toString());
       }

      }
     }

    }

   }
  }
 }

}

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

 // Creates a default async client with credentials and AWS Region loaded from
 // the
 // environment


 S3AsyncClient client =
S3AsyncClient.builder().region(Region.US_EAST_1).build();

 System.out.println("Creating the S3 Client");
```

```
  // Start the call to Amazon S3, not blocking to wait for the result
  CompletableFuture<ResponseBytes<GetObjectResponse>> responseFuture =
 client.getObject(
    GetObjectRequest.builder().bucket("textractanalyzeexpense").key("input/
sample-receipt.jpg").build(),
    AsyncResponseTransformer.toBytes());

  System.out.println("Successfully read the object");

  // When future is complete (either successfully or in error), handle the
  // response
  CompletableFuture<ResponseBytes<GetObjectResponse>> operationCompleteFuture =
 responseFuture
    .whenComplete((getObjectResponse, exception) -> {
     if (getObjectResponse != null) {
      // At this point, the file my-file.out has been created with the data
      // from S3; let's just print the object version
      // Convert this into Async call and remove the below block from here and
 put it
      // outside


      TextractClient textractclient =
 TextractClient.builder().region(Region.US_EAST_1).build();

      AnalyzeExpenseRequest request = AnalyzeExpenseRequest.builder()
        .document(
          Document.builder().s3Object(S3Object.builder().name("YOURObjectName")
            .bucket("YOURBucket").build()).build())
        .build();

      AnalyzeExpenseResponse result = textractclient.analyzeExpense(request);

      System.out.print(result.toString());

      ByteArrayInputStream bais = new
 ByteArrayInputStream(getObjectResponse.asByteArray());
      try {
       BufferedImage image = ImageIO.read(bais);
       System.out.println("Successfully read the image");
       JFrame frame = new JFrame("Expense Image");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       TextractAnalyzeExpense panel = new TextractAnalyzeExpense(result, image);
```

```
       panel.setPreferredSize(new Dimension(image.getWidth(),
 image.getHeight()));
       frame.setContentPane(panel);
       frame.pack();
       frame.setVisible(true);
      } catch (IOException e) {
       throw new RuntimeException(e);
      } catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      }
     } else {
      // Handle the error
      exception.printStackTrace();
     }
    });

  // We could do other work while waiting for the AWS call to complete in
  // the background, but we'll just wait for "whenComplete" to finish instead
  operationCompleteFuture.join();

 }
}
```

Node.Js

```
                    // Import required AWS SDK clients and commands for Node.js
import { AnalyzeExpenseCommand } from  "@aws-sdk/client-textract";
import  { TextractClient } from "@aws-sdk/client-textract";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
// Create SNS service object.
const textractClient = new TextractClient({ region: REGION });

const bucket = 'bucket'
const photo = 'photo'

// Set params
const params = {
```

```
    Document: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
  }

const process_text_detection = async () => {
    try {
        const aExpense = new AnalyzeExpenseCommand(params);
        const response = await textractClient.send(aExpense);
        //console.log(response)
        response.ExpenseDocuments.forEach(doc => {
            doc.LineItemGroups.forEach(items => {
                items.LineItems.forEach(fields => {
                    fields.LineItemExpenseFields.forEach(expenseFields =>{
                        console.log(expenseFields)
                    })
                }
                )}
                )
            }
        )
        return response; // For unit tests.
      } catch (err) {
        console.log("Error", err);
      }
}

process_text_detection()
```

4. 这将为您提供的 JSON 输出`AnalyzeExpense`operation.

使用 Amazon Textract 分析身份文档

要分析身份证件，您可以使用 AnalyZEID API，然后将文档文件作为输入进行传递。`AnalyzeID`返回一个 JSON 结构，其中包含分析的文本。有关更多信息，请参阅 分析身份证件。

您可以提供输入文档作为图像字节数组 (base64 编码的图像字节) 或 Amazon S3 对象。在此过程中，您将图像文件上传到您的 S3 存储桶并指定文件名称。

分析身份证明文件 (API)

1. 如果您尚未执行以下操作，请：

   a. 使用创建或更新 IAM 用户AmazonTextractFullAccess和AmazonS3ReadOnlyAccess权限。有关更多信息，请参阅 第 1 步：设置 AWS 账户并创建 IAM 用户。

   b. 安装和配置 AWS CLI 和 AWS 开发工具包。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。

2. 将包含文档的图像上传到您的 S3 存储桶。

   有关说明，请参阅将对象上传到 Amazon S3中的Amazon Simple Service 用户指南.

3. 使用以下示例调用 AnalyzeID 操作。

   CLI

   以下示例从 S3 存储桶获取输入文件并运行AnalyzeID对它进行操作。在下面的代码中，替换#将您的 S3 存储桶的名称用于##将包含存储桶中的文件的名称以及的值##有的名字region与您的账户关联。

```
aws textract analyze-id --document-pages '[{"S3Object":
{"Bucket":"bucket","Name":"name"}}]' --region region
```

   您还可以通过向输入中添加另一个 S3 对象，使用驾驶执照的正面和背面调用 API。

```
aws textract analyze-id --document-pages '[{"S3Object":
{"Bucket":"bucket","Name":"name front"}}, {"S3Object":
{"Bucket":"bucket","Name":"name back"}}]' --region us-east-1
```

   如果您在 Windows 设备上访问 CLI，请使用双引号而不是单引号，并用反斜杠（即\）转义内部双引号，以解决可能遇到的任何解析器错误。有关示例，请参阅下面的内容：

```
aws textract analyze-id --document-pages "[{\"S3Object\":{\"Bucket\":\"bucket\",
\"Name\":\"name\"}}]" --region region
```

Python

以下示例从 S3 存储桶获取输入文件并运行AnalyzeID在其上操作，返回检测到的键值对。在下面的代码中，替换*bucket_name*将您的 S3 存储桶的名称用于*file_name*将包含存储桶中的文件的名称以及的值##有的名字region与您的账户关联。

```
import boto3

bucket_name = "bucket-name"
file_name = "file-name"
region = "region-name"

def analyze_id(region, bucket_name, file_name):

    textract_client = boto3.client('textract', region_name=region)
    response = textract_client.analyze_id(DocumentPages=[{"S3Object":
{"Bucket":bucket_name,"Name":file_name}}])

    for doc_fields in response['IdentityDocuments']:
        for id_field in doc_fields['IdentityDocumentFields']:
            for key, val in id_field.items():
                if "Type" in str(key):
                    print("Type: " + str(val['Text']))
            for key, val in id_field.items():
                if "ValueDetection" in str(key):
                    print("Value Detection: " + str(val['Text']))
            print()

analyze_id(region, bucket_name, file_name)
```

Java

以下示例从 S3 存储桶获取输入文件并运行AnalyzeID对其进行操作，返回检测到的数据。在函数 main 中，将以下值s3bucket和sourceDoc将包含您在步骤 2 中使用的 Amazon S3 存储桶名称和文档图像的名称。

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
```

```
package com.amazonaws.samples;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.textract.AmazonTextractClient;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.*;
import java.util.ArrayList;
import java.util.List;

public class AnalyzeIdentityDocument {

    public static void main(String[] args) {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <s3bucket><sourceDoc> \n\n" +
                "Where:\n" +
                "    s3bucket - the Amazon S3 bucket where the document is
 located. \n" +
                "    sourceDoc - the name of the document. \n";

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

        String s3bucket = "bucket-name"; //args[0];
        String sourceDoc = "sourcedoc-name";  //args[1];
        AmazonTextractClient textractClient = (AmazonTextractClient)
 AmazonTextractClientBuilder.standard()
                .withRegion(Regions.US_EAST_1)
                .build();

        getDocDetails(textractClient, s3bucket, sourceDoc);
    }

    public static void getDocDetails(AmazonTextractClient textractClient, String
 s3bucket, String sourceDoc ) {

       try {

            S3Object s3 = new S3Object();
            s3.setBucket(s3bucket);
            s3.setName(sourceDoc);
```

```
            com.amazonaws.services.textract.model.Document myDoc = new
 com.amazonaws.services.textract.model.Document();
            myDoc.setS3Object(s3);

            List<Document> list1 = new ArrayList();
            list1.add(myDoc);

            AnalyzeIDRequest idRequest = new AnalyzeIDRequest();
            idRequest.setDocumentPages(list1);

            AnalyzeIDResult result = textractClient.analyzeID(idRequest);
            List<IdentityDocument> docs =  result.getIdentityDocuments();
            for (IdentityDocument doc: docs) {

                List<IdentityDocumentField>idFields =
 doc.getIdentityDocumentFields();
                for (IdentityDocumentField field: idFields) {
                    System.out.println("Field type is "+
 field.getType().getText());
                    System.out.println("Field value is "+
 field.getValueDetection().getText());
                }
            }

       } catch (Exception e) {
            e.printStackTrace();
       }
    }
}
```

4. 这将为您提供的 JSON 输出`AnalyzeID`operation.

使用异步操作处理文档

Amazon Textract 可以检测和分析 PDF 或 TIFF 格式的多页文档中的文本。这包括发票和收据。多页文档处理是一项异步操作。异步处理文档对于处理大型多页文档非常有用。例如，包含超过 1,000 页的 PDF 文件需要一段时间才能处理。异步处理 PDF 文件允许应用程序在等待过程完成的同时完成其他任务。

本节介绍了如何使用 Amazon Textract 异步检测和分析多页或单页文档中的文本。多页文档必须为 PDF 或 TIFF 格式。使用异步操作处理的单页文档可以采用 JPEG、PNG、TIFF 或 PDF 格式。

您可以将 Amazon Textract 异步操作用于以下目的：

- 文本检测 — 您可以检测多页文档中的行和单词。异步操作是StartDocumentTextDetection和GetDocumentTextDetection. 有关更多信息，请参阅 检测文本。
- 文本分析 — 您可以识别多页文档上检测到的文本之间的关系。异步操作是StartDocumentAnalysis和GetDocumentAnalysis. 有关更多信息，请参阅 分析文档。
- 费用分析 — 您可以识别多页发票和收据的数据关系。Amazon Textract 将每张发票或多页文档的收据页面视为单个收据或发票。它不会将多页文档的上下文从一个页面保留到另一页。异步操作是StartExpenseAnalysis和GetExpenseAnalysis. 有关更多信息，请参阅 分析发票和收据。

主题

- 调用 Amazon Textract 异步操作
- 为异步操作配置 Amazon Textract
- 检测或分析多页文档中的文本
- Amazon Textract 结果通知

调用 Amazon Textract 异步操作

Amazon Textract 提供了一个异步 API，您可以用它来处理 PDF 或 TIFF 格式的多页文档。您还可以使用异步操作来处理 JPEG、PNG、TIFF 或 PDF 格式的单页文档。

本主题中的信息使用文本检测操作来展示如何使用 Amazon Textract 异步操作。同样的方法适用于的文本分析操作the section called “StartDocumentAnalysis”和the section called “GetDocumentAnalysis”. 它也适用于the section called “StartExpenseAnalysis”和the section called “GetExpenseAnalysis”.

有关示例，请参阅 检测或分析多页文档中的文本。

Amazon Textract 异步处理 Amazon S3 存储桶中的文档。你可以通过调用`Start`操作，例如StartDocumentTextDetection. 此请求的完成状态将发布到 Amazon Simple Notification Service (Amazon SNS) 主题。要从 Amazon SNS 主题获取完成状态，您可使用 Amazon Simple Queue Service (Amazon SQS) 队列或AWS Lambdafunction. 在获得完成状态之后，请调用 `Get` 操作 (如 GetDocumentTextDetection) 以获取请求的结果。

默认情况下，异步调用的结果将被加密并存储在 Amazon Textract 拥有的存储桶中 7 天，除非您使用操作指定 Amazon S3 存储桶`OutputConfig`参数。

下表显示了 Amazon Textract 支持的不同类型异步处理的相应启动和 Get 操作：

启动/获取亚马Amazon Textract 异步操作的API 操作

处理类型	启动 API	获取 API
文本检测	StartDocumentTextDetection	GetDocumentTextDetection
文本分析	StartDocumentAnalysis	GetDocumentAnalysis
费用分析	StartePense 分析	GetExpense 分析

对于使用的示例AWS Lambda函数，请参阅使用 Amazon Textract 进行大规模文档处理.

下图显示了检测 Amazon S3 存储桶中的文档图像中的文档文本的过程。在此图中，Amazon SQS 队列将从 Amazon SNS 主题获取完成状态。

上图显示的过程与分析文本和发票/收据的过程相同。你通过打电话开始分析文本the section called “StartDocumentAnalysis”然后通过致电开始分析发票/收据the section called “StartExpenseAnalysis”你可以通过打电话获得结果the section called “GetDocumentAnalysis”要么the section called “GetExpenseAnalysis”分别。

开始文本检测

您可通过调用启动 Amazon Textract 文本检测请求StartDocumentTextDetection. 下面是由 `StartDocumentTextDetection` 传递的 JSON 请求的示例。

```
{
```

```
    "DocumentLocation": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "image.pdf"
        }
    },
    "ClientRequestToken": "DocumentDetectionToken",
    "NotificationChannel": {
        "SNSTopicArn": "arn:aws:sns:us-east-1:nnnnnnnnnn:topic",
        "RoleArn": "arn:aws:iam::nnnnnnnnnn:role/roleTopic"
    },
    "JobTag": "Receipt"
}
```

输入参数DocumentLocation提供文档文件名和要从中检索此文档的 Amazon S3 存储桶。NotificationChannel包含 Amazon Textract 在文本检测请求完成时通知的 Amazon SNS 主题的 Amazon Resource Name (ARN) 的资源名称 (ARN)。Amazon SNS 主题必须与您调用的 Amazon Textract 终端节点位于同一 AWS 区域。NotificationChannel还包含允许 Amazon Textract 向 Amazon SNS 主题进行发布的角色的 ARN。您可通过创建 IAM 服务角色为 Amazon Textract 发布您的 Amazon SNS 主题授予权限。有关更多信息，请参阅 为异步操作配置 Amazon Textract。

您也可以指定可选的输入参数，JobTag，使您能够标识处于已发布到 Amazon SNS 主题的完成状态的任务或一组任务。例如，您可以使用JobTag以确定正在处理的文件类型，例如纳税表或收据。

为防止分析任务意外重复，您可以选择性地提供幂等令牌 ClientRequestToken。如果你提供了一个值ClientRequestToken，Start操作返回相同JobId对于多个相同的呼叫Start操作，例如StartDocumentTextDetection. ClientRequestToken 令牌的使用期限为 7 天。7 天后，您可以重复使用它。如果您在令牌使用期限内重复使用令牌，则会出现以下情况：

- 如果您对相同的 Start 操作和相同的输入参数重复使用此令牌，则会返回相同的 JobId。此任务不会再次执行，Amazon Textract 不会向注册的 Amazon SNS 主题发送完成状态。
- 如果您对相同的 Start 操作重复使用此令牌，并且只进行了细微的输入参数更改，则会引发 idempotentparametermismatchexception (HTTP 状态代码：400) 异常。
- 如果您对其他 Start 操作重复使用此令牌，操作将成功。

另一个可选参数是OutputConfig，它允许你调整输出的放置位置。默认情况下，Amazon Textract 将在内部存储结果，并且只能通过获取 API 操作访问。与OutputConfig启用后，您可以设置输出将发送到的存储桶的名称和结果的文件前缀，您可以在其中下载结果。此外，您还可以设置KMSKeyID用

于加密输出的客户管理密钥的参数。如果不设置此参数，Amazon Textract 将使用AWS 托管式密钥适用于 Amazon S3

> **Note**
>
> 在使用此参数之前，请确保您拥有输出存储桶的 PutObject 权限。此外，请确保您拥有解密、ReEncrypt、GenerateDataKey 和 describeKey 权限AWS KMS关键如果你决定使用它。

对 StartDocumentTextDetection 操作的响应是作业标识符 (JobId)。使用JobId使用可跟踪请求并在 Amazon Textract 将完成状态发布到 Amazon SNS 主题之后获取分析结果。以下是示例：

```
{"JobId":"270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3"}
```

如果你同时开始太多的工作，请致电StartDocumentTextDetection提高LimitExceededException异常（HTTP 状态代码：400），直到并发运行的任务数量低于 Amazon Textract 服务限制。

如果您发现活动量猛增时会引发 LLimitExceededException 异常，请考虑使用 Amazon SQS 队列管理传入请求。联系人AWS如果 Amazon SQS 队列无法管理平均数量的并发请求，您仍会收到，请联系 Support。LimitExceededException异常情况。

获取 Amazon Textract 分析请求的完成状态

Amazon Textract 会向注册的 Amazon SNS 主题发送分析完成通知。通知将在 JSON 字符串中包含操作的任务标识符和完成状态。成功的文本检测请求有SUCCEEDED状态。例如，以下结果展示了成功处理文本检测任务。

```
{
    "JobId": "642492aea78a86a40665555dc375ee97bc963f342b29cd05030f19bd8fd1bc5f",
    "Status": "SUCCEEDED",
    "API": "StartDocumentTextDetection",
    "JobTag": "Receipt",
    "Timestamp": 1543599965969,
    "DocumentLocation": {
        "S3ObjectName": "document",
        "S3Bucket": "bucket"
    }
}
```

有关更多信息，请参阅 Amazon Textract 结果通知。

要获取 Amazon Textract 向 Amazon SNS 主题发布的状态信息，请使用以下选项之一：

- AWS Lambda— 您可以订阅AWS Lambda您写入到 Amazon SNS 主题的函数。此函数在 Amazon Textract 通知 Amazon SNS 主题请求已完成时调用。如果您希望服务器端代码处理文本检测请求的结果，请使用 Lambda 函数。例如，在将信息返回到客户端应用程序之前，您可能希望使用服务器端代码来注释图像或创建有关检测到的文本的报告。
- Amazon SQS— 您可以为 Amazon SQS 队列订阅 Amazon SNS 主题。您随后将轮询 Amazon SQS 队列以检索 Amazon Textract 在文本检测请求完成后发布的完成状态。有关更多信息，请参阅 检测或分析多页文档中的文本。如果您希望仅从客户端应用程序调用 Amazon Textract 操作，请使用 Amazon SQS 队列。

> **Important**
>
> 我们建议不要通过反复调用 Amazon Textract 来获取请求完成状态。Getoperation. 这是因为 Amazon Textract 限制了Get操作（如果提出的请求过多）。如果您同时处理多个文档，那么监控一个 SQS 队列中的完成通知比为每个任务的状态分别轮询 Amazon Textract 更加简单有效。

获取 Amazon Textract 文本检测结果

要获取文本检测请求的结果，请先确保从 Amazon SNS 主题检索的完成状态为SUCCEEDED. 然后调用 `GetDocumentTextDetection`，它将传递从 `StartDocumentTextDetection` 返回的 `JobId` 值。请求 JSON 类似于以下示例：

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3",
    "MaxResults": 10,
    "SortBy": "TIMESTAMP"
}
```

`JobId`是文本检测操作的标识符。由于文本检测可以生成大量数据，请使用`MaxResults`以指定要在单个结果中返回的最大数量Getoperation. 的默认值`MaxResults`是 1,000。如果您指定的值大于 1,000，则仅返回 1,000 个结果。如果此操作未返回所有结果，将返回下一页的分页令牌。要获取下一页结果，请在`NextToken`参数。

> **Note**
>
> Amazon Textract 将保留 7 天的异步操作结果。此时间过后，您无法检索结果。

这些区域有：GetDocumentTextDetection操作响应 JSON 类似于以下内容。返回的检测到的页面总数DocumentMetadata. 检测到的文本将在Blocks数组。有关的信息Block对象，请参阅文本检测和文档分析响应对象.

```
{
    "DocumentMetadata": {
        "Pages": 1
    },
    "JobStatus": "SUCCEEDED",
    "Blocks": [
        {
            "BlockType": "PAGE",
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Height": 1.0,
                    "Left": 0.0,
                    "Top": 0.0
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.0
                    },
                    {
                        "X": 1.0,
                        "Y": 0.0
                    },
                    {
                        "X": 1.0,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
```

```
            },
            "Id": "64533157-c47e-401a-930e-7ca1bb3ac3fa",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "4297834d-dcb1-413b-8908-3b96866ebbb5",
                        "1d85ba24-2877-4d09-b8b2-393833d769e9",
                        "193e9c47-fd87-475a-ba09-3fda210d8784",
                        "bd8aeb62-961b-4b47-b78a-e4ed9eeecd0f"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 53.301639556884766,
            "Text": "ellooworio",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.9999999403953552,
                    "Height": 0.5365243554115295,
                    "Left": 0.0,
                    "Top": 0.46347561478614807
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 0.9999999403953552,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 0.9999999403953552,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
```

```
            },
            "Id": "4297834d-dcb1-413b-8908-3b96866ebbb5",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "170c3eb9-5155-4bec-8c44-173bba537e70"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 89.15632629394531,
            "Text": "He llo,",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33642634749412537,
                    "Height": 0.49159330129623413,
                    "Left": 0.13885067403316498,
                    "Top": 0.17169663310050964
                },
                "Polygon": [
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
                        "Y": 0.6632899641990662
                    },
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.6632899641990662
                    }
                ]
            },
            "Id": "1d85ba24-2877-4d09-b8b2-393833d769e9",
            "Relationships": [
```

```
                {
                    "Type": "CHILD",
                    "Ids": [
                        "516ae823-3bab-4f9a-9d74-ad7150d128ab",
                        "6bcf4ea8-bbe8-4686-91be-b98dd63bc6a6"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 82.44834899902344,
            "Text": "worlo",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33182239532470703,
                    "Height": 0.3766750991344452,
                    "Left": 0.5091826915740967,
                    "Top": 0.23131252825260162
                },
                "Polygon": [
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.607987642288208
                    },
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.607987642288208
                    }
                ]
            },
            "Id": "193e9c47-fd87-475a-ba09-3fda210d8784",
            "Relationships": [
                {
                    "Type": "CHILD",
```

```
                    "Ids": [
                        "ed135c3b-35dd-4085-8f00-26aedab0125f"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 88.50325775146484,
            "Text": "world",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.35004907846450806,
                    "Height": 0.19635874032974243,
                    "Left": 0.527581512928009,
                    "Top": 0.30100569128990173
                },
                "Polygon": [
                    {
                        "X": 0.527581512928009,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.49736443161964417
                    },
                    {
                        "X": 0.527581512928009,
                        "Y": 0.49736443161964417
                    }
                ]
            },
            "Id": "bd8aeb62-961b-4b47-b78a-e4ed9eeecd0f",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "9e28834d-798e-4a62-8862-a837dfd895a6"
                    ]
```

```
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 53.301639556884766,
            "Text": "ellooworio",
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Height": 0.5365243554115295,
                    "Left": 0.0,
                    "Top": 0.46347561478614807
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 1.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 1.0,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
            },
            "Id": "170c3eb9-5155-4bec-8c44-173bba537e70",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 88.46246337890625,
            "Text": "He",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.15350718796253204,
```

```
                    "Height": 0.29955607652664185,
                    "Left": 0.13885067403316498,
                    "Top": 0.21856294572353363
                },
                "Polygon": [
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.21856294572353363
                    },
                    {
                        "X": 0.292357861995697,
                        "Y": 0.21856294572353363
                    },
                    {
                        "X": 0.292357861995697,
                        "Y": 0.5181190371513367
                    },
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.5181190371513367
                    }
                ]
            },
            "Id": "516ae823-3bab-4f9a-9d74-ad7150d128ab",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 89.8501968383789,
            "Text": "llo,",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.17724157869815826,
                    "Height": 0.49159327149391174,
                    "Left": 0.2980354428291321,
                    "Top": 0.17169663310050964
                },
                "Polygon": [
                    {
                        "X": 0.2980354428291321,
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
```

```
                    "Y": 0.17169663310050964
                },
                {
                    "X": 0.47527703642845154,
                    "Y": 0.6632899045944214
                },
                {
                    "X": 0.2980354428291321,
                    "Y": 0.6632899045944214
                }
            ]
        },
        "Id": "6bcf4ea8-bbe8-4686-91be-b98dd63bc6a6",
        "Page": 1
    },
    {
        "BlockType": "WORD",
        "Confidence": 82.44834899902344,
        "Text": "worlo",
        "Geometry": {
            "BoundingBox": {
                "Width": 0.33182239532470703,
                "Height": 0.3766750991344452,
                "Left": 0.5091826915740967,
                "Top": 0.23131252825260162
            },
            "Polygon": [
                {
                    "X": 0.5091826915740967,
                    "Y": 0.23131252825260162
                },
                {
                    "X": 0.8410050868988037,
                    "Y": 0.23131252825260162
                },
                {
                    "X": 0.8410050868988037,
                    "Y": 0.607987642288208
                },
                {
                    "X": 0.5091826915740967,
                    "Y": 0.607987642288208
                }
            ]
```

```
            },
            "Id": "ed135c3b-35dd-4085-8f00-26aedab0125f",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 88.50325775146484,
            "Text": "world",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.35004907846450806,
                    "Height": 0.19635874032974243,
                    "Left": 0.527581512928009,
                    "Top": 0.30100569128990173
                },
                "Polygon": [
                    {
                        "X": 0.527581512928009,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.49736443161964417
                    },
                    {
                        "X": 0.527581512928009,
                        "Y": 0.49736443161964417
                    }
                ]
            },
            "Id": "9e28834d-798e-4a62-8862-a837dfd895a6",
            "Page": 1
        }
    ]
}
```

为异步操作配置 Amazon Textract

您将介绍如何将 Amazon Textract 配置为与 Amazon Simple Notification Service (Amazon SNS) 主题和 Amazon Simple Queue Service (Amazon SQS) 队列组合使用。

> **Note**
>
> 如果您使用这些说明设置检测或分析多页文档中的文本例如，您不需要执行步骤 3 — 6。示例包括用于创建和配置 Amazon SNS 主题和 Amazon SQS 队列的代码。

要配置 Amazon Textract

1. 设置AWS用于访问 Amazon Textract 的账户。有关更多信息，请参阅 第 1 步：设置 AWS 账户并创建 IAM 用户。

   确保用户具有至少以下权限：

   - AmazonTextractFullAccess
   - AmazonS3ReadOnlyAccess
   - AmazonSNSFullAccess
   - AmazonSQSFullAccess
2. 安装和配置所需的 AWS 开发工具包。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。
3. 创建 Amazon SNS 主题. 在主题名称前加上卓越亚马逊提取. 记下主题的 Amazon 资源名称 (ARN)。确保主题与AWS您将与 AWS 账户配合使用的终端节点。
4. 创建 Amazon SQS 标准队列通过使用Amazon SQS 控制台. 记录队列 ARN。
5. 为队列订阅主题（您在步骤 3 中创建）。
6. 为 Amazon SNS 主题授予权限，以向 Amazon SQS 队列发送消息.
7. 创建 IAM 服务角色以授予 Amazon Textract 访问您的 Amazon SNS 主题的权限。记下服务角色的 Amazon 资源名称 (ARN)。有关更多信息，请参阅 授予 Amazon Textract 访问您的 Amazon SNS 主题。
8. 添加以下内联策略适用于您在步骤 1 中创建的 IAM 用户。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "MySid",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "Service role ARN from step 7"
        }
    ]
}
```

为内联策略提供一个名称。

9. 现在，您就可以运行中的示例了检测或分析多页文档中的文本.

授予 Amazon Textract 访问您的 Amazon SNS 主题

异步操作完成后，Amazon Textract 需要向您的 Amazon SNS 主题发送消息的权限。您使用 IAM 服务角色授予 Amazon Textract 访问 Amazon SNS 主题的权限。

创建 Amazon SNS 主题时，必须在主题名称前加上**AmazonTextract**— 例如：**AmazonTextractMyTopicName**.

1. 登录 IAM 控制台 (https://console.aws.amazon.com/iam)。
2. 在导航窗格中，选择角色。
3. 选择 Create role（创建角色）。
4. 适用于选择受信任实体的类型，选择AWS 服务.
5. 适用于选择将使用此角色的服务，选择Textract.
6. 选择 Next:。Permissions (下一步：权限)。
7. 验证AmazonTextractServiceRole策略已包含在附加的策略列表中。要在列表中显示策略，请在筛选策略.
8. 选择 Next:。标签。
9. 您不需要添加标签，因此，请选择后续：审核。
10. 在审核部分中，对于角色名称，键入角色的名称（例如，TextractRole）。In角色描述，请更新角色的描述，然后选择创建角色.
11. 选择新角色以打开角色的详细信息页面。
12. 在摘要中，复制角色 ARN 值并保存它。

13. 选择 Trust Relationships (信任关系)。
14. 选择编辑信任关系，并确保信任政策如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "textract.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

15. 选择 Update Trust Policy。

检测或分析多页文档中的文本

此过程向您展示如何使用 Amazon Textract 检测操作、存储在 Amazon S3 存储桶中的文档、Amazon SNS 主题和 Amazon SQS 队列来检测或分析多页文档中的文本。多页文档处理是一项异步操作。有关更多信息，请参阅 调用 Amazon Textract 异步操作。

您可以选择希望代码执行的处理类型：文本检测、文本分析或费用分析。

处理结果将在数组中返回the section called “Block”对象，具体取决于您使用的处理类型。

要检测多页文档中的文本或分析多页文档，请执行以下操作：

1. 创建 Amazon SNS 主题和 Amazon SQS 队列。
2. 为队列订阅主题。
3. 为向队列发送消息的主题授予权限。
4. 开始处理文档。对您选择的分析类型使用适当的操作：
   - StartDocumentTextDetection用于文本检测任务。
   - StartDocumentAnalysis用于文本分析任务。
   - StartExpenseAnalysis用于支出分析任务。
5. 从 Amazon SQS 队列获取完成状态。示例代码跟踪作业标识符 (`JobId`) 这是由`Startoperation`. 它仅获取与从完成状态读取的任务标识符匹配的结果。如果其他应用程序使用的是同一队列和

主题，这一点很重要。为简便起见，该示例会删除不匹配的任务。请考虑将删除的任务添加到 Amazon SQS 死信队列以进行进一步调查。

6. 通过为所选分析类型调用适当的操作来获取并显示处理结果：
   - GetDocumentTextDetection用于文本检测任务。
   - GetDocumentAnalysis用于文本分析任务。
   - GetExpenseAnalysis用于支出分析任务。
7. 删除 Amazon SNS 主题和 Amazon SQS 队列。

执行异步操作

此过程的示例代码提供在 Java、Python 和AWS CLI. 在开始之前，请安装适当的AWSSDK。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。

检测或分析多页文档中的文本

1. 配置用户访问 Amazon Textract 并配置 Amazon Textract 访问 Amazon SNS 的权限。有关更多信息，请参阅 为异步操作配置 Amazon Textract。要完成此过程，您需要一个 PDF 格式的多页文档文件。请跳过步骤 3 — 6，因为示例代码将创建并配置 Amazon SNS 主题和 Amazon SQS 队列。如果很复杂t在 CLI 示例中，您无需设置 SQS 队列。
2. 将 PDF 或 TIFF 格式的多页文档文件上传到 Amazon S3 存储桶。（也可以处理 JPEG、PNG、TIFF 或 PDF 格式的单页文档）。

   有关说明，请参阅将对象上传到 Amazon S3中的Amazon Simple Storage Service 用户指南.
3. 使用以下命令适用于 Java 的 AWS SDK、SDK for Python (Boto3)，或AWS CLI用于检测文本或分析多页文档中的文本的代码。在`main`函数：
   - 替换的值`roleArn`使用您保存的 IAM 角色 ARN授予 Amazon Textract 访问您的 Amazon SNS 主题.
   - 替换的值`bucket`和`document`将包含您在步骤 2 中指定的存储桶和文档文件名。
   - 替换的值`type`的输入参数`ProcessDocument`函数与您要执行的处理类型。使用`ProcessType.DETECTION`来检测文本。使用`ProcessType.ANALYSIS`来分析文本。
   - 对于 Python 示例，请替换`region_name`您的客户运营所在的地区。

   对于AWS CLI例如，执行以下操作：

- 打电话时StartDocumentTextDetection，替换的值bucket-name使用 S3 存储桶的名称，然后替换file-name将与您在步骤 2 中指定的文件名称。通过替换来指定存储桶的区域region-name用你所在地区的名称。请注意，CLI 示例没有使用 SQS。
- 打电话时GetDocumentTextDetection替换job-id-number使用job-id返回方StartDocumentTextDetection. 通过替换来指定存储桶的区域region-name用你所在地区的名称。

Java

```
package com.amazonaws.samples;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.amazonaws.auth.policy.Condition;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.SQSActions;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.QueueAttributeName;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
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.DocumentLocation;
import com.amazonaws.services.textract.model.DocumentMetadata;
import com.amazonaws.services.textract.model.GetDocumentAnalysisRequest;
import com.amazonaws.services.textract.model.GetDocumentAnalysisResult;
import com.amazonaws.services.textract.model.GetDocumentTextDetectionRequest;
```

```
import com.amazonaws.services.textract.model.GetDocumentTextDetectionResult;
import com.amazonaws.services.textract.model.NotificationChannel;
import com.amazonaws.services.textract.model.Relationship;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.StartDocumentAnalysisRequest;
import com.amazonaws.services.textract.model.StartDocumentAnalysisResult;
import com.amazonaws.services.textract.model.StartDocumentTextDetectionRequest;
import com.amazonaws.services.textract.model.StartDocumentTextDetectionResult;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;;
public class DocumentProcessor {

    private static String sqsQueueName=null;
    private static String snsTopicName=null;
    private static String snsTopicArn = null;
    private static String roleArn= null;
    private static String sqsQueueUrl = null;
    private static String sqsQueueArn = null;
    private static String startJobId = null;
    private static String bucket = null;
    private static String document = null;
    private static AmazonSQS sqs=null;
    private static AmazonSNS sns=null;
    private static AmazonTextract textract = null;

    public enum ProcessType {
        DETECTION,ANALYSIS
    }

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

        String document = "document";
        String bucket = "bucket";
        String roleArn="role";

        sns = AmazonSNSClientBuilder.defaultClient();
        sqs= AmazonSQSClientBuilder.defaultClient();
        textract=AmazonTextractClientBuilder.defaultClient();

        CreateTopicandQueue();
        ProcessDocument(bucket,document,roleArn,ProcessType.DETECTION);
        DeleteTopicandQueue();
        System.out.println("Done!");
```

```
   }
   // Creates an SNS topic and SQS queue. The queue is subscribed to the
topic.
   static void CreateTopicandQueue()
   {
       //create a new SNS topic
       snsTopicName="AmazonTextractTopic" +
Long.toString(System.currentTimeMillis());
       CreateTopicRequest createTopicRequest = new
CreateTopicRequest(snsTopicName);
       CreateTopicResult createTopicResult =
sns.createTopic(createTopicRequest);
       snsTopicArn=createTopicResult.getTopicArn();

       //Create a new SQS Queue
       sqsQueueName="AmazonTextractQueue" +
Long.toString(System.currentTimeMillis());
       final CreateQueueRequest createQueueRequest = new
CreateQueueRequest(sqsQueueName);
       sqsQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();
       sqsQueueArn = sqs.getQueueAttributes(sqsQueueUrl,
Arrays.asList("QueueArn")).getAttributes().get("QueueArn");

       //Subscribe SQS queue to SNS topic
       String sqsSubscriptionArn = sns.subscribe(snsTopicArn, "sqs",
sqsQueueArn).getSubscriptionArn();

       // Authorize queue
         Policy policy = new Policy().withStatements(
                 new Statement(Effect.Allow)
                 .withPrincipals(Principal.AllUsers)
                 .withActions(SQSActions.SendMessage)
                 .withResources(new Resource(sqsQueueArn))
                 .withConditions(new
Condition().withType("ArnEquals").withConditionKey("aws:SourceArn").withValues(snsTopic
                 );


         Map queueAttributes = new HashMap();
         queueAttributes.put(QueueAttributeName.Policy.toString(),
policy.toJson());
         sqs.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueUrl,
queueAttributes));
```

```
         System.out.println("Topic arn: " + snsTopicArn);
         System.out.println("Queue arn: " + sqsQueueArn);
         System.out.println("Queue url: " + sqsQueueUrl);
         System.out.println("Queue sub arn: " + sqsSubscriptionArn );
     }
   static void DeleteTopicandQueue()
   {
       if (sqs !=null) {
           sqs.deleteQueue(sqsQueueUrl);
           System.out.println("SQS queue deleted");
       }

       if (sns!=null) {
           sns.deleteTopic(snsTopicArn);
           System.out.println("SNS topic deleted");
       }
   }

   //Starts the processing of the input document.
   static void ProcessDocument(String inBucket, String inDocument, String
inRoleArn, ProcessType type) throws Exception
   {
       bucket=inBucket;
       document=inDocument;
       roleArn=inRoleArn;

       switch(type)
       {
           case DETECTION:
               StartDocumentTextDetection(bucket, document);
               System.out.println("Processing type: Detection");
               break;
           case ANALYSIS:
               StartDocumentAnalysis(bucket,document);
               System.out.println("Processing type: Analysis");
               break;
           default:
               System.out.println("Invalid processing type. Choose Detection or
Analysis");
               throw new Exception("Invalid processing type");

       }
```

```
        System.out.println("Waiting for job: " + startJobId);
        //Poll queue for messages
        List<Message> messages=null;
        int dotLine=0;
        boolean jobFound=false;

        //loop until the job status is published. Ignore other messages in
queue.
        do{
            messages = sqs.receiveMessage(sqsQueueUrl).getMessages();
            if (dotLine++<40){
                System.out.print(".");
            }else{
                System.out.println();
                dotLine=0;
            }

            if (!messages.isEmpty()) {
                //Loop through messages received.
                for (Message message: messages) {
                    String notification = message.getBody();

                    // Get status and job id from notification.
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found was " + operationJobId);
                    // Found job. Get the results and display.
                    if(operationJobId.asText().equals(startJobId)){
                        jobFound=true;
                        System.out.println("Job id: " + operationJobId );
                        System.out.println("Status : " +
operationStatus.toString());
                        if (operationStatus.asText().equals("SUCCEEDED")){
                            switch(type)
                            {
                                case DETECTION:
                                    GetDocumentTextDetectionResults();
```

```
                                    break;
                                case ANALYSIS:
                                    GetDocumentAnalysisResults();
                                    break;
                                default:
                                    System.out.println("Invalid processing type.
Choose Detection or Analysis");
                                    throw new Exception("Invalid processing
type");

                            }
                        }
                        else{
                            System.out.println("Document analysis failed");
                        }


sqs.deleteMessage(sqsQueueUrl,message.getReceiptHandle());
                    }

                    else{
                        System.out.println("Job received was not job " +
startJobId);
                        //Delete unknown message. Consider moving message to
dead letter queue

sqs.deleteMessage(sqsQueueUrl,message.getReceiptHandle());
                    }
                }
            }
            else {
                Thread.sleep(5000);
            }
        } while (!jobFound);

        System.out.println("Finished processing document");
    }

    private static void StartDocumentTextDetection(String bucket, String
document) throws Exception{

        //Create notification channel
        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
```

```
                .withRoleArn(roleArn);

        StartDocumentTextDetectionRequest req = new
StartDocumentTextDetectionRequest()
                .withDocumentLocation(new DocumentLocation()
                    .withS3Object(new S3Object()
                        .withBucket(bucket)
                        .withName(document)))
                .withJobTag("DetectingText")
                .withNotificationChannel(channel);

        StartDocumentTextDetectionResult startDocumentTextDetectionResult =
textract.startDocumentTextDetection(req);
        startJobId=startDocumentTextDetectionResult.getJobId();
    }

 //Gets the results of processing started by StartDocumentTextDetection
    private static void GetDocumentTextDetectionResults() throws Exception{
        int maxResults=1000;
        String paginationToken=null;
        GetDocumentTextDetectionResult response=null;
        Boolean finished=false;

        while (finished==false)
        {
            GetDocumentTextDetectionRequest documentTextDetectionRequest= new
GetDocumentTextDetectionRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);
            response =
textract.getDocumentTextDetection(documentTextDetectionRequest);
            DocumentMetadata documentMetaData=response.getDocumentMetadata();

            System.out.println("Pages: " +
documentMetaData.getPages().toString());

            //Show blocks information
            List<Block> blocks= response.getBlocks();
            for (Block block : blocks) {
                DisplayBlockInfo(block);
            }
            paginationToken=response.getNextToken();
            if (paginationToken==null)
```

```
                finished=true;

        }

    }

    private static void StartDocumentAnalysis(String bucket, String document)
throws Exception{
        //Create notification channel
        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
                .withRoleArn(roleArn);

        StartDocumentAnalysisRequest req = new StartDocumentAnalysisRequest()
                .withFeatureTypes("TABLES","FORMS")
                .withDocumentLocation(new DocumentLocation()
                    .withS3Object(new S3Object()
                        .withBucket(bucket)
                        .withName(document)))
                .withJobTag("AnalyzingText")
                .withNotificationChannel(channel);

        StartDocumentAnalysisResult startDocumentAnalysisResult =
textract.startDocumentAnalysis(req);
        startJobId=startDocumentAnalysisResult.getJobId();
    }
    //Gets the results of processing started by StartDocumentAnalysis
    private static void GetDocumentAnalysisResults() throws Exception{

        int maxResults=1000;
        String paginationToken=null;
        GetDocumentAnalysisResult response=null;
        Boolean finished=false;

        //loops until pagination token is null
        while (finished==false)
        {
            GetDocumentAnalysisRequest documentAnalysisRequest= new
GetDocumentAnalysisRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);

            response = textract.getDocumentAnalysis(documentAnalysisRequest);
```

```
            DocumentMetadata documentMetaData=response.getDocumentMetadata();

            System.out.println("Pages: " +
documentMetaData.getPages().toString());

            //Show blocks, confidence and detection times
            List<Block> blocks= response.getBlocks();

            for (Block block : blocks) {
                DisplayBlockInfo(block);
            }
            paginationToken=response.getNextToken();
            if (paginationToken==null)
                finished=true;
        }

    }
    //Displays Block information for text detection and text analysis
    private static void DisplayBlockInfo(Block block) {
        System.out.println("Block Id : " + block.getId());
        if (block.getText()!=null)
            System.out.println("\tDetected text: " + block.getText());
        System.out.println("\tType: " + block.getBlockType());

        if (block.getBlockType().equals("PAGE") !=true) {
            System.out.println("\tConfidence: " +
block.getConfidence().toString());
        }
        if(block.getBlockType().equals("CELL"))
        {
            System.out.println("\tCell information:");
            System.out.println("\t\tColumn: " + block.getColumnIndex());
            System.out.println("\t\tRow: " + block.getRowIndex());
            System.out.println("\t\tColumn span: " + block.getColumnSpan());
            System.out.println("\t\tRow span: " + block.getRowSpan());

        }

        System.out.println("\tRelationships");
        List<Relationship> relationships=block.getRelationships();
        if(relationships!=null) {
            for (Relationship relationship : relationships) {
                System.out.println("\t\tType: " + relationship.getType());
```

```
                System.out.println("\t\tIDs: " +
 relationship.getIds().toString());
            }
        } else {
            System.out.println("\t\tNo related Blocks");
        }

        System.out.println("\tGeometry");
        System.out.println("\t\tBounding Box: " +
 block.getGeometry().getBoundingBox().toString());
        System.out.println("\t\tPolygon: " +
 block.getGeometry().getPolygon().toString());

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

        System.out.println("\tEntity Types");
        if(entityTypes!=null) {
            for (String entityType : entityTypes) {
                System.out.println("\t\tEntity Type: " + entityType);
            }
        } else {
            System.out.println("\t\tNo 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("\tPage: " + block.getPage());
        System.out.println();
    }
}
```

AWS CLI

该AWS CLI命令启动异步检测指定文档中的文本。它返回job-id这可以用来检索检测结果。

```
aws textract start-document-text-detection --document-location
```

```
"{\"S3Object\":{\"Bucket\":\"bucket-name\",\"Name\":\"file-name\"}}" --
region region-name
```

该AWS CLI命令返回 Amazon Textract 异步操作的结果（如果提供了job-id.

```
aws textract get-document-text-detection --region region-name --job-id job-id-
number
```

如果您在 Windows 设备上访问 CLI，请使用双引号而不是单引号，并用反斜杠（即\）转义内部双引号，以解决可能遇到的任何解析器错误。有关示例，请参阅下文

```
aws textract start-document-text-detection --document-location "{\"S3Object\":
{\"Bucket\":\"bucket\",\"Name\":\"document\"}}" --region region-name
```

Python

```
import boto3
import json
import sys
import time


class ProcessType:
    DETECTION = 1
    ANALYSIS = 2


class DocumentProcessor:
    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

    def __init__(self, role, bucket, document, region):
        self.roleArn = role
```

```
        self.bucket = bucket
        self.document = document
        self.region_name = region

        self.textract = boto3.client('textract', region_name=self.region_name)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')

    def ProcessDocument(self, type):
        jobFound = False

        self.processType = type
        validType = False

        # Determine which type of processing to perform
        if self.processType == ProcessType.DETECTION:
            response = self.textract.start_document_text_detection(
                DocumentLocation={'S3Object': {'Bucket': self.bucket, 'Name':
self.document}},
                NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})
            print('Processing type: Detection')
            validType = True

        if self.processType == ProcessType.ANALYSIS:
            response = self.textract.start_document_analysis(
                DocumentLocation={'S3Object': {'Bucket': self.bucket, 'Name':
self.document}},
                FeatureTypes=["TABLES", "FORMS"],
                NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})
            print('Processing type: Analysis')
            validType = True

        if validType == False:
            print("Invalid processing type. Choose Detection or Analysis.")
            return

        print('Start Job Id: ' + response['JobId'])
        dotLine = 0
        while jobFound == False:
            sqsResponse = self.sqs.receive_message(QueueUrl=self.sqsQueueUrl,
MessageAttributeNames=['ALL'],
                                                   MaxNumberOfMessages=10)
```

```
            if sqsResponse:

                if 'Messages' not in sqsResponse:
                    if dotLine < 40:
                        print('.', end='')
                        dotLine = dotLine + 1
                    else:
                        print()
                        dotLine = 0
                    sys.stdout.flush()
                    time.sleep(5)
                    continue

                for message in sqsResponse['Messages']:
                    notification = json.loads(message['Body'])
                    textMessage = json.loads(notification['Message'])
                    print(textMessage['JobId'])
                    print(textMessage['Status'])
                    if str(textMessage['JobId']) == response['JobId']:
                        print('Matching Job Found:' + textMessage['JobId'])
                        jobFound = True
                        self.GetResults(textMessage['JobId'])
                        self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,

ReceiptHandle=message['ReceiptHandle'])
                    else:
                        print("Job didn't match:" +
                              str(textMessage['JobId']) + ' : ' +
str(response['JobId']))
                    # Delete the unknown message. Consider sending to dead
letter queue
                    self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,

ReceiptHandle=message['ReceiptHandle'])

        print('Done!')

    def CreateTopicandQueue(self):

        millis = str(int(round(time.time() * 1000)))

        # Create SNS topic
        snsTopicName = "AmazonTextractTopic" + millis
```

```
        topicResponse = self.sns.create_topic(Name=snsTopicName)
        self.snsTopicArn = topicResponse['TopicArn']

        # create SQS queue
        sqsQueueName = "AmazonTextractQueue" + millis
        self.sqs.create_queue(QueueName=sqsQueueName)
        self.sqsQueueUrl = self.sqs.get_queue_url(QueueName=sqsQueueName)
['QueueUrl']

        attribs = self.sqs.get_queue_attributes(QueueUrl=self.sqsQueueUrl,
                                                AttributeNames=['QueueArn'])
['Attributes']

        sqsQueueArn = attribs['QueueArn']

        # Subscribe SQS queue to SNS topic
        self.sns.subscribe(
            TopicArn=self.snsTopicArn,
            Protocol='sqs',
            Endpoint=sqsQueueArn)

        # Authorize SNS to write SQS queue
        policy = """{{
  "Version":"2012-10-17",
  "Statement":[
    {{
      "Sid":"MyPolicy",
      "Effect":"Allow",
      "Principal" : {{"AWS" : "*"}},
      "Action":"SQS:SendMessage",
      "Resource": "{}",
      "Condition":{{
        "ArnEquals":{{
          "aws:SourceArn": "{}"
        }}
      }}
    }}
  ]
}}""".format(sqsQueueArn, self.snsTopicArn)

        response = self.sqs.set_queue_attributes(
            QueueUrl=self.sqsQueueUrl,
            Attributes={
```

```
                'Policy': policy
            })

    def DeleteTopicandQueue(self):
        self.sqs.delete_queue(QueueUrl=self.sqsQueueUrl)
        self.sns.delete_topic(TopicArn=self.snsTopicArn)

    # Display information about a block
    def DisplayBlockInfo(self, block):

        print("Block Id: " + block['Id'])
        print("Type: " + block['BlockType'])
        if 'EntityTypes' in block:
            print('EntityTypes: {}'.format(block['EntityTypes']))

        if 'Text' in block:
            print("Text: " + block['Text'])

        if block['BlockType'] != 'PAGE':
            print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

        print('Page: {}'.format(block['Page']))

        if block['BlockType'] == 'CELL':
            print('Cell Information')
            print('\tColumn: {} '.format(block['ColumnIndex']))
            print('\tRow: {}'.format(block['RowIndex']))
            print('\tColumn span: {} '.format(block['ColumnSpan']))
            print('\tRow span: {}'.format(block['RowSpan']))

            if 'Relationships' in block:
                print('\tRelationships: {}'.format(block['Relationships']))

        print('Geometry')
        print('\tBounding Box: {}'.format(block['Geometry']['BoundingBox']))
        print('\tPolygon: {}'.format(block['Geometry']['Polygon']))

        if block['BlockType'] == 'SELECTION_ELEMENT':
            print('    Selection element detected: ', end='')
            if block['SelectionStatus'] == 'SELECTED':
                print('Selected')
            else:
                print('Not selected')
```

```
    def GetResults(self, jobId):
        maxResults = 1000
        paginationToken = None
        finished = False

        while finished == False:

            response = None

            if self.processType == ProcessType.ANALYSIS:
                if paginationToken == None:
                    response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults)
                else:
                    response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults,

NextToken=paginationToken)

            if self.processType == ProcessType.DETECTION:
                if paginationToken == None:
                    response =
self.textract.get_document_text_detection(JobId=jobId,

MaxResults=maxResults)
                else:
                    response =
self.textract.get_document_text_detection(JobId=jobId,

MaxResults=maxResults,

NextToken=paginationToken)

            blocks = response['Blocks']
            print('Detected Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))

            # Display block information
            for block in blocks:
                self.DisplayBlockInfo(block)
                print()
                print()
```

```
            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True

    def GetResultsDocumentAnalysis(self, jobId):
        maxResults = 1000
        paginationToken = None
        finished = False

        while finished == False:

            response = None
            if paginationToken == None:
                response = self.textract.get_document_analysis(JobId=jobId,

 MaxResults=maxResults)
            else:
                response = self.textract.get_document_analysis(JobId=jobId,

 MaxResults=maxResults,

 NextToken=paginationToken)

                # Get the text blocks
            blocks = response['Blocks']
            print('Analyzed Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))
            # Display block information
            for block in blocks:
                self.DisplayBlockInfo(block)
                print()
                print()

                if 'NextToken' in response:
                    paginationToken = response['NextToken']
                else:
                    finished = True


def main():
    roleArn = ''
    bucket = ''
```

```
    document = ''
    region_name = ''

    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.CreateTopicandQueue()
    analyzer.ProcessDocument(ProcessType.DETECTION)
    analyzer.DeleteTopicandQueue()


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

Node.JS

在本示例中，请替换`roleArn`使用您保存的 IAM 角色 ARN授予 Amazon Textract 访问您的 Amazon SNS 主题. 替换的值`bucket`和`document`将使用您在上面步骤 2 中指定的存储桶和文档文件名。替换的值`processType`使用你想在输入文档上使用的处理类型。最后，将值替换为`REGION`您的客户运营所在的地区。

```
 // snippet-start:[sqs.JavaScript.queues.createQueueV3]
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand, GetQueueAttributesCommand, GetQueueUrlCommand,
    SetQueueAttributesCommand, DeleteQueueCommand, ReceiveMessageCommand,
 DeleteMessageCommand } from  "@aws-sdk/client-sqs";
  import {CreateTopicCommand, SubscribeCommand, DeleteTopicCommand } from "@aws-
sdk/client-sns";
  import  { SQSClient } from "@aws-sdk/client-sqs";
  import  { SNSClient } from "@aws-sdk/client-sns";
  import  { TextractClient, StartDocumentTextDetectionCommand,
 StartDocumentAnalysisCommand, GetDocumentAnalysisCommand,
 GetDocumentTextDetectionCommand, DocumentMetadata } from "@aws-sdk/client-
textract";
  import { stdout } from "process";

  // Set the AWS Region.
  const REGION = "us-east-1"; //e.g. "us-east-1"
  // Create SNS service object.
  const sqsClient = new SQSClient({ region: REGION });
  const snsClient = new SNSClient({ region: REGION });
  const textractClient = new TextractClient({ region: REGION });

  // Set bucket and video variables
```

```
 const bucket = "bucket-name";

 const documentName = "document-name";
 const roleArn = "role-arn"
 const processType = "DETECTION"
 var startJobId = ""

 var ts = Date.now();
 const snsTopicName = "AmazonTextractExample" + ts;
 const snsTopicParams = {Name: snsTopicName}
 const sqsQueueName = "AmazonTextractQueue-" + ts;

 // Set the parameters
 const sqsParams = {
   QueueName: sqsQueueName, //SQS_QUEUE_URL
   Attributes: {
     DelaySeconds: "60", // Number of seconds delay.
     MessageRetentionPeriod: "86400", // Number of seconds delay.
   },
 };

 // Process a document based on operation type
 const processDocumment = async (type, bucket, videoName, roleArn, sqsQueueUrl,
snsTopicArn) =>
   {
   try
   {
       // Set job found and success status to false initially
     var jobFound = false
     var succeeded = false
     var dotLine = 0
     var processType = type
     var validType = false

     if (processType == "DETECTION"){
       var response = await textractClient.send(new
StartDocumentTextDetectionCommand({DocumentLocation:{S3Object:{Bucket:bucket,
Name:videoName}},
         NotificationChannel:{RoleArn: roleArn, SNSTopicArn: snsTopicArn}}))
       console.log("Processing type: Detection")
       validType = true
     }

     if (processType == "ANALYSIS"){
```

```
        var response = await textractClient.send(new
StartDocumentAnalysisCommand({DocumentLocation:{S3Object:{Bucket:bucket,
Name:videoName}},
          NotificationChannel:{RoleArn: roleArn, SNSTopicArn: snsTopicArn}}))
        console.log("Processing type: Analysis")
        validType = true
      }

      if (validType == false){
          console.log("Invalid processing type. Choose Detection or Analysis.")
          return
      }
    // while not found, continue to poll for response
    console.log(`Start Job ID: ${response.JobId}`)
    while (jobFound == false){
      var sqsReceivedResponse = await sqsClient.send(new
ReceiveMessageCommand({QueueUrl:sqsQueueUrl,
        MaxNumberOfMessages:'ALL', MaxNumberOfMessages:10}));
      if (sqsReceivedResponse){
        var responseString = JSON.stringify(sqsReceivedResponse)
        if (!responseString.includes('Body')){
          if (dotLine < 40) {
            console.log('.')
            dotLine = dotLine + 1
          }else {
            console.log('')
            dotLine = 0
          };
          stdout.write('', () => {
            console.log('');
          });
          await new Promise(resolve => setTimeout(resolve, 5000));
          continue
        }
      }

        // Once job found, log Job ID and return true if status is succeeded
        for (var message of sqsReceivedResponse.Messages){
            console.log("Retrieved messages:")
            var notification = JSON.parse(message.Body)
            var rekMessage = JSON.parse(notification.Message)
            var messageJobId = rekMessage.JobId
            if (String(rekMessage.JobId).includes(String(startJobId))){
                console.log('Matching job found:')
```

```
                console.log(rekMessage.JobId)
                jobFound = true
                // GET RESUlTS FUNCTION HERE
                var operationResults = await GetResults(processType,
rekMessage.JobId)
                //GET RESULTS FUMCTION HERE
                console.log(rekMessage.Status)
            if (String(rekMessage.Status).includes(String("SUCCEEDED"))){
                succeeded = true
                console.log("Job processing succeeded.")
                var sqsDeleteMessage = await sqsClient.send(new
DeleteMessageCommand({QueueUrl:sqsQueueUrl,
ReceiptHandle:message.ReceiptHandle}));
            }
            }else{
            console.log("Provided Job ID did not match returned ID.")
            var sqsDeleteMessage = await sqsClient.send(new
DeleteMessageCommand({QueueUrl:sqsQueueUrl,
ReceiptHandle:message.ReceiptHandle}));
            }
        }

   console.log("Done!")
   }
   }catch (err) {
       console.log("Error", err);
     }
 }

 // Create the SNS topic and SQS Queue
 const createTopicandQueue = async () => {
   try {
     // Create SNS topic
     const topicResponse = await snsClient.send(new
CreateTopicCommand(snsTopicParams));
     const topicArn = topicResponse.TopicArn
     console.log("Success", topicResponse);
     // Create SQS Queue
     const sqsResponse = await sqsClient.send(new
CreateQueueCommand(sqsParams));
     console.log("Success", sqsResponse);
     const sqsQueueCommand = await sqsClient.send(new
GetQueueUrlCommand({QueueName: sqsQueueName}))
     const sqsQueueUrl = sqsQueueCommand.QueueUrl
```

```
     const attribsResponse = await sqsClient.send(new
GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames:
['QueueArn']}))
     const attribs = attribsResponse.Attributes
     console.log(attribs)
     const queueArn = attribs.QueueArn
     // subscribe SQS queue to SNS topic
     const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
topicArn, Protocol:'sqs', Endpoint: queueArn}))
     const policy = {
       Version: "2012-10-17",
       Statement: [
         {
           Sid: "MyPolicy",
           Effect: "Allow",
           Principal: {AWS: "*"},
           Action: "SQS:SendMessage",
           Resource: queueArn,
           Condition: {
             ArnEquals: {
               'aws:SourceArn': topicArn
             }
           }
         }
       ]
     };

     const response = sqsClient.send(new SetQueueAttributesCommand({QueueUrl:
sqsQueueUrl, Attributes: {Policy: JSON.stringify(policy)}}))
     console.log(response)
     console.log(sqsQueueUrl, topicArn)
     return [sqsQueueUrl, topicArn]

   } catch (err) {
     console.log("Error", err);

   }
 }

 const deleteTopicAndQueue = async (sqsQueueUrlArg, snsTopicArnArg) => {
   const deleteQueue = await sqsClient.send(new DeleteQueueCommand({QueueUrl:
sqsQueueUrlArg}));
   const deleteTopic = await snsClient.send(new DeleteTopicCommand({TopicArn:
snsTopicArnArg}));
```

```
    console.log("Successfully deleted.")
  }

  const displayBlockInfo = async (block) => {
    console.log(`Block ID: ${block.Id}`)
    console.log(`Block Type: ${block.BlockType}`)
    if (String(block).includes(String("EntityTypes"))){
        console.log(`EntityTypes: ${block.EntityTypes}`)
    }
    if (String(block).includes(String("Text"))){
        console.log(`EntityTypes: ${block.Text}`)
    }
    if (!String(block.BlockType).includes('PAGE')){
        console.log(`Confidence: ${block.Confidence}`)
    }
    console.log(`Page: ${block.Page}`)
    if (String(block.BlockType).includes("CELL")){
        console.log("Cell Information")
        console.log(`Column: ${block.ColumnIndex}`)
        console.log(`Row: ${block.RowIndex}`)
        console.log(`Column Span: ${block.ColumnSpan}`)
        console.log(`Row Span: ${block.RowSpan}`)
        if (String(block).includes("Relationships")){
            console.log(`Relationships: ${block.Relationships}`)
        }
    }

    console.log("Geometry")
    console.log(`Bounding Box: ${JSON.stringify(block.Geometry.BoundingBox)}`)
    console.log(`Polygon: ${JSON.stringify(block.Geometry.Polygon)}`)

    if (String(block.BlockType).includes('SELECTION_ELEMENT')){
      console.log('Selection Element detected:')
      if (String(block.SelectionStatus).includes('SELECTED')){
        console.log('Selected')
      } else {
        console.log('Not Selected')
      }

    }
  }

  const GetResults = async (processType, JobID) => {
```

```
    var maxResults = 1000
    var paginationToken = null
    var finished = false

    while (finished == false){
      var response = null
      if (processType == 'ANALYSIS'){
        if (paginationToken == null){
          response = textractClient.send(new
GetDocumentAnalysisCommand({JobId:JobID, MaxResults:maxResults}))

        }else{
          response = textractClient.send(new
GetDocumentAnalysisCommand({JobId:JobID, MaxResults:maxResults,
NextToken:paginationToken}))
        }
      }

      if(processType == 'DETECTION'){
        if (paginationToken == null){
          response = textractClient.send(new
GetDocumentTextDetectionCommand({JobId:JobID, MaxResults:maxResults}))

        }else{
          response = textractClient.send(new
GetDocumentTextDetectionCommand({JobId:JobID, MaxResults:maxResults,
NextToken:paginationToken}))
        }
      }

      await new Promise(resolve => setTimeout(resolve, 5000));
      console.log("Detected Documented Text")
      console.log(response)
      //console.log(Object.keys(response))
      console.log(typeof(response))
      var blocks = (await response).Blocks
      console.log(blocks)
      console.log(typeof(blocks))
      var docMetadata = (await response).DocumentMetadata
      var blockString = JSON.stringify(blocks)
      var parsed = JSON.parse(JSON.stringify(blocks))
      console.log(Object.keys(blocks))
      console.log(`Pages: ${docMetadata.Pages}`)
      blocks.forEach((block)=> {
```

```
        displayBlockInfo(block)
        console.log()
        console.log()
      })

      //console.log(blocks[0].BlockType)
      //console.log(blocks[1].BlockType)


      if(String(response).includes("NextToken")){
        paginationToken = response.NextToken
      }else{
        finished = true
      }
    }

  }


  // DELETE TOPIC AND QUEUE
  const main = async () => {
    var sqsAndTopic = await createTopicandQueue();
    var process = await processDocumment(processType, bucket, documentName,
 roleArn, sqsAndTopic[0], sqsAndTopic[1])
    var deleteResults = await deleteTopicAndQueue(sqsAndTopic[0],
 sqsAndTopic[1])
  }

main()
```

4. 运行该代码。此操作可能需要一段时间才能完成。完成后，将显示检测到或分析的文本的块列表。

Amazon Textract 结果通知

Amazon Textract 会将 Amazon Textract 分析请求的结果（包括完成状态）发布到 Amazon Simple Notification Service (Amazon SNS) 主题。要从 Amazon SNS 主题获取通知，请使用 Amazon SQS 队列或AWS Lambdafunction. 有关更多信息，请参阅 调用 Amazon Textract 异步操作。有关示例，请参阅 检测或分析多页文档中的文本。

结果的格式为以下 JSON 格式：

```
{
```

```
  "JobId": "String",
  "Status": "String",
  "API": "String",
  "JobTag": "String",
  "Timestamp": Number,
  "DocumentLocation": {
    "S3ObjectName": "String",
    "S3Bucket": "String"
  }
}
```

下表介绍 Amazon Textract 响应中的各种参数。

参数	描述
JobId	Amazon Textract 分配给作业的唯一标识符。此标识符匹配从`Start`操作，例如StartDocumentTextDetection.
状态	任务的状态。有效值为 “成功”、“失败” 或 “错误”。
API	用于分析输入文档的 Amazon Textract 操作，如StartDocumentTextDetection要么StartDocumentAnalysis.
JobTag	作业的用户指定的标识符。你指定`JobTag`在打电话给`Start`操作，例如StartDocumentTextDetection.
时间戳	指示作业完成时间的 Unix 时间戳，以毫秒为单位返回。
文档位置	有关已处理的文档的详细信息。包含文件名和将文件存储到的 Amazon S3 存储桶。

处理受限的呼叫和断开的连接

如果您超过每秒最大交易次数 (TPS)、导致服务限制您的应用程序，或者连接断开，Amazon Textract 操作可能会失败。例如，如果您在短时间内对 Amazon Textract 操作进行了太多调用，则会限制您的呼叫并发送`ProvisionedThroughputExceededException`操作响应中出现错误。有关 Amazon Textract TPS 配额的信息，请参阅Amazon Textract 配额.

您可以通过自动重试操作来管理限制和断开的连接。您可以通过包括`Config`当您创建 Amazon Textract 客户端时，将参数。我们建议重试计数为 5。这些区域有：AWS在失败并引发异常之前，开发工具包会重试操作指定次数。有关更多信息，请参阅 。AWS 中的错误重试和指数退避.

> **Note**
>
> 自动重试适用于同步操作和异步操作。在指定自动重试之前，请确保您拥有最新版本的 AWS 开发工具包。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。

以下示例说明了当您处理多个文档时，如何自动重试 Amazon Textract 操作。

先决条件

- 如果您尚未执行以下操作，请：
  a. 使用创建或更新 IAM 用户`AmazonTextractFullAccess`和`AmazonS3ReadOnlyAccess`权限。有关更多信息，请参阅 第 1 步：设置 AWS 账户并创建 IAM 用户。
  b. 安装和配置 AWS CLI 和 AWS 开发工具包。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。

自动重试操作

1. 将多个文档图像上传到 S3 存储桶以运行同步示例。将多页文档上传到 S3 存储桶并运行`StartDocumentTextDetection`在它上面运行异步示例。

   有关说明，请参阅将对象上传到 Amazon S3中的Amazon Simple Storage Service 用户指南.
2. 以下示例演示了如何使用`Config`参数以自动重试操作。同步示例调用`DetectDocumentText`操作，而异步示例调用`GetDocumentTextDetection`operation.

Sync Example

使用以下示例调用DetectDocumentText对 Amazon S3 存储桶中的文档进行操作。Inmain，更改的值bucket到您的 S3 存储桶。更改的值documents转到您在步骤 2 中上传的文档图像的名称。

```
import boto3
from botocore.client import Config
# Documents

def process_multiple_documents(bucket, documents):

    config = Config(retries = dict(max_attempts = 5))

    # Amazon Textract client
    textract = boto3.client('textract', config=config)

    for documentName in documents:

        print("\nProcessing:
 {}\n=========================================".format(documentName))

        # Call Amazon Textract
        response = textract.detect_document_text(
            Document={
                'S3Object': {
                    'Bucket': bucket,
                    'Name': documentName
                }
            })

        # Print detected text
        for item in response["Blocks"]:
            if item["BlockType"] == "LINE":
                print ('\033[94m' +  item["Text"] + '\033[0m')


def main():
    bucket = ""
    documents = ["document-image-1.png",
    "document-image-2.png", "document-image-3.png",
    "document-image-4.png", "document-image-5.png" ]
```

```
    process_multiple_documents(bucket, documents)



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

Async Example

使用以下示例调用 GetDocumentTextDetection 操作。假定您已经打过电话StartDocumentTextDetection在您的 Amazon S3 存储桶中的文档上，并获得了JobId. Inmain，更改的值bucket对于您的 S3 存储桶和的值roleArn转到分配给你的 Textract 角色的 Arn。您还需要更改的值document将替换为您的 Amazon S3 存储桶中的多页文档的名称。最后，将值替换为region_name提供您所在区域的名称并提供GetResults以您的名称为的函数jobId.

```
import boto3
from botocore.client import Config

class DocumentProcessor:
    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

    def __init__(self, role, bucket, document, region):
        self.roleArn = role
        self.bucket = bucket
        self.document = document
        self.region_name = region
        self.config = Config(retries = dict(max_attempts = 5))

        self.textract = boto3.client('textract', region_name=self.region_name,
 config=self.config)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')
```

```
# Display information about a block
    def DisplayBlockInfo(self, block):

        print("Block Id: " + block['Id'])
        print("Type: " + block['BlockType'])
        if 'EntityTypes' in block:
            print('EntityTypes: {}'.format(block['EntityTypes']))

        if 'Text' in block:
            print("Text: " + block['Text'])

        if block['BlockType'] != 'PAGE':
            print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

        print('Page: {}'.format(block['Page']))

        if block['BlockType'] == 'CELL':
            print('Cell Information')
            print('\tColumn: {} '.format(block['ColumnIndex']))
            print('\tRow: {}'.format(block['RowIndex']))
            print('\tColumn span: {} '.format(block['ColumnSpan']))
            print('\tRow span: {}'.format(block['RowSpan']))

            if 'Relationships' in block:
                print('\tRelationships: {}'.format(block['Relationships']))

        print('Geometry')
        print('\tBounding Box: {}'.format(block['Geometry']['BoundingBox']))
        print('\tPolygon: {}'.format(block['Geometry']['Polygon']))

        if block['BlockType'] == 'SELECTION_ELEMENT':
            print('    Selection element detected: ', end='')
            if block['SelectionStatus'] == 'SELECTED':
                print('Selected')
            else:
                print('Not selected')

    def GetResults(self, jobId):
        maxResults = 1000
        paginationToken = None
        finished = False

        while finished == False:
```

```
            response = None

            if paginationToken == None:
                response =
 self.textract.get_document_text_detection(JobId=jobId,

 MaxResults=maxResults)
            else:
                response =
 self.textract.get_document_text_detection(JobId=jobId,

 MaxResults=maxResults,

 NextToken=paginationToken)

            blocks = response['Blocks']
            print('Detected Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))

            # Display block information
            for block in blocks:
                self.DisplayBlockInfo(block)
                print()
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True

def main():
    roleArn = 'role-arn'
    bucket = 'bucket-name'
    document = 'document-name'
    region_name = 'region-name'
    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.GetResults("job-id")

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

Amazon Textract 的最佳实践

Amazon Textract 使用机器学习来像个人一样阅读文档。它从文档中提取文本、表格和表单。使用以下最佳实践从文档中获取最佳结果。

提供最佳输入文档

以下列出了可以优化输入文档以获得更好结果的几种方法。

- 确保您的文档文本使用 Amazon Textract 支持的语言。目前，Amazon Textract 支持英语、西班牙语、德语、意大利语、法语和葡萄牙语。
- 提供高质量的图像，理想情况下至少为 150 DPI。
- 如果您的文档已采用 Amazon Textract 支持的其中一种文件格式（PDF、TIFF、JPEG 和 PNG），请勿在将文档上传到 Amazon Textract 之前对文档进行转换或降样。

为了在从文档中的表格中提取文本时获得最佳效果，请确保：

- 文档中的表格在视觉上与页面上的周围元素分开。例如，表格不会叠加到图像或复杂模式上。
- 表格中的文字是直立的。例如，文本不会相对于页面上的其他文本进行旋转。

从表中提取文本时，在以下情况下可能会看到不一致的结果：

- 合并的跨越多列的表格单元格。
- 单元格、行或列与同一表的其他部分不同的表格。

我们建议使用文本检测作为解决方法。

使用置信度得分

您应该考虑 Amazon Textract API 操作返回的信心分数及其使用案例的敏感性。置信度得分是一个介于 0 与 100 之间的数字，用于表示给定预测的准确性。它可以帮助你就如何使用结果做出明智的决定。

在对检测错误（误报）敏感的应用程序中，强制实施最低置信度评分阈值。申请应放弃低于该阈值的结果，或者将情况标记为需要更高级别的人工审查。

最佳阈值取决于应用程序。出于存档目的，例如记录手写笔记，可能低至 50%。涉及财务决策的业务流程可能需要 90% 或更高的阈值。

考虑使用人工审核

还可以考虑将人工评论纳入工作流程中。这对于敏感的应用程序尤其重要，例如涉及财务决策的业务流程。

教程

the section called “Block”从 Amazon Textract 操作返回的对象包含文本检测和文本分析操作的结果，例如the section called “AnalyzeDocument”. 以下 Python 教程演示了使用 Block 对象的一些不同方法。例如，您可以将表格信息导出为以逗号分隔的值 (CSV) 文件。

本教程使用同步 Amazon Textract 操作来返回所有结果。如果你想使用异步操作，例如the section called “StartDocumentAnalysis”，您需要更改示例代码才能容纳多批退货`Block`对象。要利用异步操作示例，请确保您已按照为异步操作配置 Amazon Textract.

有关向您展示其他使用 Amazon Textract 的方法的示例，请参阅其他代码示例.

主题

- 先决条件
- 从表单文档中提取键值对
- 将表导出到 CSV 文件
- 创建AWS Lambda函数
- 其他代码示例

先决条件

您必须先配置您的环境，才能运行此部分中的示例。

配置您的环境

1. 使用创建或更新 IAM 用户`AmazonTextractFullAccess`权限。有关更多信息，请参阅 第 1 步：设置 AWS 账户并创建 IAM 用户。
2. 安装和配置 AWS CLI 和 AWS 开发工具包。有关更多信息，请参阅 第 2 步：设置AWS CLI和AWS软件开发工具包。

从表单文档中提取键值对

以下 Python 示例演示了如何从表单文档中提取键值对。the section called “Block”存储在地图中的对象。阻止对象通过调用返回the section called “AnalyzeDocument”. 有关更多信息，请参阅 表单数据（键值对）。

您可以使用以下功能：

- `get_kv_map`— 调用AnalyzeDocument，并将 KEY 和 VALUE BLOCK 对象存储在地图中。
- `get_kv_relationship`和`find_value_block`— 从地图构造键值关系。

从表单文档中提取键-值对

1. 配置您的环境。有关更多信息，请参阅 先决条件。
2. 将以下示例代码保存到名为的文件中textract_python_kv_parser.py.

```
import boto3
import sys
import re
import json


def get_kv_map(file_name):

    with open(file_name, 'rb') as file:
        img_test = file.read()
        bytes_test = bytearray(img_test)
        print('Image loaded', file_name)

    # process using image bytes
    client = boto3.client('textract')
    response = client.analyze_document(Document={'Bytes': bytes_test},
 FeatureTypes=['FORMS'])

    # Get the text blocks
    blocks=response['Blocks']


    # get key and value maps
    key_map = {}
    value_map = {}
    block_map = {}
    for block in blocks:
        block_id = block['Id']
        block_map[block_id] = block
        if block['BlockType'] == "KEY_VALUE_SET":
            if 'KEY' in block['EntityTypes']:
                key_map[block_id] = block
```

```
            else:
                value_map[block_id] = block

    return key_map, value_map, block_map


def get_kv_relationship(key_map, value_map, block_map):
    kvs = {}
    for block_id, key_block in key_map.items():
        value_block = find_value_block(key_block, value_map)
        key = get_text(key_block, block_map)
        val = get_text(value_block, block_map)
        kvs[key] = val
    return kvs


def find_value_block(key_block, value_map):
    for relationship in key_block['Relationships']:
        if relationship['Type'] == 'VALUE':
            for value_id in relationship['Ids']:
                value_block = value_map[value_id]
    return value_block


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    word = blocks_map[child_id]
                    if word['BlockType'] == 'WORD':
                        text += word['Text'] + ' '
                    if word['BlockType'] == 'SELECTION_ELEMENT':
                        if word['SelectionStatus'] == 'SELECTED':
                            text += 'X '


    return text


def print_kvs(kvs):
    for key, value in kvs.items():
        print(key, ":", value)
```

```
def search_value(kvs, search_key):
    for key, value in kvs.items():
        if re.search(search_key, key, re.IGNORECASE):
            return value

def main(file_name):

    key_map, value_map, block_map = get_kv_map(file_name)

    # Get Key Value relationship
    kvs = get_kv_relationship(key_map, value_map, block_map)
    print("\n\n== FOUND KEY : VALUE pairs ===\n")
    print_kvs(kvs)

    # Start searching a key value
    while input('\n Do you want to search a value for a key? (enter "n" for exit)
 ') != 'n':
        search_key = input('\n Enter a search key:')
        print('The value is:', search_value(kvs, search_key))

if __name__ == "__main__":
    file_name = sys.argv[1]
    main(file_name)
```

3. 在命令提示符处，输入以下命令：Replacefile使用要分析的文档图像文件执行。

```
textract_python_kv_parser.py file
```

4. 出现提示时，请输入输入文档中的密钥。如果代码检测到密钥，则会显示密钥的值。

将表导出到 CSV 文件

这些 Python 示例演示了如何将表从文档图像导出为以逗号分隔的值 (CSV) 文件。

同步文档分析的示例从调用中收集表格信息the section called “AnalyzeDocument”. 异步文档分析的示例调用the section called “StartDocumentAnalysis”然后从中检索结果the section called “GetDocumentAnalysis”如同Block对象。

表格信息返回为the section called “Block”来自调用的对象the section called “AnalyzeDocument”. 有关更多信息，请参阅 表。这些区域有：Block对象存储在用于将表数据导出到 CSV 文件的地图结构中。

Synchronous

在此示例中，您将使用以下函数：

- get_table_csv_results— 调用AnalyzeDocument，然后构建文档中检测到的表格的映射。创建所有检测到的表格的 CSV 表示形式。
- generate_table_csv— 为单个表生成 CSV 文件。
- get_rows_columns_map— 从地图中获取行和列。
- get_text— 从单元格中获取文本。

将表导出到 CSV 文件

1. 配置您的环境。有关更多信息，请参阅 先决条件。
2. 将以下示例代码保存到名为的文件中textract_python_table_parser.py.

```
import webbrowser, os
import json
import boto3
import io
from io import BytesIO
import sys
from pprint import pprint


def get_rows_columns_map(table_result, blocks_map):
    rows = {}
    for relationship in table_result['Relationships']:
        if relationship['Type'] == 'CHILD':
            for child_id in relationship['Ids']:
                cell = blocks_map[child_id]
                if cell['BlockType'] == 'CELL':
                    row_index = cell['RowIndex']
                    col_index = cell['ColumnIndex']
                    if row_index not in rows:
                        # create new row
                        rows[row_index] = {}

                    # get the text value
                    rows[row_index][col_index] = get_text(cell, blocks_map)
    return rows
```

```
def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    word = blocks_map[child_id]
                    if word['BlockType'] == 'WORD':
                        text += word['Text'] + ' '
                    if word['BlockType'] == 'SELECTION_ELEMENT':
                        if word['SelectionStatus'] =='SELECTED':
                            text +=  'X '
    return text


def get_table_csv_results(file_name):

    with open(file_name, 'rb') as file:
        img_test = file.read()
        bytes_test = bytearray(img_test)
        print('Image loaded', file_name)

    # process using image bytes
    # get the results
    client = boto3.client('textract')

    response = client.analyze_document(Document={'Bytes': bytes_test},
 FeatureTypes=['TABLES'])

    # Get the text blocks
    blocks=response['Blocks']
    pprint(blocks)

    blocks_map = {}
    table_blocks = []
    for block in blocks:
        blocks_map[block['Id']] = block
        if block['BlockType'] == "TABLE":
            table_blocks.append(block)

    if len(table_blocks) <= 0:
        return "<b> NO Table FOUND </b>"
```

```
    csv = ''
    for index, table in enumerate(table_blocks):
        csv += generate_table_csv(table, blocks_map, index +1)
        csv += '\n\n'

    return csv

def generate_table_csv(table_result, blocks_map, table_index):
    rows = get_rows_columns_map(table_result, blocks_map)

    table_id = 'Table_' + str(table_index)

    # get cells.
    csv = 'Table: {0}\n\n'.format(table_id)

    for row_index, cols in rows.items():

        for col_index, text in cols.items():
            csv += '{}'.format(text) + ","
        csv += '\n'

    csv += '\n\n\n'
    return csv

def main(file_name):
    table_csv = get_table_csv_results(file_name)

    output_file = 'output.csv'

    # replace content
    with open(output_file, "wt") as fout:
        fout.write(table_csv)

    # show the results
    print('CSV OUTPUT FILE: ', output_file)


if __name__ == "__main__":
    file_name = sys.argv[1]
    main(file_name)
```

3. 在命令提示符处，输入以下命令：Replacefile将显示您要分析的文档图像文件的名称。

```
python textract_python_table_parser.py file
```

运行示例时，CSV 输出将保存在名为的文件中output.csv.

Asynchronous

在此示例中，您将使用两个不同的脚本。第一个脚本启动了异步分析文档的过程StartDocumentAnalysis并获取Block返回的信息GetDocumentAnalysis. 第二个脚本采用返回的Block每个页面的信息，将数据格式化为表格，然后将表格保存到 CSV 文件中。

将表导出到 CSV 文件

1. 配置您的环境。有关更多信息，请参阅 先决条件。
2. 确保你已按照见上的说明进行操作为异步操作配置 Amazon Textract. 该页面上记录的过程使您能够发送和接收有关异步作业完成状态的消息。
3. 在以下代码示例中，替换roleArn将 Arn 分配给您在步骤 2 中创建的角色。替换的值bucket将包含您的文档的 S3 存储桶的名称。替换的值document将包含您的 S3 存储桶中的文档的名称。替换的值region_name将包含您的存储桶所在区域的名称。

   将以下示例代码保存到名为的文件中start_doc_analysis_for_table_extraction.py。.

```
import boto3
import time

class DocumentProcessor:

    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

    def __init__(self, role, bucket, document, region):
        self.roleArn = role
        self.bucket = bucket
```

```
        self.document = document
        self.region_name = region

        self.textract = boto3.client('textract', region_name=self.region_name)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')

    def ProcessDocument(self):

        jobFound = False

        response =
 self.textract.start_document_analysis(DocumentLocation={'S3Object': {'Bucket':
 self.bucket, 'Name': self.document}},
                FeatureTypes=["TABLES", "FORMS"],
 NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})
        print('Processing type: Analysis')

        print('Start Job Id: ' + response['JobId'])

        print('Done!')

    def CreateTopicandQueue(self):

        millis = str(int(round(time.time() * 1000)))

        # Create SNS topic
        snsTopicName = "AmazonTextractTopic" + millis

        topicResponse = self.sns.create_topic(Name=snsTopicName)
        self.snsTopicArn = topicResponse['TopicArn']

        # create SQS queue
        sqsQueueName = "AmazonTextractQueue" + millis
        self.sqs.create_queue(QueueName=sqsQueueName)
        self.sqsQueueUrl = self.sqs.get_queue_url(QueueName=sqsQueueName)
['QueueUrl']

        attribs = self.sqs.get_queue_attributes(QueueUrl=self.sqsQueueUrl,
                                                AttributeNames=['QueueArn'])
['Attributes']

        sqsQueueArn = attribs['QueueArn']
```

```
        # Subscribe SQS queue to SNS topic
        self.sns.subscribe(TopicArn=self.snsTopicArn, Protocol='sqs',
 Endpoint=sqsQueueArn)

        # Authorize SNS to write SQS queue
        policy = """{{
      "Version":"2012-10-17",
      "Statement":[
        {{
          "Sid":"MyPolicy",
          "Effect":"Allow",
          "Principal" : {{"AWS" : "*"}},
          "Action":"SQS:SendMessage",
          "Resource": "{}",
          "Condition":{{
            "ArnEquals":{{
              "aws:SourceArn": "{}"
            }}
          }}
        }}
      ]
    }}""".format(sqsQueueArn, self.snsTopicArn)

        response = self.sqs.set_queue_attributes(
            QueueUrl=self.sqsQueueUrl,
            Attributes={
                'Policy': policy
            })

def main():
    roleArn = 'role-arn'
    bucket = 'bucket-name'
    document = 'document-name'
    region_name = 'region-name'

    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.CreateTopicandQueue()
    analyzer.ProcessDocument()

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

4. 运行该代码。该代码将打印一个 JobId。向下复制这个 JobId。

5. 等待您的作业完成处理，完成后，将以下代码复制到名为的文件get_doc_analysis_for_table_extraction.py. 替换的值`jobId`用你之前复制的 Job ID。替换的值`region_name`使用与 Textract 角色关联的区域的名称。替换的值`file_name`使用您要为输出 CSV 指定的名称。

```
import boto3
from pprint import pprint

jobId = 'job-id'
region_name = 'region-name'
file_name = "output-file-name.csv"

textract = boto3.client('textract', region_name=region_name)

# Display information about a block
def DisplayBlockInfo(block):
    print("Block Id: " + block['Id'])
    print("Type: " + block['BlockType'])
    if 'EntityTypes' in block:
        print('EntityTypes: {}'.format(block['EntityTypes']))

    if 'Text' in block:
        print("Text: " + block['Text'])

    if block['BlockType'] != 'PAGE':
        print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

def GetResults(jobId, file_name):
    maxResults = 1000
    paginationToken = None
    finished = False

    while finished == False:

        response = None

        if paginationToken == None:
            response = textract.get_document_analysis(JobId=jobId,
 MaxResults=maxResults)
        else:
            response = textract.get_document_analysis(JobId=jobId,
 MaxResults=maxResults,
```

```
 NextToken=paginationToken)

        blocks = response['Blocks']
        table_csv = get_table_csv_results(blocks)
        output_file = file_name
        # replace content
        with open(output_file, "at") as fout:
            fout.write(table_csv)
        # show the results
        print('Detected Document Text')
        print('Pages: {}'.format(response['DocumentMetadata']['Pages']))
        print('OUTPUT TO CSV FILE: ', output_file)

        # Display block information
        for block in blocks:
            DisplayBlockInfo(block)
            print()
            print()

        if 'NextToken' in response:
            paginationToken = response['NextToken']
        else:
            finished = True


def get_rows_columns_map(table_result, blocks_map):
    rows = {}
    for relationship in table_result['Relationships']:
        if relationship['Type'] == 'CHILD':
            for child_id in relationship['Ids']:
                try:
                    cell = blocks_map[child_id]
                    if cell['BlockType'] == 'CELL':
                        row_index = cell['RowIndex']
                        col_index = cell['ColumnIndex']
                        if row_index not in rows:
                            # create new row
                            rows[row_index] = {}

                        # get the text value
                        rows[row_index][col_index] = get_text(cell, blocks_map)
                except KeyError:
                    print("Error extracting Table data - {}:".format(KeyError))
```

```
                    pass
    return rows


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    try:
                        word = blocks_map[child_id]
                        if word['BlockType'] == 'WORD':
                            text += word['Text'] + ' '
                        if word['BlockType'] == 'SELECTION_ELEMENT':
                            if word['SelectionStatus'] == 'SELECTED':
                                text += 'X '
                    except KeyError:
                        print("Error extracting Table data -
 {}:".format(KeyError))

    return text


def get_table_csv_results(blocks):

    pprint(blocks)

    blocks_map = {}
    table_blocks = []
    for block in blocks:
        blocks_map[block['Id']] = block
        if block['BlockType'] == "TABLE":
            table_blocks.append(block)

    if len(table_blocks) <= 0:
        return "<b> NO Table FOUND </b>"

    csv = ''
    for index, table in enumerate(table_blocks):
        csv += generate_table_csv(table, blocks_map, index + 1)
        csv += '\n\n'

    return csv
```

```
def generate_table_csv(table_result, blocks_map, table_index):
    rows = get_rows_columns_map(table_result, blocks_map)

    table_id = 'Table_' + str(table_index)

    # get cells.
    csv = 'Table: {0}\n\n'.format(table_id)

    for row_index, cols in rows.items():

        for col_index, text in cols.items():
            csv += '{}'.format(text) + ","
        csv += '\n'

    csv += '\n\n\n'
    return csv

response_blocks = GetResults(jobId, file_name)
```

6. 运行该代码。

   获得结果后，请务必删除关联的 SNS 和 SQS 资源，否则可能会为它们产生费用。

创建AWS Lambda函数

您可以从内部调用 Amazon Textract API 操作AWS Lambdafunction. 以下说明演示了如何在 Python 中创建调用的 Lambda 函数。the section called “DetectDocumentText”. 它返回一组the section called “Block”对象。要运行此示例，您需要一个包含 PNG 或 JPEG 格式文档的 Amazon S3 存储桶。要创建函数，您可以使用控制台。

有关使用 Lambda 函数大规模处理文档的示例，请参阅使用 Amazon Textract 进行大规模文档处理.

要从 Lambda 函数调用 DetectDocumentText 操作，请执行以下操作：

第 1 步：创建 Lambda 部署程序包

1. 打开一个命令窗口。
2. 输入以下命令以创建具有最新版本的部署程序包AWSSDK。

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

第 2 步：创建 Lambda 函数

1. 登录到 AWS 管理控制台，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 选择 Create function（创建函数）。
3. 指定以下内容。
   - 选择 Author from scratch (从头创作)。
   - 对于 Function name (函数名称)，输入一个名称。
   - 适用于运行时，选择Python 3.7要么Python 3.6.
   - 适用于选择或创建执行角色，选择创建具有基本 Lambda 权限的新角色.
4. 选择创建函数创建 Lambda 函数。
5. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
6. 在导航窗格中，选择角色.
7. 从资源列表中，选择 Lambda 为您创建的 IAM 角色。角色名称以 Lambda 函数的名称开头。
8. 选择Permissions (权限)选项卡，然后选择附加策略.
9. 选择 AmazonTextractFullAccess 权限和 AmazonS3ReadOnlyAccess 政策。
10. Select附加策略.

有关更多信息，请参阅 。使用控制台创建 Lambda 函数

第 3 步：创建并添加层

1. 打开 AWS Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 在导航窗格中，选择 Layers（层）。
3. 选择 Create layer (创建层)。
4. 适用于名称，输入一个名称。
5. 对于说明，输入说明。

6. 适用于代码条目类型，选择上传 .zip 文件然后选择上传.
7. 在对话框中，选择 zip 文件 (boto3-layer.zip)，即您在其中创建的 zip 文件第 1 步：创建 Lambda 部署程序包.
8. 适用于兼容运行时，选择您在中选择的运行时版本第 2 步：创建 Lambda 函数.
9. 选择Create创建层。
10. 选择导航窗格菜单图标。
11. 在导航窗格中，选择 Functions（函数）。
12. 在资源列表中，选择您在中创建的函数。第 2 步：创建 Lambda 函数.
13. 选择配置而且在设计师部分，选择层（在您的 Lambda 函数名称下）。
14. 在层部分，选择添加层.
15. 选择从运行时兼容的图层列表中选择.
16. In兼容层中，选择名称和版本在步骤 3 中创建的层。
17. 选择 Add (添加)。

第 4 步：将 python 代码添加到函数

1. In设计师，选择函数。
2. 在函数代码编辑器中，将以下内容添加到文件。lambda_function.py. 更改的值bucket和document到存储桶和文档中。

```
import json
import boto3

def lambda_handler(event, context):

    bucket="bucket"
    document="document"
    client = boto3.client('textract')



    #process using S3 object
    response = client.detect_document_text(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    #Get the text blocks
    blocks=response['Blocks']
```

```
    return {
        'statusCode': 200,
        'body': json.dumps(blocks)
    }
```

3. 选择Save（保存）保存 Lambda 函数。

第 5 步：测试 Lambda

1. Select测试.
2. 为Event name (事件名称).
3. 选择创建。
4. 输出，列表the section called “Block”对象，将显示在执行结果窗格中。

如果AWS Lambda函数返回超时错误，可能是 Amazon Textract API 操作调用的原因。有关延长超时期限的信息AWS Lambda请参阅函数AWS Lambda 函数配置.

有关从您的代码调用 Lambda 函数的信息，请参阅调用AWS Lambda函数.

其他代码示例

下表提供指向更多 Amazon Textract 代码示例的链接。

示例	描述
Amazon Textract 代码示例	展示您可以使用 Amazon Textract 的各种方法。
使用 Amazon Textract 进行大规模文档处理	显示大规模处理文档的无服务器参考体系结构。
Amazon Textract 解析器	演示如何解析the section called “Block”Amazon Textract 操作返回的对象。
Amazon Textract 文档代码示例	本指南中使用的代码示例。
提取器	演示如何将 Amazon Textract 输出转换为多种格式。

示例	描述
使用 Amazon Textract 生成可搜索的 PDF 文档	演示如何根据不同类型的输入文档（如 JPG/PNG 格式图像和扫描的 PDF 文档）创建可搜索的 PDF 文档。

适用于 Amazon Textract 的代码示例

以下代码示例显示如何将 Amazon Textract 与AWS软件开发套件 (SDK)。

这些示例可分为以下几类：

操作

展示如何调用具体服务函数的代码节选。

跨服务示例

跨多个工作的示例应用程序AWS服务。

有关的完整列表AWSSDK 开发人员指南和代码示例，请参阅将 Amazon Textract 与AWS开发工具包. 本主题还包括有关入门的信息以及有关以前 SDK 版本的详细信息。

代码示例

- 适用于 Amazon Textract 的操作
  - 使用 Amazon Textract 和AWS开发工具包
  - 使用 Amazon Textract 和AWS开发工具包
  - 使用AWS开发工具包
  - 使用 Amazon Textract 和AWS开发工具包
  - 使用 Amazon Textract 和AWS开发工具包
- 适用于 Amazon Textract 的跨服务示例
  - 创建 Amazon Textract 浏览器应用程序
  - 使用从图像中提取的文本中检测实体AWS开发工具包

适用于 Amazon Textract 的操作

以下代码示例演示如何使用执行单个 Amazon Textract 操作：AWS开发工具包。这些摘录称为 Amazon Textract API，不打算单独运行。每个示例都包含一个指向 GitHub 的链接，其中包含了有关如何在上下文中设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅Amazon Textract API 参考.

示例

- 使用 Amazon Textract 和AWS开发工具包
- 使用 Amazon Textract 和AWS开发工具包
- 使用AWS开发工具包
- 使用 Amazon Textract 和AWS开发工具包
- 使用 Amazon Textract 和AWS开发工具包

使用 Amazon Textract 和AWS开发工具包

以下代码示例显示如何使用 Amazon Textract 分析文档。

Java

SDK for Java 2.x

```
   public static void analyzeDoc(TextractClient textractClient, String
sourceDoc) {

       try {
           InputStream sourceStream = new FileInputStream(new File(sourceDoc));
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           // Get the input Document object as bytes
           Document myDoc = Document.builder()
                   .bytes(sourceBytes)
                   .build();

           List<FeatureType> featureTypes = new ArrayList<FeatureType>();
           featureTypes.add(FeatureType.FORMS);
           featureTypes.add(FeatureType.TABLES);

           AnalyzeDocumentRequest analyzeDocumentRequest =
AnalyzeDocumentRequest.builder()
                   .featureTypes(featureTypes)
                   .document(myDoc)
                   .build();

           AnalyzeDocumentResponse analyzeDocument =
textractClient.analyzeDocument(analyzeDocumentRequest);
           List<Block> docInfo = analyzeDocument.blocks();
```

```
            Iterator<Block> blockIterator = docInfo.iterator();

            while(blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is "
 +block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 在 GitHub 中查找说明和更多代码。
- 有关 API 详细信息，请参阅AnalyzeDocument在AWS SDK for Java 2.xAPI 参考.

Python

SDK for Python (Boto3)

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def analyze_file(
            self, feature_types, *, document_file_name=None,
 document_bytes=None):
        """
        Detects text and additional elements, such as forms or tables, in a local
 image
        file or from in-memory byte data.
```

```
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to
detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={'Bytes': document_bytes}, FeatureTypes=feature_types)
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- 在 GitHub 中查找说明和更多代码。
- 有关 API 详细信息，请参阅AnalyzeDocument在AWSSDK for Python (Boto3) 的 API 参考.

有关的完整列表AWSSDK 开发人员指南和代码示例，请参阅将 Amazon Textract 与AWS开发工具包. 本主题还包括有关入门的信息以及有关以前 SDK 版本的详细信息。

使用 Amazon Textract 和AWS开发工具包

以下代码示例显示如何使用 Amazon Textract 检测文档中的文本。

Java

SDK for Java 2.x

检测输入文档中的文本。

```
   public static void detectDocText(TextractClient textractClient,String
sourceDoc) {
```

```
        try {

            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the Detect operation
            DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);

            List<Block> docInfo = textResponse.blocks();

            Iterator<Block> blockIterator = docInfo.iterator();

            while(blockIterator.hasNext()) {
                Block block = blockIterator.next();
                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 | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

检测 Amazon S3 存储桶中文档中的文本。

```
   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();

           // Create a DetectDocumentTextRequest object
           DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                   .document(myDoc)
                   .build();

           // Invoke the detectDocumentText method
           DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);

           List<Block> docInfo = textResponse.blocks();

           Iterator<Block> blockIterator = docInfo.iterator();

           while(blockIterator.hasNext()) {
               Block block = blockIterator.next();
               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);
       }
```

```
    }
```

- 在 GitHub 中查找说明和更多代码。
- 有关 API 详细信息，请参阅DetectDocumentText在AWS SDK for Java 2.xAPI 参考.

Python

SDK for Python (Boto3)

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.detect_document_text(
                Document={'Bytes': document_bytes})
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
```

```
            raise
        else:
            return response
```

- 在 GitHub 中查找说明和更多代码。
- 有关 API 详细信息，请参阅DetectDocumentText在AWSSDK for Python (Boto3) 的 API 参考.

有关的完整列表AWSSDK 开发人员指南和代码示例，请参阅将 Amazon Textract 与AWS开发工具包. 本主题还包括有关入门的信息以及有关以前 SDK 版本的详细信息。

使用AWS开发工具包

以下代码示例显示如何获取 Amazon Textract 文档分析作业的数据。

Python

SDK for Python (Boto3)

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                 detected in the image.
        """
        try:
```

```
            response = self.textract_client.get_document_analysis(
                JobId=job_id)
            job_status = response['JobStatus']
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
            return response
```

- 在 GitHub 中查找说明和更多代码。
- 有关 API 详细信息，请参阅GetDocumentAnalysis在AWSSDK for Python (Boto3) 的 API 参考.

有关的完整列表AWSSDK 开发人员指南和代码示例，请参阅将 Amazon Textract 与AWS开发工具包. 本主题还包括有关入门的信息以及有关以前 SDK 版本的详细信息。

使用 Amazon Textract 和AWS开发工具包

以下代码示例显示如何使用 Amazon Textract 开始异步分析文档。

Java

SDK for Java 2.x

```
   public static String startDocAnalysisS3 (TextractClient textractClient,
String bucketName, String docName) {

       try {

           List<FeatureType> myList = new ArrayList<FeatureType>();
           myList.add(FeatureType.TABLES);
           myList.add(FeatureType.FORMS);

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

           DocumentLocation location = DocumentLocation.builder()
```

```
                    .s3Object(s3Object)
                    .build();

            StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                    .documentLocation(location)
                    .featureTypes(myList)
                    .build();

            StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

            // Get the job ID
            String jobId = response.jobId();
            return jobId;

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "" ;
    }

    private static String getJobResults(TextractClient textractClient, String
jobId) {

        boolean finished = false;
        int index = 0 ;
        String status = "" ;

       try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                    .jobId(jobId)
                    .maxResults(1000)
                    .build();

            GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
```

```
                finished = true;
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++ ;
        }
        return status;

       } catch( InterruptedException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
       return "";
    }
```

- 在 GitHub 中查找说明和更多代码。
- 有关 API 详细信息，请参阅StartDocumentAnalysis在AWS SDK for Java 2.xAPI 参考.

Python

适用于 Python (Boto3) 的 SDK

启动异步作业以分析文档。

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_analysis_job(
            self, bucket_name, document_file_name, feature_types, sns_topic_arn,
            sns_role_arn):
        """
```

```
        Starts an asynchronous job to detect text and additional elements, such
as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract
publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
image.
        :param document_file_name: The name of the document image stored in
Amazon S3.
        :param feature_types: The types of additional document features to
detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
topic
                              where job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
(IAM)
                             role that can be assumed by Textract and grants
permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name':
document_file_name}},
                NotificationChannel={
                    'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn},
                FeatureTypes=feature_types)
            job_id = response['JobId']
            logger.info(
                "Started text analysis job %s on %s.", job_id,
document_file_name)
        except ClientError:
            logger.exception("Couldn't analyze text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- 在 GitHub 中查找说明和更多代码。

- 有关 API 详细信息，请参阅StartDocumentAnalysis在AWSSDK for Python (Boto3) 的 API 参考.

有关的完整列表AWSSDK 开发人员指南和代码示例，请参阅将 Amazon Textract 与AWS开发工具包. 本主题还包括有关入门的信息以及有关以前 SDK 版本的详细信息。

使用 Amazon Textract 和AWS开发工具包

以下代码示例显示如何使用 Amazon Textract 启动文档中的异步文本检测。

Python

适用于 Python (Boto3) 的 SDK

启动异步作业以检测文档中的文本。

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_detection_job(
            self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn):
        """
        Starts an asynchronous job to detect text elements in an image stored in
 an
        Amazon S3 bucket. Textract publishes a notification to the specified
 Amazon SNS
        topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
 image.
        :param document_file_name: The name of the document image stored in
 Amazon S3.
```

```
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
topic
                              where the job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
(IAM)
                             role that can be assumed by Textract and grants
permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_text_detection(
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name':
document_file_name}},
                NotificationChannel={
                    'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn})
            job_id = response['JobId']
            logger.info(
                "Started text detection job %s on %s.", job_id,
document_file_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- 在 GitHub 中查找说明和更多代码。
- 有关 API 详细信息，请参阅StartDocumentTextDetection在AWSSDK for Python (Boto3) 的 API 参考.

有关的完整列表AWSSDK 开发人员指南和代码示例，请参阅将 Amazon Textract 与AWS开发工具包. 本主题还包括有关入门的信息以及有关以前 SDK 版本的详细信息。

适用于 Amazon Textract 的跨服务示例

以下示例应用程序使用AWS软件开发工具包将 Amazon Textract 与其他组合AWS服务。每个示例都包含指向 GitHub 的链接，您可以在其中找到如何设置和运行应用程序的说明。

示例

- 创建 Amazon Textract 浏览器应用程序
- 使用从图像中提取的文本中检测实体AWS开发工具包

创建 Amazon Textract 浏览器应用程序

以下代码示例展示如何通过交互式应用程序探索 Amazon Textract 输出。

JavaScript

SDK for JavaScript V3

展示如何使用 适用于 JavaScript 的 AWS SDK 构建 React 应用程序，该应用程序使用 Amazon Textract 从文档图像中提取数据并在交互式网页中显示该数据。此示例在 Web 浏览器中运行，需要经过身份验证的 Amazon Cognito 身份才能获得凭证。它使用 Amazon Simple Storage Service (Amazon S3) 进行存储；对于通知，它将轮询订阅 Amazon Simple Notification Service (Amazon SNS) 主题的 Amazon Simple Queue Service (Amazon SQS) 队列。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

适用于 Python (Boto3) 的 SDK

展示如何将 适用于 Python (Boto3) 的 AWS SDK 和 Amazon Textract 一起使用来检测文档图像中的文本、表单和表格元素。输入图像和 Amazon Textract 输出在 Tkinter 应用程序中显示，该应用程序可让您探索检测到的元素。

- 将文档图像提交到 Amazon Textract 并探索检测到的元素的输出。
- 将图像直接提交到 Amazon Textract，或通过 Amazon Simple Storage Service (Amazon S3) 存储桶提交图像。

- 使用异步 API 启动任务，在任务完成后将通知发布到 Amazon Simple Notification Service (Amazon SNS) 主题。
- 轮询 Amazon Simple Queue Service (Amazon SQS) 队列，以获取任务完成消息并显示结果。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

有关的完整列表AWSSDK 开发人员指南和代码示例，请参阅将 Amazon Textract 与AWS开发工具包. 本主题还包括有关入门的信息以及有关以前 SDK 版本的详细信息。

使用从图像中提取的文本中检测实体AWS开发工具包

以下代码示例演示了如何使用 Amazon Comprehend 检测 Amazon Textract 从存储在 Amazon S3 中的图像中提取的文本中的实体。

Python

适用于 Python (Boto3) 的 SDK

显示如何使用适用于 Python (Boto3) 的 AWS SDK在 Jupyter 笔记本中检测从图像中提取的文本中的实体。此示例使用 Amazon Textract 从 Amazon S3 中存储的图像中提取文本，然后使用 Amazon ComplAmazon Comprehend 检测提取文本中的实体。

此示例是 Jupyter 笔记本，必须在可以托管笔记本电脑的环境中运行。有关如何使用 Amazon SageMaker 运行示例的说明，请参阅T提取并理解笔记本电脑.ipynb.

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Comprehend
- Amazon S3
- Amazon Textract

有关的完整列表AWSSDK 开发人员指南和代码示例，请参阅将 Amazon Textract 与AWS开发工具包. 本主题还包括有关入门的信息以及有关以前 SDK 版本的详细信息。

使用亚马逊 Augmented AI 将人工评论添加到 Amazon Textract 输出中

Amazon Augmented AI (Amazon A2I) 是一项机器学习 (ML) 服务，可轻松构建人工审核机器学习分析的工作流程。

Amazon Textract 与 Amazon A2I 集成。您可以使用它将置信度较低的文档分析结果发送给人工审阅者。

你可以使用 Amazon Textract`AnalyzeDocument`用于从表单和 Amazon A2I 控制台中提取数据的 API。您可以指定 Amazon A2I 将预测发送给审阅者的条件。您可以根据重要表单键的置信度阈值设置条件。例如，如果有密钥，则可以将文档发送给人工审阅者名称或者它的相关价值Jane Doe被检测到的信心很低。

主题

- Amazon A2I 的核心概念
- 使用 Amazon A2I

Amazon A2I 的核心概念

查看以下术语以熟悉 Amazon A2I 的核心概念。

人工审核激活条件

您可以使用 Amazon A2I激活条件以指定何时将文档发送给人类进行审阅，以及要求工作人员审阅的表单内容。

例如，当信心低下检测到重要密钥时，您可以设置激活条件，使 Amazon Textract 路由表格到亚马逊 A2I，例如电话号码. 在此示例中，要求人工评论者查看电话号码Amazon Textract 检测到的关联字段和值。

您可以使用以下激活条件来指定何时将表单发送给人类进行审查：

- 根据表单键置信度分数触发对特定表单键的人工审查。人工审核者需要审查这些表单键和相关值。
- 在特定表单键丢失时触发人工审查。要求人类审阅者确定这些表单键和相关值。
- 针对指定范围内的置信度分数触发对 Amazon Textract 识别的所有表单键的人工审核。

- 随机将表单示例发送给工作人员以进行审查。人工审核者需要审查 Amazon Textract 检测到的所有表单键和值。

当激活条件依赖于表单键置信度分数时，您可以使用两种类型的预测置信度阈值来触发人工审核：

- 识别信心— 在表单中检测到的键值对的置信度分数。
- 资格审查信— 表单中键值对中包含的文本的置信度分数。

如果您指定置信度阈值，亚马逊 A2I 仅将那些属于阈值以内的预测发送给人类评论者。您可以随时调整这些阈值，以便在准确性和成本效益之间取得适当的平衡。这可以帮助您实施审计以定期监控预测准确性。

> **Note**
>
> 您可以使用 Amazon A2I 自定义任务类型进一步自定义将文档发送给人类以供审查的条件。使用此任务类型，您可以直接在应用程序中指定人工审核的条件。有关信息，请参阅将 Amazon Adure Augmented AI 与自定义任务类型结合（在 Amazon SageMaker 开发人员指南中）。

人工审查工作流程（流程定义）

你使用人工审查工作流，也称作流定义，以指定用于创建人工审核工作流程的资源，以及指定激活条件。

您指定的资源包括：

- 具有调用 Amazon A2I API 操作权限的 IAM 角色
- 您希望将人工审核输出存储到的 Amazon S3 存储桶。
- 你的人力工作团队
- 一个员工任务模板其中包括帮助员工完成审查任务的说明和示例

您还可以使用人工审核工作流程来指定激活条件。有关更多信息，请参阅 人工审核激活条件。

您可以使用单个人工审核工作流程创建多个人类循环.

您可以在 SageMaker 控制台或使用 SageMaker API 创建人工审核工作流程。有关信息，请参阅 创建人工审核工作流程。

工作者任务模板

你使用员工任务模板创建用于人工审查任务的工作人员 UI。

工作人员 UI 会显示您的文档和工作人员说明。它还提供了工作人员可用于完成任务的工具。

创建人工审核工作流程时，您可以使用 SageMaker 控制台配置您的工作人员任务模板。有关信息，请参阅 创建人工审核工作流程。

工作团队

一个工作团队是您将人工审查任务发送给的一群人类工作者。

当您创建人工审核工作流程时，您需要指定单个工作团队。

借助 Amazon A2I，您可以使用自己的组织中的审核人员池。您还可以访问由 500,000 多名独立承包商组成的员工队伍，他们已经通过 Amazon Mechanical Turk 执行机器学习任务。另一种选择是使用经 AWS 预先筛选的供应商，以确保质量和遵守安全程序。

Amazon A2I 还为审阅者提供了一个 Web 界面，其中包含完成审核任务所需的所有说明和工具。

对于每种类型的人力（私有人力、供应商和 Mechanical Turk），您可以创建多个工作团队。您可以在多个人工审核工作流程中使用每个工作团队。要了解如何创建人力和工作团队，请参阅创建和管理人力（在 Amazon SageMaker 开发人员指南中）。

> **Important**
>
> 单击这里以查看目前涵盖亚马逊 Augmented AI 的合规性计划。请注意，如果您将 Amazon Augmented AI 与其他 AWS 服务（例如 Amazon Rekognition 和 Amazon Textract）一起使用，则 Amazon Augmented AI 可能不在与其他服务相同的合规性计划范围内。您需要对如何使用 Amazon Augmented AI 负责，包括了解该服务将如何处理或存储客户数据及其对数据环境合规性的任何影响。您应与 AWS 账户团队讨论自己的工作负载目标；他们可以帮助您评估该服务是否非常适合您提出的使用案例和架构。
> 目前，除了公共和供应商劳动力案例外，亚马逊 Augmented AI 符合 PCI 标准。
> 有关亚马逊 Augmented AI HIPAA 合规性的信息，请单击这里.

人类循环

您可以使用人工循环创建人工审核任务。

您将人工审查任务分配给人工团队中的工作人员。要求工作人员查看 Amazon Textract 在您在激活条件中指定的输入文档中检测到的键值对。

例如，假设一张照片对其中包含苹果的信心在 50% 到 60% 之间。这属于人类审查的信心阈值之内，并发送给工作人员。工作人员检查文档中是否有苹果，将其标记为包含或不包含苹果。然后，亚马逊 A2I 将文档发回工作流程。

当你打电话给 Amazon Textract`AnalyzeDocument`并指定人工审阅工作流程（流程定义）和人工循环名称，当满足人工审核工作流程中指定的激活条件时，将创建人工审阅任务。这些任务是使用您在人工审核工作流程中指定的资源创建的。

使用 Amazon A2I

以下步骤可帮助您将 Amazon A2I 集成到 Amazon Textract 单页文档分析任务中。您应当执行以下操作：

1. 使用 Amazon A2I 控制台（建议新用户使用）或 Amazon A2I API 创建人工审核工作流程。
2. 要分析表单并在必要时包括人工评论，请使用`AnalyzeDocument`操作并指定人工审核工作流程的 Amazon 资源名称 (ARN)。响应告诉您是否需要人工审核。
3. 使用 Amazon A2I 控制台和 API 监控您的人工循环。
4. 在发送结果的 Amazon S3 存储桶中查看人工审查结果。

要设置 SageMaker 笔记本实例并使用示例笔记本，请参阅使用 Amazon Textract 和 Augmented AI 的端到端演示中的Amazon SageMaker 开发人员指南

> **Note**
>
> 本节介绍了如何为 Amazon A2I（Amazon Textract）任务类型创建人工审核工作流程。要进一步自定义亚马逊 A2I 和 Amazon Textract 集成，您可以使用 Amazon A2I 自定义任务类型。使用此选项，您可以提供自定义工作人员任务模板，并指定直接在应用程序中发送文档以供人工审阅的条件。想要了解有关信息，请参阅将 Amazon Adure Augmented AI 与自定义任务类型结合中的Amazon SageMaker 开发人员指南.

主题

- 创建人工审核工作流程
- 分析文档

- 监控人工循环
- 查看输出数据和工作人员指标

创建人工审核工作流程

您可以使用 Amazon A2I 控制台（建议新用户使用）或 Amazon A2I 创建人工审核工作流程`CreateFlowDefinition`operation.

主题

- 创建人工审核工作流程（控制台）
- 创建人工审核工作流程 (API)

创建人工审核工作流程（控制台）

您可以在 Amazon S3 中使用自己的文档完成此示例，也可以下载此示例文档然后将其放入您的 Amazon S3 存储桶中。

确保 S3 存储桶位于同一个位置AWS您使用的区域是 Amazon Textract。要创建存储桶，请参阅创建存储桶中的Amazon Storage Service 控制台用户指南.

> **Note**
>
> Amazon A2I 控制台嵌入在 SageMaker 控制台中。要使用控制台，您需要访问 SageMaker 控制台和创建工作团队的权限。要开始，您可以使用AmazonSageMakerFullAccess包括在 SageMaker 中执行大多数操作的所有必需权限的 IAM 托管策略。有关更多信息，请参阅 。适用于 Amazon SageMaker 的 Identity and Access Management中的Amazon SageMaker 开发人员指南.

主题

- 第 1 步：创建工作团队（控制台）
- 第 2 步：创建人工审核工作流程（控制台）

第 1 步：创建工作团队（控制台）

首先，在 Amazon A2I 控制台中创建一个工作组并将自己添加为工作人员，以便您可以在工作人员门户中预览人工审核任务，工作团队成员可以查看分配给他们的不同任务和文档。

使用工作人员电子邮件创建私有人力 (控制台)

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中的下Ground Truth，选择为人力添加标签.
3. 选择 Private (私有)，然后选择 Create private team (创建私有团队)。
4. 选择 Invite new workers by email (通过电子邮件邀请新工作人员)。
5. 在此示例中，请输入您的电子邮件地址和您希望能够预览工作人员门户的其他任何其他人的电子邮件地址。您可以将包含最多 50 个电子邮件的列表（以逗号分隔）粘贴或键入到电子邮件地址。
6. 输入一个组织名称和联系人电子邮件
7. 选择 Create private team (创建私有团队)。

如果您将自己添加到私人工作团队，则会收到来自的电子邮件no-reply@verificationemail.com带登录信息。使用此电子邮件中的链接重置密码并登录工作人员门户。在你打电话后，你的人工评论任务将出现在这里AnalyzeDocument.

第 2 步：创建人工审核工作流程（控制台）

在此步骤中，您将创建 Amazon Textract 人工审核工作流程。

要创建人工审核工作流程（控制台）

1. 在以下位置打开 Amazon A2I 控制台https://console.aws.amazon.com/a2i访问人工审核工作流页.
2. 选择创建人工审核工作流.
3. 适用于名称中，输入工作流程名称。
4. 适用于S3 bucket中，选择您希望 Amazon A2I 将人工审核任务结果存储到的存储桶。如果您不选择存储桶，请更改该存储桶以输入存储桶的名称。
5. UNDERIAM 角色，请选择创建新角色. 显示一个窗口，其中包含标题创建 IAM 角色. 使用此窗口指定您希望此角色有权访问的 Amazon S3 存储桶。如果你没有选择任何 S3 存储桶中，指定您在步骤 4 中指定的输出存储桶以及包含输入文档的存储桶。
6. 适用于任务类型，选择Textract-键值对提取。
7. InAmazon Textract 表单提取-调用人工评论的条件中，指定激活条件。我们建议您为文档中至少一个键设置置置置信度高阈值，以触发人工审核，以便您可以在工作人员门户中预览工作人员任务。

   如果您使用本演练中提供的示例文档，请按如下方式指定激活条件：

   a. 选择根据表单键置信度分数或特定表单键缺失时触发针对特定表单键的人工审查。

b. 适用于键名称输入**`Mail Address`**.

c. 设置识别信心介于阈值0和99.

d. 设置资格审查信介于阈值0和99.

e. 选择针对 Amazon Textract 识别的所有表单键的人工审查，并在特定范围内的置信度分数。

f. 适用于**`identification confidence`**，请选择 0 和 90 之间的任何数字。

g. 适用于**`qualification confidence`**，请选择 0 和 90 之间的任何数字。

如果 Amazon Textract 返回的信心分数低于 99，则会触发人工评论邮寄地址及其值，或者如果文档中检测到的任何键值对返回的置信度分数低于 90。

8. UNDER创建员工任务模板，请选择从默认模板创建.
9. 适用于模板名称，输入一个描述性名称。
10. 适用于任务描述，添加类似如下所示的内容：

```
Read the instructions and review the document.
```

11. 适用于员工选择私密.
12. 从菜单中选择您创建的私人团队。
13. 选择 Create (创建)。

创建人工审核工作流程后，它将显示在人工审核工作流页. 当状态是处于活动状态中，复制并保存工作流 ARN。

创建人工审核工作流程 (API)

您可以创建人工审核工作流程，或流定义使用 Amazon A2I，`CreateFlowDefinition`operation.

在此示例中，您可以在 Amazon S3 中使用自己的文档，也可以下载此示例文档然后将其存储在 S3 存储桶中。

确保您的 Amazon S3 存储桶位于同一位置AWS你计划用来调用的区域`AnalyzeDocument`. 要创建存储桶，请按照中的说明操作创建存储桶中的Amazon Storage Service 控制台用户指南.

先决条件

要使用 Amazon A2I API 创建人工审核工作流程，您必须完成以下先决条件：

- 配置 IAM 角色，同时调用 Amazon A2I 和 Amazon Textract API 操作的权限。要开始使用，您可以将 AWS 策略、AmazonAugmentedAIFullAccess 和 AmazonTextractFullAccess 附加到 IAM 角色。记录 IAM 角色 Amazon 资源名称 (ARN)，以后您将需要它。

  有关使用 Amazon Textract 时的更精细的权限，请参阅Amazon Textract 基于身份的策略示例. 对于 Amazon A2I，请参阅Amazon Advanced Augmented AI 中的权限和安全性中的Amazon SageMaker 开发人员指南.
- 创建一个私人工作团队并记录工作组 ARN。如果您是 Amazon A2I 的新用户，请按照中的说明进行操作。第 1 步：创建工作团队（控制台）.
- 创建工作人员任务模板。按照中的说明进行操作创建工作线程任务模板以使用 Amazon A2I 控制台创建模板。在创建模板时，选择提取形式提取为了模板类型. 在模板中，替换`s3_arn`将与您的文档的 Amazon S3 ARN 结合在一起。在中添加其他员工说明`<full-instructions header="Instructions"></full-instructions>`.

  如果您要预览模板，请确保您的 IAM 角色具有中介绍的权限。启用工作人员任务模板预览.

  创建模板后，记录工作人员任务模板 ARN。

您使用在中创建的资源先决条件配置`CreateFlowDefinition`请求. 在此请求中，您还要指定 JSON 格式的激活条件。要了解如何配置激活条件，请参阅将人工循环激活条件 JSON 架构与 Amazon Textract 一起使用.

创建人工审核工作流程（适用于 Python 的 AWS 开发工具包 (Boto3)）

要使用此示例，请替换*##*将文本包含您的规范和资源。

首先，使用以下代码将激活条件编码为 JSON 对象。如果 Amazon Textract 返回的信心分数低于 99，则会触发人工评论邮寄地址及其值，或者如果文档中检测到的任何键值对返回的置信度分数低于 90。如果您使用本示例中提供的示例文档，则这些激活条件将创建人工审阅任务。

```
import json

humanLoopActivationConditions = json.dumps("{
                "Conditions": [
                    {
                        "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "Mail Address",
                            "KeyValueBlockConfidenceLessThan": 99,
                            "WordBlockConfidenceLessThan": 99
```

```
                    }
                },
                {
                    "ConditionType": "ImportantFormKeyConfidenceCheck",
                    "ConditionParameters": {
                        "ImportantFormKey": "*",
                        "KeyValueBlockConfidenceLessThan": 90,
                        "WordBlockConfidenceLessThan": 90
                    }
                }
            ]
        }"
)
```

使用humanLoopActivationConditions要配置create_flow_definition请求. 以下示例使用 SDK for Python (Boto3) 调用create_flow_definition在 us-west-2 区 AWS 中。它指定使用私有工作团队。

```
response = client.create_flow_definition(
    FlowDefinitionName='string',
    HumanLoopRequestSource={
         'AwsManagedHumanLoopRequestSource': "AWS/Textract/AnalyzeDocument/Forms/V1"
    },
    HumanLoopActivationConfig={
        'HumanLoopActivationConditionsConfig': {
            'HumanLoopActivationConditions': humanLoopActivationConditions
        }
    },
    HumanLoopConfig={
        'WorkteamArn': "arn:aws:sagemaker:us-west-2:111122223333:workteam/private-
crowd/work-team-name",
        'HumanTaskUiArn': "arn:aws:sagemaker:us-west-2:111122223333:human-task-
ui/worker-task-template-name",
        'TaskTitle': "Add a task title",
        'TaskDescription': "Describe your task",
        'TaskCount': 1,
        'TaskAvailabilityLifetimeInSeconds': 3600,
        'TaskTimeLimitInSeconds': 86400,
        'TaskKeywords': ["Document Review", "Content Review"]
        }
    },
    OutputConfig={
        'S3OutputPath': "s3://DOC-EXAMPLE-BUCKET/prefix/",
```

```
    },
    RoleArn="arn:aws:iam::111122223333:role/role-name"
)
```

分析文档

要将 Amazon A2I 融入到 Amazon Textract 文档分析工作流程中，您可以配置HumanLoopConfig中的AnalyzeDocumentoperation.

InHumanLoopConfig，您可以在中指定您的人工审核工作流（流定义）ARNFlowDefinitionArn，然后给你的人类循环一个名字HumanLoopName.

Analyze the Document (AWS SDK for Python (Boto3))

以下示例使用 SDK for Python (Boto3) 调用analyze_document在 us-west-2 中。替换#####用你的资源发短信。有关更多信息，请参阅 。分析 _文档中的AWS适用于 Python 的开发工具包 (Boto) API 参考.

```
client.analyze_document(Document={'S3Object': {"Bucket": "DOC-EXAMPLE-BUCKET",
 "Name": "document-name.png"}},
        HumanLoopConfig={"FlowDefinitionArn":"arn:aws:sagemaker:us-
west-2:111122223333:flow-definition/flow-definition-name",
                         "HumanLoopName":"human-loop-name",
                         "DataAttributes":{"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent",]}},
        FeatureTypes=["FORMS"])
```

Analyze the Document (AWS CLI)

以下示例使用AWS要打电话的 CLIanalyze_document. 这些示例与AWSCLI 版本 2。第一个是速记语法，第二个是 JSON 语法。有关更多信息，请参阅 。分析文档中的AWS CLI命令参考.

```
aws textract analyze-document \
     --document '{"S3Object":{"Bucket":"bucket_name","Name":"file_name"}}' \
     --human-loop-config
 HumanLoopName="test",FlowDefinitionArn="arn:aws:sagemaker:eu-west-1:xyz:flow-
definition/
hl_name",DataAttributes='{ContentClassifiers=["FreeOfPersonallyIdentifiableInformation","Fre
```

```
     --feature-types '["FORMS"]'
```

```
aws textract analyze-document \
     --document '{"S3Object":{"Bucket":"bucket_name","Name":"file_name"}}' \
     --human-loop-config \
          '{"HumanLoopName":"test","FlowDefinitionArn":"arn:aws:sagemaker:eu-
west-1:xyz:flow-definition/hl_name","DataAttributes": {"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation","FreeOfAdultContent"]}}' \
     --feature-types '["FORMS"]'
```

> **Note**
>
> 避免在 —人循环配置参数中使用白色空格，因为这可能会导致代码的处理问题。

对此请求的回复包括HumanLoop 激活输出，它表明是否创建了人类循环，如果是的话，为什么。如果创建了人类循环，则此对象还包含HumanLoopArn.

有关和示例的更多信息，请参阅AnalyzeDocument操作，请参阅使用 Amazon Textract 分析文档文本.

监控人工循环

您可以使用 Amazon A2I 控制台和 API 查看有关人类循环的详细信息，并在出现错误时停止活动的人类循环。

查看人工循环详情

您可以在 Amazon A2I 控制台中查看人类循环状态，也可以使用Amazon A2I 运行时 API.

查找有关人类循环的详细信息（控制台）

1. 在以下位置打开 Amazon A2I 控制台https://console.aws.amazon.com/a2i访问人工审核工作流页.
2. 选择您还用于配置的人工审核工作流程HumanLoopConfig在AnalyzeDocument.
3. 在人类循环部分中，选择要查看其详细信息的人工循环。

要查找有关你的人类循环 (API) 的详细信息：

使用 Amazon A2IDescribeHumanLoopoperation. 指定你用来调用的人类循环名称AnalyzeDocument.

以下适用于 Python (Boto3) 的示例调用describe_human_loop.

```
response = client.describe_human_loop(HumanLoopName="human-loop-name")
```

停止人工循环

人类循环开始后，您可以使用 Amazon A2I 控制台和 API 将其停止。

停止你的人类循环（控制台）

1. 在以下位置打开 Amazon A2I 控制台https://console.aws.amazon.com/a2i访问人工审核工作流页.
2. 选择您用于配置的人工审核工作流程HumanLoopConfig中的AnalyzeDocumentoperation.
3. 在人类循环部分中，选择要停止的人工循环。
4. 选择 Stop (停止)。

停止你的人类循环（API）

使用 Amazon A2IStopHumanLoopoperation. 指定您用于调用的人工循环的名称。AnalyzeDocument.

以下 SDK for Python (Boto3) 的示例调用stop_human_loop.

```
response = client.stop_human_loop(HumanLoopName="human-loop-name")
```

查看输出数据和工作人员指标

当工作人员完成人工审查任务时，Amazon A2I 会将您的输出数据存储在您在人工审核工作流程中指定的 Amazon S3 存储桶中。

如果您使用私人劳动力，则输出数据包含可用于跟踪个人员活动的员工元数据。

在 Amazon S3 中查找输出数据

Amazon A2I 使用您的人工审核工作流程名称作为存储使用该人工审核工作流程创建的人类循环输出数据的文件名称的前缀。

人类循环输出的路径使用以下模式，其中*YYYY*/*MM*/*DD*/*hh*/*mm*/*ss*用年份表示人类循环创建日期（YYYY)、月 (MM）、和天（DD) 以及创建时间与小时 (hh)、分钟 (mm）、第二个（ss)。

```
s3://output-bucket-specified-in-flow-definition/flow-definition-
name/YYYY/MM/DD/hh/mm/ss/human-loop-name/output.json
```

要查看人类循环的输出，请使用 Amazon A2I 控制台。

要查看人类循环输出

1. 在以下位置打开 Amazon A2I 控制台https://console.aws.amazon.com/a2i访问人工审核工作流页.
2. 选择您用于配置的人工审核工作流程HumanLoopConfig在AnalyzeDocument.
3. 在人类循环部分中，选择要查看其输出的人类循环。
4. UNDER输出位置，请选择指向输出数据的链接。

跟踪私人员活动

当您使用私人工作人员执行人工审查任务时，输出数据包括有关完成审核的员工的以下信息：

- 该 workerId。
- In workerMetadata:
  - identityProviderType— 用于管理私有人力的服务。
  - issuer— 与分配给此人工审核任务的工作团队关联的 Amazon Cognito 用户池或 OIDC 身份提供商 (IdP) 发行者。
  - sub— 表示工作人员的唯一标识符。如果您使用 Amazon Cognito 创建了人力，则可以使用 Amazon Cognito 使用此 ID 检索有关此工作人员的详细信息（例如姓名或用户名）。要了解如何操作，请参阅管理和搜索用户账户在Amazon Cognito 开发人员指南.

以下是如果您使用 Amazon Cognito 创建私有人力，您可能会看到的输出示例。

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Cognito",
                    "issuer": "https://cognito-idp.aws-region.amazonaws.com/aws-
region_123456789",
                    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
```

以下是您是否使用自己的 OIDC IdP 来创建私人劳动力队伍时可能会看到的输出示例：

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Oidc",
                    "issuer": "https://example-oidc-ipd.com/adfs",
                    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
```

要了解有关使用私有人力的更多信息，请参阅使用私有人力中的Amazon SageMaker 开发人员指南.

Amazon Textract 中的安全性

AWS 的云安全性具有优先级最高。作为 AWS 客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

使用以下主题了解如何保护您的 Amazon Textract 资源。

主题

- Amazon Textract 中的数据保护
- 适用于 Amazon Textract 的 Identity and Access Management
- 日志记录和监控
- 使用 记录 Amazon Textract API 调用AWS CloudTrail
- Amazon Textract 的合规性验证
- Amazon Textract 中的恢复能力
- Amazon Textract 中的基础设施安全性
- Amazon Textract 中的配置和漏洞分析
- Amazon Textract 和接口 VPC 终端节点 (AWS PrivateLink）

Amazon Textract 中的数据保护

Amazon Textract 符合AWS 责任共担模式，其中包括数据保护的法规和指南。AWS负责保护运行所有AWS服务。AWS维护对此基础设施上托管的数据的控制，包括用于处理客户内容和个人数据的安全配置控制。AWS和客户APN合作伙伴（充当数据控制者或数据处理者）负责他们在AWS云。

出于数据保护目的，我们建议您保护 AWS 账户凭证并使用 AWS Identity and Access Management (IAM) 设置单独的用户账户。这可以确保仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 AWS 资源进行通信。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如 Name (名称) 字段）。这包括使用 Amazon Textract 或其他AWS使用控制台、API、AWS CLI，或者AWS开发工具包。您输入到 Amazon Textract 或其他服务中的任何数据都可能被选取以包含在诊断日志中。当您向外部服务器提供 URL 时，请勿在 URL 中包含凭证信息来验证您对该服务器的请求。

有关数据保护的更多信息，请参阅AWS安全性博客上的AWS责任共担模式和 GDPR 博客文章。

Amazon Textract 中的加密

数据加密是指在传输中和静态数据时保护数据。您可以使用 Amazon S3 托管密钥来保护您的数据或 AWS KMS key在运输过程中，还有标准传输层安全性。

静态加密

在 Amazon Textract 中加密数据的主要方法是服务器端加密。从 Amazon S3 存储桶传递的输入文档由 Amazon S3 加密，并在您访问它们时进行解密。只要您验证了您的请求并且拥有访问权限，您访问加密和未加密对象的方式就没有区别。例如，如果您使用预签名的 URL 来共享您的对象，那么对于加密和解密对象，该 URL 的工作方式是相同的。此外，当您列出存储桶中的对象时，`List`API 会返回所有对象的列表（无论对象是否加密）。

Amazon Textract 使用两种互斥的服务器端加密方法。

具有 Amazon S3 托管密钥的服务器端加密 (SSE-S3)

当您将服务器端加密用于 Amazon S3 托管密钥 (SSE-S3) 时，每个对象均使用唯一密钥加密。作为额外的保护，此方法将使用定期轮换的主密钥加密密密密钥本身。Amazon S3 服务器端加密使用可用的最强数据块密码之一、256 位高级加密标准 (AES-256) 来加密您的数据。有关更多信息，请参阅使用服务器端加密与 Amazon S3 托管加密密钥 (SSE-S3) 保护数据。

使用存储在 AWS Key Management Service 中的 KMS 密钥的服务器端加密 (SSE-KMS)

在 AWS Key Management Service 中存储 KMS 密钥的服务器端加密 (SSE-KMS) 与 SSE-S3 类似，使用该服务具有一些额外的好处，但也要额外收取费用。使用 KMS 密钥需要单独的权限，该密钥可进一步防止未经授权地访问 Amazon S3 中的对象。SSE-KMS 还向您提供审核跟踪，显示您的 KMS 密钥的使用时间和使用者。此外，您可以创建和管理 KMS 密钥或使用AWS 托管式密钥这是您、服务和区域独有的。有关更多信息，请参阅 。使用服务器端加密保护数据将 KMS 密钥存储在 AWS Key Management Service (SSE-KMS) 中。

传输中加密

对于传输中的数据，Amazon Textract 使用传输层安全性 (TLS) 加密服务和代理之间发送的数据。此外，Amazon Textract 使用 VPC 终端节点在 Amazon Textract 处理文档时使用的各种微服务之间发送数据。

互联网络流量保密性

Amazon Textract 仅通过 HTTPS 终端节点进行通信，Amazon Textract 支持的所有区域都支持这些端点

适用于 Amazon Textract 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一种 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制哪些人可以身份验证（已登录）和授权（有权限）可以使用 Amazon Textract 资源。IAM 是一个可以免费使用的 AWS 服务。

主题

- Audience
- 使用身份进行身份验证
- 使用策略管理访问
- Amazon Textract 如何与 IAM 配合使用
- Amazon Textract 基于身份的策略示例
- 排查 Amazon Textract 身份和访问的问题

Audience

如何使用AWS Identity and Access Management(IAM) 因您在 Amazon Textract 中执行的操作而异。

服务用户— 如果您使用 Amazon Textract 服务来完成作业，则您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon Textract 功能来完成工作，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Amazon Textract 中的功能，请参阅排查 Amazon Textract 身份和访问的问题.

服务管理员— 如果您在公司负责管理 Amazon Textract 资源，您可能对 Amazon Textract 具有完全访问权限。您有责任确定您的员工应访问哪些 Amazon Textract 功能和资源。然后，您必须向 IAM 管理

员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon Textract 搭配使用的更多信息，请参阅Amazon Textract 如何与 IAM 配合使用.

IAM 管理员— 如果您是 IAM 管理员，您可能希望了解有关您可以如何编写策略以管理对 Amazon Textract 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon Textract 基于身份的策略示例，请参阅Amazon Textract 基于身份的策略示例.

使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。有关使用AWS 管理控制台登录的更多信息，请参阅 IAM 用户指南中的以 IAM 用户或根用户身份登录AWS 管理控制台。

您必须作为AWS 账户根用户、IAM 用户或代入 IAM 角色以进行身份验证（登录到AWS）。您还可以使用公司的单一登录身份验证方法，甚至使用 Google 或 Facebook 登录。在这些情况下，您的管理员以前使用 IAM 角色设置了联合身份验证。在您使用来自其它公司的凭证访问 AWS 时，您间接地代入了角色。

要直接登录到AWS 管理控制台，请将密码与根用户电子邮件地址或 IAM 用户名一起使用。您可以使用根用户或 IAM 用户访问密钥以编程方式访问 AWS。AWS 提供了 SDK 和命令行工具，可使用您的凭证对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求签名。使用 Signature Version 4（用于对入站 API 请求进行验证的协议）完成此操作。有关验证请求的更多信息，请参阅《AWS 一般参考》中的 Signature Version 4 签名流程。

无论使用何种身份验证方法，您可能还需要提供其它安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的在 AWS 中使用多重身份验证 (MFA)。

AWS 账户根用户

当您首次创建AWS 账户时，最初使用的是一个对账户中所有AWS服务和资源有完全访问权限的单点登录身份。此身份称为AWS 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不使用根用户执行日常任务，即使是管理任务。相反，请遵循仅使用根用户创建您的第一个 IAM 用户的最佳实践。然后请妥善保存根用户凭证，仅用它们执行少数账户和服务管理任务。

IAM 用户和组

IAM 用户是AWS 账户内对某个人员或应用程序具有特定权限的一个身份。IAM 用户可能具有长期凭证，例如用户名和密码或一组访问密钥。要了解如何生成访问密钥，请参阅 IAM 用户指南 中的管理

IAM 用户的访问密钥。为 IAM 用户生成访问密钥时，请确保查看并安全保存密钥对。您以后无法找回秘密访问密钥，而是必须生成新的访问密钥对。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是AWS 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS 管理控制台 中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义 URL 以代入角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 临时 IAM 用户权限 – IAM 用户可以代入 IAM 角色，以暂时获得不同的权限以执行特定的任务。
- 联合身份用户访问 – 您可以不创建 IAM 用户，而是使用来自 Directory Service、您的企业用户目录或 Web 身份提供商的现有身份。这些用户被称为联合用户。在通过身份提供商请求访问权限时，AWS将为联合身份用户分配角色。有关联合身份用户的更多信息，请参阅 IAM 用户指南 中的联合身份用户和角色。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信委托人）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南 中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 AWS 服务使用其它 AWS 服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的委托人的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 委托人权限 – 当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为委托人。策略向委托人授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他相关操作，请参阅Amazon Textract 的操作、资源和条件键中的服务授权参考.

- 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 AWS 服务委派权限的角色。
- 服务相关角色 – 服务相关角色是与 AWS 服务关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南 中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 IAM 身份或AWS资源，以便控制AWS中的访问。策略是AWS中的对象；在与标识或资源相关联时，策略定义它们的权限。您可以通过 root 用户或 IAM 用户身份登录，也可以代入 IAM 角色。随后，当您提出请求时，AWS会评估相关的基于身份或基于资源的策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个委托人可以对什么资源执行操作，以及在什么条件下执行。

每个 IAM 实体（用户或角色）最初没有任何权限。换言之，预设情况下，用户什么都不能做，甚至不能更改他们自己的密码。要为用户授予执行某些操作的权限，管理员必须将权限策略附加到用户。或者，管理员可以将用户添加到具有预期权限的组中。当管理员为某个组授予访问权限时，该组内的全部用户都会获得这些访问权限。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS 管理控制台、AWS CLI 或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅 IAM 用户指南中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到AWS 账户中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户托管式策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略 和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定委托人可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定委托人。委托人可以包括账户、用户、角色、联合身份用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 AWS 托管式策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 IAM 实体的权限边界。

- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体的权限，包括每个 AWS 账户 根用户。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon Textract 如何与 IAM 配合使用

在使用 IAM 管理对 Amazon Textract 的访问权限之前，您应该了解哪些 IAM 功能可用于 Amazon Textract。要高级了解 Amazon Textract 和其他方式AWS使用 IAM 的服务，请参阅AWS使用 IAM 的 IAM 服务中的IAM 用户指南.

主题

- Amazon Textract 基于身份的策略
- Amazon Textract 基于资源的策略
- 基于 Amazon Textract 的标签的授权
- Amazon Textract IAM 角色

Amazon Textract 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。Amazon Textract 支持特定的操作、资源和条件键。要了解在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南 中的 IAM JSON 策略元素参考。

操作

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个委托人 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 Action 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

Amazon Textract 中的异步操作需要授予两个操作权限，一个用于开始操作，另一个用于获取操作。此外，如果您使用 Amazon S3 存储桶传递文档，则需要授予账户读取访问权限。

在 Amazon Textract 中，所有政策操作都以：textract:. 例如，要授予某人使用 Amazon Textract 操作的权限。AnalyzeDocument操作，你包括textract:AnalyzeDocument他们的政策中的行动。策略语句必须包含 Action 或 NotAction 元素。Amazon Textract 定义了一组自己的操作，以描述您可以使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示。

```
"Action": [
      "textract:action1",
      "textract:action2"
```

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 Describe 开头的所有操作，请包括以下操作。

```
"Action": "textract:Describe*"
```

有关 Amazon Textract 操作的列表，请参阅Amazon Textract 定义的操作中的IAM 用户指南.

Resources（资源）

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个委托人可以对什么资源执行操作，以及在什么条件下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon Resource Name（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

Amazon Textract 不支持在策略中指定资源 ARN。

条件键

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个委托人可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则 AWS 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅 IAM 用户指南 中的 IAM 策略元素：变量和标签。

AWS支持全局条件键和特定于服务的条件键。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的 AWS 全局条件上下文键。

Amazon Textract 不提供任何服务特定的条件键，但支持使用某些全局条件键。所有列表AWS全局条件键，请参阅AWS全局条件上下文键中的IAM 用户指南.

示例

要查看 Amazon Textract 基于身份的策略的示例，请参阅。Amazon Textract 基于身份的策略示例.

Amazon Textract 基于资源的策略

Amazon Textract 不支持基于资源的策略。

基于 Amazon Textract 的标签的授权

Amazon Textract 不支持标记资源或基于标签的访问控制。

Amazon Textract IAM 角色

IAM 角色是 AWS 账户中具有特定权限的实体。

将临时凭证用于 Amazon Textract

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用 AWS STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

Amazon Textract 支持使用临时凭证。

服务相关角色

服务相关角色允许 AWS 服务访问其它服务中的资源以代表您完成操作。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Amazon Textract 不支持服务相关角色。

> **Note**
>
> 由于 Amazon Textract 不支持服务相关角色，它不支持 AWS 服务委托人。有关服务主体的更多信息，请参阅。AWS 服务委托人中的IAM 用户指南

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其它服务中的资源以代表您完成操作。服务角色显示在您的 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

Amazon Textract 支持服务角色。

Amazon Textract 基于身份的策略示例

默认情况下，IAM 用户和角色没有创建或修改 Amazon Textract 资源的权限。它们还无法使用 AWS 管理控制台、AWS CLI 或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南 中的在 JSON 选项卡上创建策略。

主题

- 策略最佳实践
- 允许用户查看他们自己的权限

- 在 Amazon Textract 中授予对同步操作的访问权限
- 在 Amazon Textract 中授予对异步操作的访问权限

策略最佳实践

基于身份的策略非常强大。它们确定某个人是否可以创建、访问或删除您账户中的 Amazon Textract 资源。这些操作可能会使 AWS 账户 产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用AWS托管策略— 要快速开始使用 Amazon Textract，请使用AWS为您的员工授予他们所需的权限的托管策略。这些策略已在您的账户中提供，并由 AWS 维护和更新。有关更多信息，请参阅 IAM 用户指南中的开始使用 AWS 托管式策略中的权限。
- 授予最低权限 – 创建自定义策略时，仅授予执行任务所需的许可。最开始只授予最低权限，然后根据需要授予其它权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。有关更多信息，请参阅 IAM 用户指南 中的授予最低权限。
- 为敏感操作启用 MFA – 为增强安全性，要求 IAM 用户使用多重身份验证 (MFA) 来访问敏感资源或 API 操作。要了解更多信息，请参阅《IAM 用户指南》中的在 AWS 中使用多重身份验证 (MFA)。
- 使用策略条件来增强安全性 – 在切实可行的范围内，定义基于身份的策略在哪些情况下允许访问资源。例如，您可编写条件来指定请求必须来自允许的 IP 地址范围。您也可以编写条件，以便仅允许指定日期或时间范围内的请求，或者要求使用 SSL 或 MFA。有关更多信息，请参阅 。IAM JSON 策略元素：Condition中的IAM 用户指南.

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
```

```
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

在 Amazon Textract 中授予对同步操作的访问权限

此示例策略向您的 IAM 用户授予对 Amazon Textract 中的同步操作的访问权限AWSaccount.

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument"
            ],
            "Resource": "*"
        }
    ]
```

在 Amazon Textract 中授予对异步操作的访问权限

以下示例策略为您的 IAM 用户提供了AWS账户访问 Amazon Textract 中使用的所有异步操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:StartDocumentTextDetection",
                "textract:StartDocumentAnalysis",
                "textract:GetDocumentTextDetection",
                "textract:GetDocumentAnalysis"
            ],
            "Resource": "*"
        }
    ]
}
```

排查 Amazon Textract 身份和访问的问题

可以使用以下信息，以帮助您诊断和修复在使用 Amazon Textract 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 Amazon Textract 中执行操作
- 我无权执行 iam:PassRole
- 我想要查看我的访问密钥
- 我是管理员并希望允许其他人访问 Amazon Textract
- 我想要允许我的以外的人员AWS用于访问我的 Amazon Textract 资源的账户

我无权在 Amazon Textract 中执行操作

如果 AWS 管理控制台 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

下面的示例错误发生在`mateojackson`IAM 用户尝试运行`DetectDocumentText`在测试图片上但没有`textract:DetectDocumentText`权限。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 textract:DetectDocumentText on resource: textimage.png
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `textract:DetectDocumentText` 操作访问 `textimage.png` 资源。

我无权执行 iam:PassRole

如果您收到错误消息，提示您无权执行 `iam:PassRole` 操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。要求该人员更新您的策略，以允许您将角色传递给 Amazon Textract。

有些 AWS 服务允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的 IAM 用户时，会发生以下示例错误：`marymajor`尝试使用控制台在 Amazon Textract 中执行操作。但是，服务必须具有服务角色所授予的权限才可执行操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，Mary 请求她的管理员来更新其策略，以允许她执行 `iam:PassRole` 操作。

我想要查看我的访问密钥

在创建 IAM 用户访问密钥后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

> **Important**
>
> 请不要向第三方提供访问密钥，即便是为了帮助找到您的规范用户 ID 也不行。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果丢失了您的秘密访问密钥，您必须为 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南中的管理访问密钥。

我是管理员并希望允许其他人访问 Amazon Textract

要允许其他人访问 Amazon Textract，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 AWS。然后，您必须将策略附加到实体，以便在 Amazon Textract 中为他们（它们）授予正确的权限。

要立即开始使用，请参阅 IAM 用户指南中的创建您的第一个 IAM 委派用户和组。

我想要允许我的以外的人员AWS用于访问我的 Amazon Textract 资源的账户

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon Textract 是否支持这些功能，请参阅。Amazon Textract 如何与 IAM 配合使用.
- 要了解如何为您拥有的 AWS 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 AWS 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方AWS 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的 AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

日志记录和监控

要监控 Amazon Textract，请使用 Amazon CloudWatch。本节提供有关如何为 Amazon Textract 设置监控的信息。它还提供了 Amazon Textract 指标的参考内容。

主题

- 监控 Amazon Textract
- Amazon Textract 的 CloudWatch 指标

监控 Amazon Textract

借助 CloudWatch，您可以获取个别 Amazon Textract 操作的指标，也可以获取账户的全局 Amazon Textract 指标。您可以使用指标跟踪基于 Amazon TExtractt 的解决方案的运行状况并设置警报，以便在一个或多个指标超出定义的阈值时通知您。例如，您可以查看已发生的服务器错误数的指标。您还可以查看特定 Amazon Textract 操作成功的次数的指标。要查看指标，您可以使用Amazon CloudWatch，AWS CLI，或者CloudWatch API.

将 CloudWatch 指标用于 Amazon Textract

要使用指标，您必须指定以下信息：

- 指标维度或无维度。维度 是帮助您对某指标进行唯一标识的名称/值对。Amazon Textract 有一个名为的维度运算. 它提供了特定操作的指标。如果您未指定维度，则指标的范围限定为账户内的所有 Amazon Textract 操作。
- 指标名称，如 `UserErrorCount`。

您可 Amazon Textract 用AWS 管理控制台，AWS CLI，或 CloudWatch API。您还可以通过某个 Amazon AWS 软件开发工具包 (SDK) 或 CloudWatch API 工具来使用 CloudWatch API。控制台将根据来自 CloudWatch API 的原始数据显示一系列图表。根据您的需求差异，您可能倾向于使用控制台中显示的图表，也可能倾向于检索自 API 的图表。

下面的列表显示这些指标的一些常见用途。这些是入门建议，并不全面。

如何？	相关指标
如何获知我的应用程序是否已达到每秒最大请求数？	监控 `ThrottledCount` 指标的 `Sum` 统计数据。
如何监控请求错误？	使用 `UserErrorCount` 指标的 `Sum` 统计数据。
如何查找请求总数？	使用 `ResponseTime` 指标的 `SampleCount` 统计数据。这包括任何导致错误的请求。如果您希望仅查看成功的操作调用，请使用 `SuccessfulRequestCount` 指标。

如何？	相关指标
如何监控 Amazon Textract 操作调用的延迟？	使用 `ResponseTime` 指标。

您必须具有适当的 CloudWatch 权限才能使用 CloudWatch 监控 Amazon Textract。有关更多信息，请参阅 Amazon CloudWatch 的身份验证和访问控制。

访问 Amazon Textract 指标

以下示例显示如何通过 CloudWatch 控制台（AWS CLI和 CloudWatch API。

要查看指标（控制台）

1. 访问 https://console.aws.amazon.com/cloudwatch/，打开 CloudWatch 控制台。
2. 选择指标，选择所有指标选项卡，然后选择Amazon Textract.
3. 选择按操作，然后选择指标。

   例如，选择StartDocumentAnalysis衡量异步文档分析启动的次数的指标。
4. 选择日期范围的值。图表中显示指标计数。

查看成功的指标**`StartDocumentAnalysis`**在一段时间内发出的操作调用 (CLI)

- 打开 AWS CLI 并输入以下命令：

  ```
  aws cloudwatch get-metric-statistics \
      --metric-name SuccessfulRequestCount \
      --start-time 2019-02-01T00:00:00Z \
      --period 3600 \
      --end-time 2019-03-01T00:00:00Z \
      --namespace AWS/Textract \
      --dimensions Name=Operation,Value=StartDocumentAnalysis \
      --statistics Sum
  ```

  此示例显示一段时间内成功的 `StartDocumentAnalysis` 操作调用。有关更多信息，请参阅 get-metric-statistics。

访问指标 (CloudWatch API)

- 调用 GetMetricStatistics。有关更多信息，请参阅 。Amazon CloudWatch API 参考.

创建警报

您可以创建在警报改变状态时发送 Amazon Simple Notification Service (Amazon SNS) 消息的 CloudWatch 警报。告警会监控您指定的时间段内的某个指标。它在多个时间段内根据相对于给定阈值的指标值，执行一项或多项操作。操作是一个发送到 Amazon SNS 主题或 Auto Scaling 策略的通知。

警报只会调用操作进行持续的状态变更。CloudWatch 警报不会仅仅因为处于特定状态而调用操作。该状态必须改变并在指定数量的时间段内一直保持。

设置警报 (控制台)

1. 登录AWS 管理控制台并打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择Alarms，然后选择创建警报. 此操作将打开创建警报向导.
3. 选择选择指标。
4. 在所有指标选项卡上，选择Textract.
5. 选择按操作，然后选择指标。

   例如，选择StartDocumentAnalysis以针对最大数目的异步文档分析操作设置警报。
6. 选择 Graphed metrics（已绘制图表指标）选项卡。
7. 对于 Statistic (统计数据)，选择 Sum (总计)。
8. 选择选择指标。
9. 填写 Name 和 Description。对于 Whenever，选择 >=，然后输入您选择的最大值。
10. 如果您希望 CloudWatch 在达到警报状态时向您发送电子邮件，请为无论何时发出警报：，选择状态是警报. 要将警报发送到现有 Amazon SNS 主题，请为发送通知到：中，选择一个现有 SNS 主题。要为新的电子邮件订阅列表设置名称和电子邮件地址，请选择新清单. CloudWatch 会保存列表并将其显示在字段中，以便您可使用它来设置将来的警报。

> **Note**
>
> 如果您使用新清单要创建一个新的 Amazon SNS 主题，必须先验证电子邮件地址，然后目标收件人才能接收通知。Amazon SNS 仅在警报进入警报状态时发送电子邮件。如果在验证电子邮件地址之前此警报状态发生了变化，那么目标收件人不会接收到通知。

11. 选择 Create Alarm（创建告警）。

设置警报 (AWS CLI)

- 打开 AWS CLI 并输入以下命令。更改的值`alarm-actions`参数以引用您之前创建的 Amazon SNS 主题。

```
aws cloudwatch put-metric-alarm \
    --alarm-name StartDocumentAnalysisUserErrors \
    --alarm-description "Alarm when more than 10 StartDocumentAnalysys user errors
 occur within 5 minutes" \
    --metric-name UserErrorCount \
    --namespace AWS/Textract \
    --statistic Sum \
    --period 300 \
    --threshold 10 \
    --comparison-operator GreaterThanThreshold \
    --evaluation-periods 1 \
    --unit Count \
    --dimensions Name=Operation,Value=StartDocumentAnalysis \
    --alarm-actions arn:aws:sns:us-east-1:111111111111:alarmtopic
```

此示例显示如何为 5 分钟内出现 10 个以上的用户错误的情况创建警报，以便通过`StartDocumentAnalysis`. 有关更多信息，请参阅 put-metric-alarm。

设置警报 (CloudWatch API)

- 调用 `PutMetricAlarm`。有关更多信息，请参阅 。Amazon CloudWatch API 参考.

Amazon Textract 的 CloudWatch 指标

本节包含有关 Amazon CloudWatch 指标的信息和运算适用于 Amazon Textract 的维度。

您也可以从 Amazon Textract 控制台查看 Amazon Textract 指标的聚合视图。

Amazon Textract 的 CloudWatch 指标

下表汇总了 Amazon Textract 指标。

指标	描述
SuccessfulRequestCount	成功的请求数。成功请求的响应代码范围为 200 至 299。 单位：计数 有效统计数据：`Sum,Average`
ThrottledCount	受限制请求的数目。当 Amazon Textract 接收的请求数超出为账户设置的每秒事务数限制时，它会限制请求。如果经常超出为账户设置的限制，您可以请求提高限制。要申请增加限制，请参阅AWS 服务限制. 单位：计数 有效统计数据：`Sum,Average`
ResponseTime	Amazon Textract 用来计算响应的时间 (以毫秒为单位)。 单位： 1. `Data Samples` 统计数据的数量 2. `Average` 统计数据的毫秒数 有效统计数据：`Data Samples,Average` **Note** 这些区域有：`ResponseTime` 指标不包括在 Amazon Textract 指标窗格中。
ServerErrorCount	服务器错误的数量。服务器错误的响应代码范围为 500 到 599。 单位：计数

指标	描述
	有效统计数据：Sum,Average
UserErrorCount	用户错误 (参数无效、图像无效、无权限等) 的数目。用户错误的响应代码范围为 400 到 499。 单位：计数 有效统计数据：Sum,Average

Amazon Textract 的 CloudWatch 维度

要检索操作特定的指标，请使用 AWS/Textract 命名空间并提供操作维度。有关维度的更多信息，请参阅维度中的Amazon CloudWatch 用户指南.

使用 记录 Amazon Textract API 调用AWS CloudTrail

Amazon Textract 与AWS CloudTrail，提供用户、角色或者执行操作的记录的服务AWSAmazon Textract 中的服务。CloudTrail 将 Amazon Textract 的所有 API 调用作为事件捕获。这些捕获包括来自 Amazon Textract 控制台的调用和对 Amazon Textract API 操作的代码调用。

如果您创建跟踪记录，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶，包括 Amazon Textract 的事件。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。通过使用 CloudTrail 收集的信息，您可以确定向 Amazon Textract 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅 AWS CloudTrail 用户指南。

CloudTrail 中的 Amazon Textract 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 Amazon Textract 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他活动一起保存在中。AWS中的服务事件事件记录. 您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录您的中事件AWS账户（包括 Amazon Textract 的事件）创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有亚马逊云科技区域。此跟踪在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定

的 Amazon S3 存储桶。此外，您还可以配置其他AWS服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取措施。有关更多信息，请参阅下列内容：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

CloudTrail 记录所有 Amazon Textract 操作，并记录在API 参考. 例如，对 `DetectDocumentText`、`AnalyzeDocument` 和 `GetDocumentText` 操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 AWS Identity and Access Management (IAM) 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它AWS服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

请求参数和未记录的响应字段

出于隐私考虑，某些请求参数和响应字段不会记录，例如，请求图像字节或响应边界框信息。CloudTrail 日志条目中提供了请求参数中提供的 Amazon S3 存储桶名称和文件名。CloudTrail 日志中没有提供有关请求中传递的图像字节的信息。下表显示了每个 Amazon Textract 操作未记录的输入参数和响应参数。

运算	请求参数	响应字段
AnalyzeDocument	`Bytes`	所有
DetectDocumentText	`Bytes`	所有
StartDocumentAnalysis	无	无
GetDocumentAnalysis	无	所有
StartDocumentTextDetection	无	无

运算	请求参数	响应字段
GetDocumentTextDetection	无	所有

了解 Amazon Textract 日志文件条目

跟踪记录是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示一个来自任何源的请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `AnalyzeDocument` 操作。输入的图像字节`document`和分析结果 (`responseElements`) 没有记录。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/janedoe",
        "accountId": "111111111111",
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "userName": "janedoe"
    },
    "eventTime": "2019-04-03T23:56:31Z",
    "eventSource": "textract.amazonaws.com",
    "eventName": "AnalyzeDocument",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "198.51.100.0",
    "userAgent": "",
    "requestParameters": {
        "document": {},
        "featureTypes": [
            "TABLES"
        ]
    },
    "responseElements": null,
    "requestID": "e387676b-d1f0-4ea7-85d6-f5a344052dce",
    "eventID": "c5db79ce-e4ea-4401-8517-784481d559f7",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
```

```
}
```

以下示例显示了用于的 CloudTrail 日志条目。StartDocumentAnalysisoperation. 日志条目包括中的 Amazon S3 存储桶名称和映像文件名documentLocation. 该日志还包括操作响应。

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111111111111:user/janedoe",
                "accountId": "11111111111",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "janedoe"
            },
            "eventTime": "2019-04-04T01:42:24Z",
            "eventSource": "textract.amazonaws.com",
            "eventName": "StartDocumentAnalysis",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "",
            "requestParameters": {
                "documentLocation": {
                    "s3Object": {
                        "bucket": "bucket",
                        "name": "document.png"
                    }
                },
                "featureTypes": [
                    "TABLES"
                ]
            },
            "responseElements": {
                "jobId":
 "f3c718b444fa603d5d625ab967008f4b620d4650c9db8ca1cae01ef7efe51373"
            },
            "requestID": "9ae352e8-9de1-41ad-b77b-85aa348c2e82",
            "eventID": "f741bca0-c3cb-4805-82ea-baf76439deef",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111111111111"
        }
```

```
    ]
}
```

Amazon Textract 的合规性验证

作为多个组成部分，第三方审计员将评估 Amazon Textract 的安全性和合规性AWS合规性计划。其中包括 HIPAA、SOC、ISO 和 PCI。

> **Note**
>
> 如果您正在通过受 PCI DSS 合规性约束的 Textract 服务处理数据，那么您必须联系 AWS Support 并按照提供给您的流程来选择退出账户。

有关列表AWS在特定合规性计划范围内的服务，请参阅合规性计划范围内的 AWS 服务. 有关常规信息，请参阅AWS合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅 。在 AWS Artifact 中下载报告.

您在使用 Amazon Textract 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。AWS提供以下资源来帮助实现合规性：

- 安全性与合规性 Quick Start 指南 - 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- 《设计符合 HIPAA 安全性和合规性要求的架构》白皮书 – 此白皮书介绍公司如何使用AWS创建符合 HIPAA 标准的应用程序。
- AWS合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- AWS Config 开发人员指南中的使用规则评估资源 – 此 AWS Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub CSPM— 这AWS服务提供中安全状态的全面视图。AWS. 安全中心可帮助您检查是否符合安全行业标准和最佳实践。

Amazon Textract 中的恢复能力

AWS全球基础设施围绕AWS区域和可用区构建。AWS区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用

区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 AWS 全球基础设施。

> **Note**
>
> 由于《通用数据保护条例》(GDPR)，不允许跨区域传输数据。

Amazon Textract 中的基础设施安全性

作为托管式服务，Amazon Textract 受AWS中描述的全局网络安全程序Amazon Web Services：安全过程概述白皮书。

你使用AWS发布的 API 调用通过网络访问 Amazon Textract。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

Amazon Textract 中的配置和漏洞分析

配置和 IT 控制是 AWS 和您（我们的客户）之间的共同责任。有关更多信息，请参阅AWS责任共担模型。

Amazon Textract 和接口 VPC 终端节点 (AWS PrivateLink)

您可以通过创建 VPC 和 Amazon Textract 之间的私有连接。接口 VPC 终端节点. 接口终端节点由以下公司提供提供支持AWS PrivateLink，该技术使您可以私下访问 Amazon Textract API，而无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。VPC 中的实例即使没有公有 IP 地址也可与 Amazon Textract API 进行通信。VPC 和 Amazon Textract 之间的流量不会脱离 AWS 网络。

每个接口终端节点均由子网中的一个或多个弹性网络接口表示。

有关更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 终端节点 (Amazon PrivateLink)。

Amazon Textract VPC 终端节点的注意事项

在为 Amazon Textract 设置接口 VPC 终端节点之前，请务必查看接口终端节点属性和限制中的Amazon VPC User Guide.

Amazon Textract 支持从 VPC 调用它的所有 API 操作。

为 Amazon Textract 创建接口 VPC 终端节点

您可以使 Amazon Textract VPC 控制台或AWS Command Line Interface(AWS CLI)。有关更多信息，请参阅 Amazon VPC 用户指南.中的创建接口端点

使用以下服务名称为 Amazon Textract 创建 VPC 终端节点：

- com.amazonaws.##.textract-用于为大多数 Amazon Textract 操作创建终端节点。
- com.amazonaws.##.textract-fips-为创建符合联邦信息处理标准 (FIPS) 出版 Amazon Textract 140-2 美国政府标准的终端节点。

如果为终端节点启用私有 DNS，则可以使用针对区域的默认 DNS 名称向 Amazon Textract 发出 API 请求，例如：`textract.us-east-1.amazonaws.com`.

有关更多信息，请参阅 Amazon VPC 用户指南中的通过接口端点访问服务。

为 Amazon Textract 创建 VPC 终端节点策略

您可以为 VPC 终端节点附加控制对 Amazon Textract 的访问的终端节点策略。该策略指定以下信息：

- 可执行操作的委托人。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅 Amazon VPC 用户指南中的使用 VPC 终端节点控制对服务的访问。

例如：适用于 Amazon Textract 操作的 VPC 终端节点策略

下面是用于 Amazon Textract 的终端节点策略示例。当附加到终端节点时，此策略会向所有委托人授予对列出的针对所有资源的 Amazon Textract 操作的访问权限。

此示例策略仅允许访问操作。`DetectDocumentText`和`AnalyzeDocument`. 用户仍可从 VPC 终端节点外部调用 Amazon Textract 操作。

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "textract:DetectDocumentText",
            "textract:AnalyzeDocument",
         ],
         "Resource":"*"
      }
   ]
}
```

API 引用

本节提供 Amazon Textract API 操作的文档。

主题

- 操作
- 数据类型

操作

支持以下操作：

- AnalyzeDocument
- AnalyzeExpense
- AnalyzeID
- DetectDocumentText
- GetDocumentAnalysis
- GetDocumentTextDetection
- GetExpenseAnalysis
- StartDocumentAnalysis
- StartDocumentTextDetection
- StartExpenseAnalysis

AnalyzeDocument

分析输入文档以了解检测到的项目间的关系。

返回的信息类型如下：

- 表单数据（键值对）。两个相关信息会分两个返回Block对象，每种类型KEY_VALUE_SET键：KeyBlock对象和一个 VALUEBlock对象。例如，名称：安娜·席尔瓦包含密钥和值。名称：是关键。安娜·席尔瓦是值。
- 表格和表格单元格数据。一张桌子Block对象包含有关检测到的表的信息。一个单元格Block对象将为表中的每个单元格返回。
- 文本的行和单词。一行Block对象包含一个或多个 WORDBlock对象。返回文档中检测到的所有行和单词（包括与FeatureTypes)。

可以在表单数据和表格中检测到选择元素，例如复选框和选项按钮（单选按钮）。一个选择 _ 元素Block对象包含有关选择元素的信息，包括选择状态。

您可以通过指定FeatureTypes列表。

输出将在列表中返回Block对象。

AnalyzeDocument是一个同步操作。要异步分析文档，请使用StartDocumentAnalysis.

有关更多信息，请参阅 。文档文本分析.

请求语法

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "FeatureTypes": [ "string" ],
   "HumanLoopConfig": {
      "DataAttributes": {
         "ContentClassifiers": [ "string" ]
```

```
        },
        "FlowDefinitionArn": "string",
        "HumanLoopName": "string"
    }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

Document

作为 base64 编码的字节或 Amazon S3 对象的输入文档。如果您使用 AWS CLI 调用 Amazon Textract 操作，则无法传递图像字节。文档必须是 JPEG、PNG、PDF 或 TIFF 格式的图像。

如果您使用 AWS 开发工具包调用 Amazon Textract，则可能不需要对使用`Bytes`字段中返回的子位置类型。

类型：Document 对象

必填项：是

FeatureTypes

要执行的分析类型的列表。将 TABLES 添加到列表以返回有关在输入文档中检测到的表的信息。添加 FORMS 以返回检测到的表单数据。要执行这两种类型的分析，请将 TABLES 和 FORMS 添加到`FeatureTypes`. 在文档中检测到的所有行和单词都包含在响应中（包括与值无关的文本`FeatureTypes`)。

类型: 字符串数组

有效值: `TABLES | FORMS`

必填项：是

HumanLoopConfig

设置用于分析文档的循环中人员工作流程的配置。

类型：HumanLoopConfig 对象

必填项：否

响应语法

```
{
   "AnalyzeDocumentModelVersion": "string",
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DocumentMetadata": {
      "Pages": number
   },
```

```
   "HumanLoopActivationOutput": {
      "HumanLoopActivationConditionsEvaluationResults": "string",
      "HumanLoopActivationReasons": [ "string" ],
      "HumanLoopArn": "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

AnalyzeDocumentModelVersion

用于分析文档的模型的版本。

类型: 字符串

Blocks

检测到和分析的物品AnalyzeDocument.

类型: 的数组数组Block对象

DocumentMetadata

有关分析文档的元数据。一个例子是页数。

类型：DocumentMetadata 对象

HumanLoopActivationOutput

显示循环中的人类评估结果。

类型：HumanLoopActivationOutput 对象

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

BadDocumentException

Amazon Textract 无法阅读文档。有关 Amazon Textract 中文档限制的更多信息，请参阅Amazon Textract 中的硬性限制.

HTTP 状态代码：400

DocumentTooLargeException

无法处理该文档，因为它太大。同步操作的最大文档大小为 10 MB。对于 PDF 文件，异步操作的最大文档大小为 500 MB。

HTTP 状态代码：400

HumanLoopQuotaExceededException

表示您已超出可用循环工作流程中的最大活跃人员数量。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，`InvalidParameterException`如果两者都不会发生异常`S3Object`要么`Bytes`值在`Document`请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

UnsupportedDocumentException

不支持输入文档的格式。用于操作的文档可以是 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 状态代码：400

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

AnalyzeExpense

AnalyzeExpense同步分析输入文档以了解文本之间的财务相关关系。

信息将返回为ExpenseDocuments并分开如下。

- LineItemGroups-包含的数据集LineItems它们存储有关文本行的信息，例如购买的物品及收据上的价格。
- SummaryFields-收据包含所有其他信息，例如标题信息或供应商名称。

请求语法

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

Document

输入文档，可以是字节或作为 S3 对象。

您可以使用Bytes财产。例如，您应使用Bytes属性来传递从本地文件系统加载的文档。使用Bytes属性必须采用 base64 编码。如果您使用 AWS 开发工具包调用 Amazon Textract API 操作，则代码可能不需要对文档文件字节进行编码。

您可以使用存储在 S3 存储桶中的图像传递给 Amazon Textract API 操作。S3Object财产。存储在 S3 存储桶中的文档不需要 base64 编码。

包含 S3 对象的 S3 存储桶的 AWS 区域必须与您用于 Amazon Textract 操作的 AWS 区域匹配。

如果您使用 AWS CLI 调用 Amazon Textract 操作，则不支持使用字节属性传递图像字节。您必须先将文档上传到 Amazon S3 存储桶，然后使用 S3Oject 属性调用操作。

要使 Amazon Textract 处理 S3 对象，用户必须具有访问 S3 对象的权限。

类型：Document 对象

必填项：是

响应语法

```
{
   "DocumentMetadata": {
      "Pages": number
   },
   "ExpenseDocuments": [
      {
         "ExpenseIndex": number,
         "LineItemGroups": [
            {
               "LineItemGroupIndex": number,
               "LineItems": [
                  {
                     "LineItemExpenseFields": [
                        {
                           "LabelDetection": {
                              "Confidence": number,
                              "Geometry": {
                                 "BoundingBox": {
                                    "Height": number,
                                    "Left": number,
                                    "Top": number,
                                    "Width": number
                                 },
                                 "Polygon": [
                                    {
                                       "X": number,
                                       "Y": number
                                    }
                                 ]
                              },
                              "Text": "string"
                           },
```

```
                        "PageNumber": number,
                        "Type": {
                           "Confidence": number,
                           "Text": "string"
                        },
                        "ValueDetection": {
                           "Confidence": number,
                           "Geometry": {
                              "BoundingBox": {
                                 "Height": number,
                                 "Left": number,
                                 "Top": number,
                                 "Width": number
                              },
                              "Polygon": [
                                 {
                                    "X": number,
                                    "Y": number
                                 }
                              ]
                           },
                           "Text": "string"
                        }
                     }
                  ]
               }
            ]
         }
      ],
      "SummaryFields": [
         {
            "LabelDetection": {
               "Confidence": number,
               "Geometry": {
                  "BoundingBox": {
                     "Height": number,
                     "Left": number,
                     "Top": number,
                     "Width": number
                  },
                  "Polygon": [
                     {
                        "X": number,
                        "Y": number
```

```
                        }
                     ]
                  },
                  "Text": "string"
               },
               "PageNumber": number,
               "Type": {
                  "Confidence": number,
                  "Text": "string"
               },
               "ValueDetection": {
                  "Confidence": number,
                  "Geometry": {
                     "BoundingBox": {
                        "Height": number,
                        "Left": number,
                        "Top": number,
                        "Width": number
                     },
                     "Polygon": [
                        {
                           "X": number,
                           "Y": number
                        }
                     ]
                  },
                  "Text": "string"
               }
            }
         ]
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

DocumentMetadata

有关输入文档的信息。

类型：DocumentMetadata 对象

ExpenseDocuments

Amazon Textract 检测到的费用。

类型: 数组的ExpenseDocument对象

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

BadDocumentException

Amazon Textract 无法阅读文档。有关 Amazon Textract 中文档限制的更多信息，请参阅Amazon Textract 中的硬性限制.

HTTP 状态代码：400

DocumentTooLargeException

无法处理该文档，因为它太大。同步操作的最大文档大小为 10 MB。对于 PDF 文件，异步操作的最大文档大小为 500 MB。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，`InvalidParameterException`如果两者都没有`S3Object`要么`Bytes`值在`Document`请求参数。请先验证您的参数，然后再次调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，请配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

UnsupportedDocumentException

输入文档的格式不受支持。操作文档可以是 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 状态代码：400

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

AnalyzeID

分析身份证件以获取相关信息。此信息被提取并返回为IdentityDocumentFields，它记录了提取的文本的标准化字段和值。与其他 Amazon Textract 操作不同，AnalyzeID不返回任何几何数据。

请求语法

```
{
   "DocumentPages": [
      {
         "Bytes": blob,
         "S3Object": {
            "Bucket": "string",
            "Name": "string",
            "Version": "string"
         }
      }
   ]
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

DocumentPages

文档正在传递给 AnalyZeID。

类型: 数组数组Document对象

数组成员：最少 1 项。最多 2 项。

必填项：是

响应语法

```
{
   "AnalyzeIDModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   },
```

```
   "IdentityDocuments": [
      {
         "DocumentIndex": number,
         "IdentityDocumentFields": [
            {
               "Type": {
                  "Confidence": number,
                  "NormalizedValue": {
                     "Value": "string",
                     "ValueType": "string"
                  },
                  "Text": "string"
               },
               "ValueDetection": {
                  "Confidence": number,
                  "NormalizedValue": {
                     "Value": "string",
                     "ValueType": "string"
                  },
                  "Text": "string"
               }
            }
         ]
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

AnalyzeIDModelVersion

用于处理文档的 AnalyzeIdentity API 的版本。

类型: 字符串

DocumentMetadata

有关输入文档的信息。

类型：DocumentMetadata 对象

IdentityDocuments

AnalyzeID 处理的文档清单。包括一个数字，表示它们在列表中的位置以及文档的响应结构。

类型: 数组数组IdentityDocument对象

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

BadDocumentException

Amazon Textract 无法阅读文档。有关 Amazon Textract 中文档限制的更多信息，请参阅Amazon Textract 中的硬性限制.

HTTP 状态代码：400

DocumentTooLargeException

无法处理该文档，因为它太大。同步操作的最大文档大小为 10 MB。对于 PDF 文件，异步操作的最大文档大小为 500 MB。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，`InvalidParameterException`如果两者都不会发生异常`S3Object`要么`Bytes`值在`Document`请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

UnsupportedDocumentException

不支持输入文档的格式。操作文档可以是 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 状态代码：400

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

DetectDocumentText

检测输入文档中的文本。Amazon Textract 可以检测文本行和构成一行文本的单词。输入文档必须是 JPEG、PNG、PDF 或 TIFF 格式的图像。DetectDocumentText在数组中返回检测到的文本Block对象。

每个文档页面都有关联Block的类型 PAGE。每页Block对象是 LINE 的父Block表示页面上检测到的文本行的对象。一行Block对象是构成该行的每个单词的父项。单词的表示为BlockWORD 类型的对象。

DetectDocumentText是一个同步操作。要异步分析文档，请使用StartDocumentTextDetection.

有关更多信息，请参阅 。文本检测.

请求语法

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

Document

作为 base64 编码的字节或 Amazon S3 对象的输入文档。如果您使用 AWS CLI 调用 Amazon Textract 操作，则无法传递图像字节。文档必须为 JPEG 或 PNG 格式的图像。

如果您使用 AWS 开发工具包调用 Amazon Textract，则可能不需要对使用Bytes字段中返回的子位置类型。

类型：Document 对象

是必需的：是

响应语法

```
{
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DetectDocumentTextModelVersion": "string",
   "DocumentMetadata": {
```

```
        "Pages": number
    }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

Blocks

数组Block包含文档中检测到的文本的对象。

类型: 数组Block对象

DetectDocumentTextModelVersion

类型: 字符串

DocumentMetadata

关于文档的元数据。它包含文档中检测到的页数。

类型：DocumentMetadata 对象

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

BadDocumentException

Amazon Textract 无法阅读文档。有关 Amazon Textract 中文档限制的更多信息，请参阅Amazon Textract 中的硬性限制.

HTTP 状态代码：400

DocumentTooLargeException

无法处理该文档，因为它太大。同步操作的最大文档大小为 10 MB。对于 PDF 文件，异步操作的最大文档大小为 500 MB。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，`InvalidParameterException`如果两者都不会发生异常`S3Object`要么`Bytes`值在`Document`请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

UnsupportedDocumentException

不支持输入文档的格式。用于操作的文档可以是 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 状态代码：400

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface

- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

GetDocumentAnalysis

获取分析文档中文本的 Amazon Textract 异步操作的结果。

通过调用来开始异步文本分析StartDocumentAnalysis，它返回作业标识符 (JobId)。文本分析操作完成后，Amazon Textract 将完成状态发布到亚马逊 Simple Notification Service (Amazon SNS) 主题，该主题在首次调用时注册StartDocumentAnalysis. 要获得文本检测操作的结果，请首先检查发布到 Amazon SNS 主题的状态值是否为SUCCEEDED. 如果是的话，打电话GetDocumentAnalysis，然后传递作业标识符（JobId) 从最初的电话到StartDocumentAnalysis.

GetDocumentAnalysis返回一个数组Block对象。返回以下类型的信息：

- 表单数据（键值对）。两个相关信息返回Block对象，每种类型KEY_VALUE_SET: KeyBlock对象和一个 VALUEBlock对象。例如，名称：安娜·席尔瓦包含密钥和值。名称：是关键。安娜·席尔瓦是值。
- 表格和表格单元格数据。一张桌子Block对象包含有关检测到的表的信息。一个单元格Block对象将为表中的每个单元格返回。
- 文本的行和单词。一行Block对象包含一个或多个 WORDBlock对象。返回文档中检测到的所有行和单词（包括与StartDocumentAnalysis FeatureTypes输入参数)。

可以在表单数据和表格中检测到选择元素，例如复选框和选项按钮（单选按钮）。一个选择 _ 元素Block对象包含有关选择元素的信息，包括选择状态。

使用MaxResults参数以限制返回的块数。如果结果超过中指定的结果MaxResults，的价值NextToken在操作响应中包含一个用于获取下一组结果的分页令牌。要获取下一页结果，请致电GetDocumentAnalysis，然后填充NextToken具有从上一次调用返回的令牌值的请求参数GetDocumentAnalysis.

有关更多信息，请参阅 。文档文本分析.

请求语法

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

JobId

文本检测任务的唯一标识符。这些区域有：JobId从返回StartDocumentAnalysis. 一个JobId值仅在 7 天内有效。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

：必需 是

MaxResults

每次分页呼叫返回的最大结果数。您可以指定的最大值是 1,000。如果指定的值大于 1,000，则返回最多 1000 个结果。默认值是 1,000。

类型: 整数

有效范围：最小值为 1。

：必需 否

NextToken

如果之前的响应不完整（因为需要检索更多块），Amazon Textract 将在响应中返回分页令牌。您可以使用此分页令牌来检索下一组块块。

类型: 字符串

长度约束：最小长度为 1。长度上限为 255。

模式：.*\S.*

：必需 否

响应语法

```
{
   "AnalyzeDocumentModelVersion": "string",
```

```
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DocumentMetadata": {
      "Pages": number
   },
   "JobStatus": "string",
   "NextToken": "string",
   "StatusMessage": "string",
   "Warnings": [
      {
```

```
        "ErrorCode": "string",
        "Pages": [ number ]
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

AnalyzeDocumentModelVersion

类型: 字符串

Blocks

文本分析操作的结果。

类型: 的数组Block对象

DocumentMetadata

有关 Amazon Textract 处理的文档的信息。DocumentMetadata在 Amazon Textract 视频操作的分页响应的每一页中返回。

类型：DocumentMetadata 对象

JobStatus

文本检测任务的当前状态。

类型: 字符串

有效值: IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

如果响应被截断，Amazon Textract 将返回此令牌。您可以在后续请求中使用此令牌来检索下一组文本检测结果。

类型: 字符串

长度约束：最小长度为 1。长度上限为 255。

模式：.*\S.*

StatusMessage

如果无法完成检测任务，则返回。包含发生什么错误的解释。

类型: 字符串

Warnings

文档分析操作期间发生的警告列表。

类型: 的数组Warning对象

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidJobIdException

向传递了无效的作业标识符GetDocumentAnalysis或者去GetDocumentAnalysis.

HTTP 状态代码：400

InvalidKMSKeyException

表示输入的 KMS 密钥没有解密权限，或者 KMS 密钥输入错误。

HTTP 状态代码：400

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，InvalidParameterException如果两者都不会发生异常S3Object要么Bytes值在Document请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，请配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

GetDocumentTextDetection

获取用于检测文档中文本的 Amazon Textract 异步操作的结果。Amazon Textract 可以检测文本行和构成一行文本的单词。

通过调用来开始异步文本检测StartDocumentTextDetection，它返回作业标识符 (JobId)。当文本检测操作完成后，Amazon Textract 向亚马逊 Simple Notification Service (Amazon SNS) 主题发布完成状态，该主题已在首次调用时注册StartDocumentTextDetection. 要获得文本检测操作的结果，请首先检查发布到 Amazon SNS 主题的状态值是否为SUCCEEDED. 如果是的话，打电话GetDocumentTextDetection，然后传递作业标识符 (JobId) 从最初的电话到StartDocumentTextDetection.

GetDocumentTextDetection返回一个数组Block对象。

每个文档页面都有关联Block的类型 PAGE。每个页面Block对象是 LINE 的父Block表示页面上检测到的文本行的对象。一行Block对象是构成该行的每个单词的父项。单词的表示为BlockWORD 类型的对象。

使用 MaxResults 参数限制返回的数据块数。如果结果超过中指定的结果MaxResults，的价值NextToken在操作响应中包含用于获取下一组结果的分页令牌。要获取下一页结果，请致电GetDocumentTextDetection，然后填充NextToken具有从上一次调用返回的令牌值的请求参数GetDocumentTextDetection.

有关更多信息，请参阅 。文本检测.

请求语法

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

JobId

文本检测作业的唯一标识符。这些区域有：JobId从返回的StartDocumentTextDetection. 一个JobId该值仅在 7 天内有效。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

：必需 是

MaxResults

每个分页呼叫返回的最大结果数。您可以指定的最大值是 1,000。如果指定的值大于 1,000，则返回最多 1000 个结果。默认值是 1,000。

类型: 整数

有效范围：最小值为 1。

：必需 否

NextToken

如果之前的响应不完整（因为需要检索更多块），Amazon Textract 将在响应中返回分页令牌。您可以使用此分页令牌来检索下一组区块。

类型: 字符串

长度约束：最小长度为 1。长度上限为 255。

模式：.*\S.*

：必需 否

响应语法

```
{
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
```

```
                "Left": number,
                "Top": number,
                "Width": number
             },
             "Polygon": [
                {
                   "X": number,
                   "Y": number
                }
             ]
          },
          "Id": "string",
          "Page": number,
          "Relationships": [
             {
                "Ids": [ "string" ],
                "Type": "string"
             }
          ],
          "RowIndex": number,
          "RowSpan": number,
          "SelectionStatus": "string",
          "Text": "string",
          "TextType": "string"
       }
    ],
    "DetectDocumentTextModelVersion": "string",
    "DocumentMetadata": {
       "Pages": number
    },
    "JobStatus": "string",
    "NextToken": "string",
    "StatusMessage": "string",
    "Warnings": [
       {
          "ErrorCode": "string",
          "Pages": [ number ]
       }
    ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

Blocks

文本检测操作的结果。

类型: 数组的数组Block对象

DetectDocumentTextModelVersion

类型: 字符串

DocumentMetadata

有关 Amazon Textract 处理的文档的信息。DocumentMetadata在 Amazon Textract 视频操作的分页响应的每一页中返回。

类型：DocumentMetadata 对象

JobStatus

文本检测作业的当前状态。

类型: 字符串

有效值: IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

如果响应被截断，Amazon Textract 将返回此令牌。您可以在后续请求中使用此令牌来检索下一组文本检测结果。

类型: 字符串

长度约束：最小长度为 1。长度上限为 255。

模式：.*\S.*

StatusMessage

如果无法完成检测作业，则返回该函数。包含发生什么错误的解释。

类型: 字符串

Warnings

文档的文本检测操作期间发生的警告列表。

类型: 数组的数组Warning对象

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidJobIdException

向传递了无效的作业标识符GetDocumentAnalysis或者去GetDocumentAnalysis.

HTTP 状态代码：400

InvalidKMSKeyException

表示输入的 KMS 密钥没有解密权限，或者 KMS 密钥输入错误。

HTTP 状态代码：400

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，`InvalidParameterException`如果两者都不会发生异常`S3Object`要么`Bytes`值在`Document`请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

GetExpenseAnalysis

获取分析发票和收据的 Amazon Textract 异步操作的结果。Amazon Textract 从输入的发票和收据中查找联系信息、购买的商品和供应商名称。

通过调用开始异步发票/收据分析StartExpenseAnalysis，它返回一个作业标识符 (JobId)。Amazon Textract 在完成发票/收据分析后，将完成状态发布到 Amazon Simple Notification Service (Amazon SNS) 主题。必须在初次调用时注册此主题StartExpenseAnalysis. 要获取发票/收据分析操作的结果，请先确保向 Amazon SNS 主题发布的状态值为SUCCEEDED. 如果是的话，打电话GetExpenseAnalysis，然后传递作业标识符 (JobId) 从最初的电话到StartExpenseAnalysis.

使用 MaxResults 参数限制返回的块数量。如果结果超过中指定的结果MaxResults，的价值NextToken在操作响应中包含用于获取下一组结果的分页令牌。要获取下一页结果，请致电GetExpenseAnalysis，然后填充NextToken具有从上一次调用返回的令牌值的请求参数GetExpenseAnalysis.

有关更多信息，请参阅 。分析发票和收据.

请求语法

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

JobId

文本检测作业的唯一标识符。这些区域有：JobId从返回StartExpenseAnalysis. 一个JobId值仅在 7 天内有效。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

必填项：是

MaxResults

每个分页呼叫返回的最大结果数量。您可以指定的最大值是 20。如果您指定的值大于 20，则最多返回 20 个结果。默认值为 20。

类型: 整数

有效范围：最小值为 1。

必填项：否

NextToken

如果之前的响应不完整（因为需要检索更多块），Amazon Textract 将在响应中返回分页令牌。您可以使用此分页令牌检索下一组块块。

类型: 字符串

长度约束：最小长度为 1。长度上限为 255。

模式：`.*\S.*`

必填项：否

响应语法

```
{
   "AnalyzeExpenseModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   },
   "ExpenseDocuments": [
      {
         "ExpenseIndex": number,
         "LineItemGroups": [
            {
               "LineItemGroupIndex": number,
               "LineItems": [
                  {
                     "LineItemExpenseFields": [
                        {
                           "LabelDetection": {
                              "Confidence": number,
```

```
                        "Geometry": {
                           "BoundingBox": {
                              "Height": number,
                              "Left": number,
                              "Top": number,
                              "Width": number
                           },
                           "Polygon": [
                              {
                                 "X": number,
                                 "Y": number
                              }
                           ]
                        },
                        "Text": "string"
                     },
                     "PageNumber": number,
                     "Type": {
                        "Confidence": number,
                        "Text": "string"
                     },
                     "ValueDetection": {
                        "Confidence": number,
                        "Geometry": {
                           "BoundingBox": {
                              "Height": number,
                              "Left": number,
                              "Top": number,
                              "Width": number
                           },
                           "Polygon": [
                              {
                                 "X": number,
                                 "Y": number
                              }
                           ]
                        },
                        "Text": "string"
                     }
                  }
               ]
            }
         ]
      }
```

```
        ],
        "SummaryFields": [
           {
              "LabelDetection": {
                 "Confidence": number,
                 "Geometry": {
                    "BoundingBox": {
                       "Height": number,
                       "Left": number,
                       "Top": number,
                       "Width": number
                    },
                    "Polygon": [
                       {
                          "X": number,
                          "Y": number
                       }
                    ]
                 },
                 "Text": "string"
              },
              "PageNumber": number,
              "Type": {
                 "Confidence": number,
                 "Text": "string"
              },
              "ValueDetection": {
                 "Confidence": number,
                 "Geometry": {
                    "BoundingBox": {
                       "Height": number,
                       "Left": number,
                       "Top": number,
                       "Width": number
                    },
                    "Polygon": [
                       {
                          "X": number,
                          "Y": number
                       }
                    ]
                 },
                 "Text": "string"
              }
```

```
            }
         ]
      }
   ],
   "JobStatus": "string",
   "NextToken": "string",
   "StatusMessage": "string",
   "Warnings": [
      {
         "ErrorCode": "string",
         "Pages": [ number ]
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

AnalyzeExpenseModelVersion

分析费用的当前模型版本。

类型: 字符串

DocumentMetadata

有关 Amazon Textract 处理的文档的信息。DocumentMetadata在 Amazon Textract 操作的分页回复的每一页中返回。

类型：DocumentMetadata 对象

ExpenseDocuments

Amazon Textract 检测到的费用。

类型: 数组ExpenseDocument对象

JobStatus

文本检测作业的当前状态。

类型: 字符串

有效值: IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

如果响应被截断，Amazon Textract 将返回此令牌。您可以在后续请求中使用此令牌检索下一组文本检测结果。

类型: 字符串

长度约束：最小长度为 1。长度上限为 255。

模式：.*\S.*

StatusMessage

如无法完成检测作业，则返回该项。包含发生什么错误的解释。

类型: 字符串

Warnings

文档的文本检测操作期间发生的警告列表。

类型: 数组Warning对象

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidJobIdException

向传递了无效的作业标识符GetDocumentAnalysis或者去GetDocumentAnalysis.

HTTP 状态代码：400

InvalidKMSKeyException

表示输入的 KMS 密钥没有解密权限，或者 KMS 密钥输入错误。

HTTP 状态代码：400

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，InvalidParameterException如果两者都不会发生异常S3Object要么Bytes值在Document请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 JavaScript 的AWS开发工具包

- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

StartDocumentAnalysis

启动对输入文档的异步分析，以了解检测到的项目（例如键值对、表和选择元素）之间的关系。

StartDocumentAnalysis可以分析 JPEG、PNG、TIFF 和 PDF 格式的文档中的文本。这些文档存储在 Amazon S3 存储桶中。使用DocumentLocation指定文档的存储桶名称和文件名。

StartDocumentAnalysis返回作业标识符 (JobId) 您用来获取操作结果。当文本分析完成后，Amazon Textract 将完成状态发布到您在中指定的 Amazon Simple Notification Service (Amazon SNS) 主题。NotificationChannel. 要获得文本分析操作的结果，请首先检查发布到 Amazon SNS 主题的状态值是否为SUCCEEDED. 如果是的话，打电话GetDocumentAnalysis，然后传递作业标识符（JobId) 从最初的电话到StartDocumentAnalysis.

有关更多信息，请参阅 。文档文本分析.

请求语法

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "FeatureTypes": [ "string" ],
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ClientRequestToken

用于标识启动请求的幂等令牌。如果你对多个使用同一个令牌StartDocumentAnalysis请求，同样JobId返回。使用ClientRequestToken以防止同样的工作不止一次被意外启动。有关更多信息，请参阅 。调用 Amazon Textract 异步操作.

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

：必需 否

DocumentLocation

要处理的文档的位置。

类型：DocumentLocation 对象

：必需 是

FeatureTypes

要执行的分析类型的列表。将 TABLES 添加到列表以返回有关输入文档中检测到的表的信息。添加 FORMS 以返回检测到的表单数据。要执行这两种类型的分析，请将 TABLES 和 FORMS 添加到FeatureTypes. 在文档中检测到的所有行和单词都包含在响应中（包括与值无关的文本）FeatureTypes)。

类型: 字符串数组

有效值: TABLES | FORMS

：必需 是

JobTag

您指定的标识符包含在发布到 Amazon SNS 主题的完成通知中。例如，您可以使用JobTag以确定完成通知对应的文件类型（例如纳税表或收据）。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：[a-zA-Z0-9_.\-:]+

：必需 否

KMSKeyId

用于加密推理结果的 KMS 密钥。这可以是密钥 ID 或密钥别名格式。提供 KMS 密钥后，KMS 密钥将用于对客户存储桶中的对象进行服务器端加密。如果未启用此参数，结果将使用 SSE-S3 加密服务器端。

类型: 字符串

长度约束：最小长度为 1。长度上限为 2048。

模式：`^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

：必需 否

NotificationChannel

您希望 Amazon Textract 将操作的完成状态发布到的 Amazon SNS 主题 ARN。

类型：NotificationChannel 对象

：必需 否

OutputConfig

设置输出是否转到客户定义的存储桶。默认情况下，Amazon Textract 将在内部保存结果以供 getDocumentAnalysis 操作访问。

类型：OutputConfig 对象

：必需 否

响应语法

```
{
   "JobId": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

JobId

文档文本检测作业的标识符。使用JobId在接下来的电话中确定任务GetDocumentAnalysis. 一个JobId值仅在 7 天内有效。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

BadDocumentException

Amazon Textract 无法阅读文档。有关 Amazon Textract 中文档限制的更多信息，请参阅Amazon Textract 中的硬性限制.

HTTP 状态代码：400

DocumentTooLargeException

无法处理该文档，因为它太大。同步操作的最大文档大小为 10 MB。对于 PDF 文件，异步操作的最大文档大小为 500 MB。

HTTP 状态代码：400

IdempotentParameterMismatchException

一个ClientRequestToken输入参数与操作一起重用，但至少有一个其他输入参数与先前对该操作的调用中的参数不同。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidKMSKeyException

表示输入的 KMS 密钥没有解密权限，或者 KMS 密钥输入错误。

HTTP 状态代码：400

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，`InvalidParameterException`如果两者都不会发生异常`S3Object`要么`Bytes`值在`Document`请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，请配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

LimitExceededException

超出了 Amazon Textract 服务限制。例如，如果同时启动太多异步作业，则调用启动操作（`StartDocumentTextDetection`例如）引发 LimitExceededException 异常 (HTTP 状态代码：400)，直到并发运行的任务数量低于 Amazon Textract 服务限制。

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

UnsupportedDocumentException

不支持输入文档的格式。操作文档可以是 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 状态代码：400

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

StartDocumentTextDetection

启动对文档中的文本开始异步检测。Amazon Textract 可以检测文本行和构成一行文本的单词。

`StartDocumentTextDetection`可以分析 JPEG、PNG、TIFF 和 PDF 格式的文档中的文本。这些文档存储在 Amazon S3 存储桶中。使用DocumentLocation以指定文档的存储桶名称和文件名。

`StartTextDetection`返回作业标识符 (`JobId`) 您用来获取操作结果。当文本检测完成后，Amazon Textract 将完成状态发布到您在中指定的 Amazon Simple Notification Service (Amazon SNS) 主题。`NotificationChannel`. 要获得文本检测操作的结果，请首先检查发布到 Amazon SNS 主题的状态值是否为`SUCCEEDED`. 如果是的话，打电话GetDocumentTextDetection，然后传递作业标识符 (`JobId`) 从最初的电话到`StartDocumentTextDetection`.

有关更多信息，请参阅 。文档文本检测.

请求语法

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ClientRequestToken

用于标识启动请求的幂等令牌。如果你对多个使用同一个令牌StartDocumentTextDetection请求，同样JobId返回。使用ClientRequestToken以防止同样的工作不止一次被意外启动。有关更多信息，请参阅 。调用 Amazon Textract 异步操作.

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

：必需 否

DocumentLocation

要处理的文档的位置。

类型：DocumentLocation 对象

：必需 是

JobTag

您指定的标识符包含在发布到 Amazon SNS 主题的完成通知中。例如，您可以使用JobTag以确定完成通知对应的文件类型（例如纳税表或收据）。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：[a-zA-Z0-9_.\-:]+

：必需 否

KMSKeyId

用于加密推断结果的 KMS 密钥。这可以是密钥 ID 或密钥别名格式。提供 KMS 密钥后，KMS 密钥将用于对客户存储桶中的对象进行服务器端加密。如果未启用此参数，结果将使用 SSE-S3 加密服务器端。

类型: 字符串

长度约束：最小长度为 1。长度上限为 2048。

模式：^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$

：必需 否

NotificationChannel

您希望 Amazon Textract 将操作的完成状态发布到的 Amazon SNS 主题 ARN。

类型：NotificationChannel 对象

：必需 否

OutputConfig

设置输出是否转到客户定义的存储桶。默认情况下，Amazon Textract 将在内部保存结果，以便通过 getDocumentTextTectIncect 操作访问。

类型：OutputConfig 对象

：必需 否

响应语法

```
{
   "JobId": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

JobId

文档的文本检测作业的标识符。使用JobId在接下来的电话中确定任务GetDocumentTextDetection. 一个JobId值仅在 7 天内有效。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

BadDocumentException

Amazon Textract 无法阅读文档。有关 Amazon Textract 中文档限制的更多信息，请参阅Amazon Textract 中的硬性限制.

HTTP 状态代码：400

DocumentTooLargeException

无法处理该文档，因为它太大。同步操作的最大文档大小为 10 MB。对于 PDF 文件，异步操作的最大文档大小为 500 MB。

HTTP 状态代码：400

IdempotentParameterMismatchException

一个`ClientRequestToken`输入参数与操作一起重用，但至少有一个其他输入参数与先前对该操作的调用中的参数不同。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidKMSKeyException

表示输入的 KMS 密钥没有解密权限，或者 KMS 密钥输入错误。

HTTP 状态代码：400

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，`InvalidParameterException`如果两者都不会发生异常`S3Object`要么`Bytes`值在`Document`请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，配置对 Amazon S3 的访问权限有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

LimitExceededException

超出了 Amazon Textract 服务限制。例如，如果您同时启动太多异步作业，则调用启动操作（`StartDocumentTextDetection`例如）引发 LimitExceededException 异常 (HTTP 状态代码：400)，直到并发运行的任务数量低于 Amazon Textract 服务限制。

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

UnsupportedDocumentException

不支持输入文档的格式。用于操作的文档可以是 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 状态代码：400

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包

- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

StartExpenseAnalysis

开始对发票或收据进行异步分析，以获取联系信息、购买物品和供应商名称等数据。

StartExpenseAnalysis可以分析 JPEG、PNG 和 PDF 格式的文档中的文本。必须将这些文档存储在 Amazon S3 存储桶中。使用DocumentLocation参数，指定您的 S3 存储桶的名称和该存储桶中的文档名称。

StartExpenseAnalysis返回作业标识符 (JobId) 你将提供给GetExpenseAnalysis检索操作结果。完成对输入发票/收据的分析后，Amazon Textract 将完成状态发布到您提供给NotificationChannel. 要获取发票和收据分析操作的结果，请确保发布到 Amazon SNS 主题的状态值为SUCCEEDED. 如果是的话，打电话GetExpenseAnalysis，然后传递作业标识符 (JobId) 那是通过你的电话返回的StartExpenseAnalysis.

有关更多信息，请参阅 。分析发票和收据.

请求语法

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

请求参数

请求接受采用 JSON 格式的以下数据。

ClientRequestToken

用于标识启动请求的幂等令牌。如果你对多个使用同一个令牌StartDocumentTextDetection请求，同样JobId返回。使用ClientRequestToken以防止同样的工作不止一次被意外启动。有关更多信息，请参阅 。调用 Amazon Textract 异步操作

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

：必填项：否

DocumentLocation

要处理的文档的位置。

类型：DocumentLocation 对象

：必填项：是

JobTag

您指定的标识符包含在发布到 Amazon SNS 主题的完成通知中。例如，您可以使用JobTag以确定完成通知对应的文件类型（例如纳税表或收据）。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：[a-zA-Z0-9_.\-:]+

：必填项：否

KMSKeyId

用于加密推断结果的 KMS 密钥。这可以是密钥 ID 或密钥别名格式。提供 KMS 密钥后，KMS 密钥将用于对客户存储桶中的对象进行服务器端加密。如果未启用此参数，结果将使用 SSE-S3 加密服务器端。

类型: 字符串

长度约束：最小长度为 1。长度上限为 2048。

模式：^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$

：必填项：否

NotificationChannel

您希望 Amazon Textract 将操作的完成状态发布到的 Amazon SNS 主题 ARN。

类型：NotificationChannel 对象

：必填项：否

OutputConfig

设置输出是否转到客户定义的存储桶。默认情况下，Amazon Textract 将在内部保存结果以供GetExpenseAnalysisoperation.

类型：OutputConfig 对象

：必填项：否

响应语法

```
{
   "JobId": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回的以下数据。

JobId

文本检测任务的唯一标识符。这些区域有：JobId从返回StartExpenseAnalysis. 一个JobId值仅在 7 天内有效。

类型: 字符串

长度约束：最小长度为 1。最大长度为 64。

模式：^[a-zA-Z0-9-_]+$

错误

AccessDeniedException

您无权执行该操作。使用授权用户或 IAM 角色的 Amazon 资源名称 (ARN) 来执行操作。

HTTP 状态代码：400

BadDocumentException

Amazon Textract 无法阅读文档。有关 Amazon Textract 中文档限制的更多信息，请参阅Amazon Textract 中的硬性限制.

HTTP 状态代码：400

DocumentTooLargeException

无法处理该文档，因为它太大。同步操作的最大文档大小为 10 MB。对于 PDF 文件，异步操作的最大文档大小为 500 MB。

HTTP 状态代码：400

IdempotentParameterMismatchException

一个`ClientRequestToken`输入参数与操作一起重用，但至少有一个其他输入参数与先前对该操作的调用不同。

HTTP 状态代码：400

InternalServerError

Amazon Textract 遇到了一个服务问题。重新尝试您的调用。

HTTP 状态代码：500

InvalidKMSKeyException

表示输入的 KMS 密钥没有解密权限，或者 KMS 密钥输入错误。

HTTP 状态代码：400

InvalidParameterException

有一个输入参数违反了约束。例如，在同步操作中，`InvalidParameterException`如果两者都不会发生异常`S3Object`要么`Bytes`值在`Document`请求参数。先验证您的参数，然后重新调用 API 操作。

HTTP 状态代码：400

InvalidS3ObjectException

Amazon Textract 无法访问请求中指定的 S3 对象。有关更多信息，配置对 Amazon S3 的访问有关故障排除信息，请参阅。Amazon S3 故障排除

HTTP 状态代码：400

LimitExceededException

超出了 Amazon Textract 服务限制。例如，如果您同时启动太多异步作业，则调用启动操作（`StartDocumentTextDetection`例如）引发 LimitExceededException 异常 (HTTP 状态代码：400)，直到并发运行的任务数量低于 Amazon Textract 服务限制。

HTTP 状态代码：400

ProvisionedThroughputExceededException

请求数超出了您的吞吐量限制。如要增加此限制，请联系 Amazon Textract。

HTTP 状态代码：400

ThrottlingException

Amazon Textract 暂时无法处理该请求。重新尝试您的调用。

HTTP 状态代码：500

UnsupportedDocumentException

不支持输入文档的格式。用于操作的文档可以是 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 状态代码：400

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- AWS Command Line Interface
- 适用于 .NET 的AWS开发工具包
- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包

- 适用于 JavaScript 的AWS开发工具包
- 适用于 PHP V3 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

数据类型

支持以下数据类型：

- AnalyzeIDDetections
- Block
- BoundingBox
- Document
- DocumentLocation
- DocumentMetadata
- ExpenseDetection
- ExpenseDocument
- ExpenseField
- ExpenseType
- Geometry
- HumanLoopActivationOutput
- HumanLoopConfig
- HumanLoopDataAttributes
- IdentityDocument
- IdentityDocumentField
- LineItemFields
- LineItemGroup
- NormalizedValue
- NotificationChannel
- OutputConfig
- Point

- Relationship
- S3Object
- Warning

AnalyzeIDDetections

用于包含 AnalyzeID 操作检测到的信息。

目录

Confidence

检测到的文本的置信度得分。

类型: Float

有效范围: 最小值为 0。最大值为 100。

必填项：否

NormalizedValue

仅返回日期，返回检测到的值的类型以及以机器可读性更强的方式写入的日期。

类型：NormalizedValue 对象

必填项：否

Text

标准化字段或与其关联的值的文本。

类型: 字符串

必填项：是

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

Block

一个Block表示在彼此接近的一组像素内在文档中识别的项目。返回的信息在Block对象取决于操作的类型。在文档的文本检测中（例如DetectDocumentText)，您可以获得有关检测到的单词和文本行的信息。在文本分析中（例如AnalyzeDocument)，您还可以获取有关文档中检测到的字段、表格和选择元素的信息。

数组Block对象由同步操作和异步操作返回。在同步操作中，例如DetectDocumentText，数组Block对象是整个结果集。在异步操作中，例如GetDocumentAnalysis，数组将通过一个或多个响应返回。

有关更多信息，请参阅 。Amazon Textract 的工作原理.

目录

BlockType

已识别的文本项目的类型。在文本检测操作中，返回以下类型：

- 页-包含 LINE 列表Block在文档页面上检测到的对象。
- 单词-在文档页面上检测到的单词。单词 是一个或多个 ISO 基本拉丁字母字符，不用空格分隔。
- 线-在文档页面上检测到的制表符分隔的连续单词的字符串。

在文本分析操作中，返回以下类型：

- 页-包含孩子列表Block在文档页面上检测到的对象。
- KEY_VALUE_SET-存储 KEY 和 VALUEBlock在文档页面上检测到的链接文本的对象。使用EntityType字段来确定 KEY_VALUE_SET 对象是否为 KEYBlock对象或 VALUEBlock对象。
- 单词-在文档页面上检测到的单词。单词 是一个或多个 ISO 基本拉丁字母字符，不用空格分隔。
- 线-在文档页面上检测到的制表符分隔的连续单词的字符串。
- 桌子-在文档页面上检测到的表格。表格是基于网格的信息，包含两行或多列，单元格跨度为一行和一列。
- 细胞-检测到的桌子里的一个细胞。单元格是包含单元格中文本的块的父项。
- 选择_元素-在文档页面上检测到的选择元素，例如选项按钮（单选按钮）或复选框。使用的值SelectionStatus以确定选择元素的状态。

类型: 字符串

有效值: KEY_VALUE_SET | PAGE | LINE | WORD | TABLE | CELL | SELECTION_ELEMENT

：必需 否

ColumnIndex

显示表格单元格的列。第一列位置是 1。ColumnIndex不是由DetectDocumentText和GetDocumentTextDetection.

类型: 整数

有效范围：最小值为 0。

：必需 否

ColumnSpan

表格单元格跨越的列数。目前，该值始终为 1，即使跨越的列数大于 1。ColumnSpan不是由DetectDocumentText和GetDocumentTextDetection.

类型: 整数

有效范围：最小值为 0。

：必需 否

Confidence

Amazon Textract 对已识别文本的准确性以及几何结构的准确性指向识别文本周围的信心得分。

类型: Float

有效范围：最小值为 0。最大值为 100。

：必需 否

EntityTypes

实体的类型。可能返回以下内容：

- 密钥-文档上字段的标识符。
- 值-字段文本。

EntityTypes不是由DetectDocumentText和GetDocumentTextDetection.

类型: 字符串数组

有效值: KEY | VALUE

：必需 否

Geometry

图像上可识别的文本的位置。它包括围绕文本的轴对齐、粗糙的边界框以及一个用于更准确的空间信息的精细多边形。

类型：Geometry 对象

：必需 否

Id

识别文本的标识符。该标识符只对于单个操作是唯一的。

类型: 字符串

模式：.*\S.*

：必需 否

Page

检测到块的页面。Page是由异步操作返回的。仅对于 PDF 或 TIFF 格式的多页文档返回大于 1 的页面值。扫描的图像 (JPEG/PNG)，即使它包含多个文档页面，也被视为单页文档。的价值Page始终为 1。同步操作不会返回Page因为每个输入文档都被视为单页文档。

类型: 整数

有效范围：最小值为 0。

：必需 否

Relationships

当前区块的子区块的列表。例如，LINE 对象都有作为文本行一部分的每个 WORD 块的子块。列表中没有关系不存在的关系对象，例如当前区块没有子块时。列表大小可以是以下内容：

- 0-该区块没有子方块。
- 1-该区块有子方块。

类型: 数组Relationship对象

：必需 否

RowIndex

表格单元格所在的行。第一行位置是 1。RowIndex不是由DetectDocumentText和GetDocumentTextDetection.

类型: 整数

有效范围：最小值为 0。

：必需 否

RowSpan

表格单元格跨越的行数。目前，该值始终为 1，即使跨越的行数大于 1。RowSpan不是由DetectDocumentText和GetDocumentTextDetection.

类型: 整数

有效范围：最小值为 0。

：必需 否

SelectionStatus

选择元素的选择状态，例如选项按钮或复选框。

类型: 字符串

有效值: SELECTED | NOT_SELECTED

：必需 否

Text

Amazon Textract 识别的单词或一行文本。

类型: 字符串

：必需 否

TextType

Amazon Textract 检测到的文本类型。可以检查手写文本和印刷文本。

类型: 字符串

有效值：HANDWRITING | PRINTED

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

BoundingBox

文档页面上检测到的页面、文本、键值对、表格、表格单元格或选择元素周围的边界框。这些区域有：`left`（x 坐标）和`top`（y 坐标）是表示边界框顶部和左侧的坐标。请注意，图像的左上角是原点 (0,0)。

这些区域有：`top`和`left`返回的值是整个文档页面大小的比率。例如，如果输入图像为 700 x 200 像素，而边界框的左上坐标为 350 x 50 像素，则 API 将返回 `left` 值 0.5 (350/700) 和 `top` 值 0.25 (50/200)。

这些区域有：`width`和`height`值表示边界框的维度（以占整个文档页面维度的比例显示）。例如，如果文档页面大小为 700 x 200 像素，且边界框宽度为 70 像素，则返回的宽度为 0.1。

目录

Height

高度边界框的高度（以占整个文档页面高度的比例显示）。

类型: Float

必需 否

Left

左坐标（以占整个文档页面宽度的比例显示）。

类型: Float

必需 否

Top

顶部边界框的顶部坐标（以占整个文档页面高度的比例显示）。

类型: Float

必需 否

Width

宽度边界框的宽度（以占整个文档页面宽度的比例显示）。

类型: Float

必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

Document

输入文档，可以是字节或作为 S3 对象。

您可以使用`Bytes`财产。例如，您应使用`Bytes`属性来传递从本地文件系统加载的文档。通过使用`Bytes`属性必须采用 base64 编码。如果您使用 AWS 开发工具包调用 Amazon Textract API 操作，则代码可能不需要对文档文件字节进行编码。

您可以使用存储在 S3 存储桶中的图像传递给 Amazon Textract API 操作。`S3Object`财产。存储在 S3 存储桶中的文档不需要 base64 编码。

包含 S3 对象的 S3 存储桶的 AWS 区域必须与您用于 Amazon Textract 操作的 AWS 区域匹配。

如果您使用 AWS CLI 调用 Amazon Textract 操作，则不支持使用字节属性传递图像字节。您必须先将文档上传到 Amazon S3 存储桶，然后再使用 S3Object 属性调用操作。

如果 Amazon Textract 处理 S3 对象，用户必须具有访问 S3 对象的权限。

目录

Bytes

base64 编码的文档字节的 blob。以 BLOB 字节为单位提供的文档的最大大小为 5 MB。文档字节必须采用 PNG 或 JPEG 格式。

如果您使用 AWS 开发工具包调用 Amazon Textract，则可能不需要对使用`Bytes`字段中返回的子位置类型。

类型: Base64 编码的二进制数据对象

长度约束：最小长度为 1。最大长度为 10485760。

必填项：否

S3Object

将 S3 对象标识为文档源。存储在 S3 存储桶中的文档的最大大小为 5 MB。

类型：S3Object 对象

必填项：否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

DocumentLocation

包含要处理的文档的 Amazon S3 存储桶。它被异步操作使用，例如StartDocumentTextDetection.

输入文档可以是 JPEG 或 PNG 的图像文件。它也可以是 PDF 格式的文件。

目录

S3Object

包含输入文档的 Amazon S3 存储桶。

类型：S3Object 对象

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

DocumentMetadata

有关输入文档的信息。

目录

Pages

在文档中检测到的页数。

类型: 整数

有效范围：最小值为 0。

对于是必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

ExpenseDetection

用于存储有关 Amazon Textract 检测到的价值或标签的信息的对象。

目录

Confidence

对检测的信心，以百分比表示

类型: Float

有效范围：最小值为 0。最大值为 100。

：必需 否

Geometry

有关文档页面上下列项目所在位置的信息：检测到的页、文本、键值对、表、表单元和选择元素。

类型：Geometry 对象

：必需 否

Text

Amazon Textract 识别的单词或行文本

类型: 字符串

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

ExpenseDocument

保存分析所返回的所有信息的结构

目录

ExpenseIndex

表示信息来自文档中的哪张发票或收据。第一个文档为 1，第二个文档将是 2，依此类推。

类型: 整数

有效范围：最小值为 0。

：必需 否

LineItemGroups

在文档的每个表格上检测到的信息，分隔成`LineItems`.

类型: 数组为LineItemGroup对象

：必需 否

SummaryFields

Amazon Textract 在表外找到的任何信息。

类型: 数组为ExpenseField对象

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

ExpenseField

检测到的信息细分，分为类型、标签检测和值检测类别

目录

LabelDetection

检测到的元素的明确说明的标签。

类型：ExpenseDetection 对象

必填项：否

PageNumber

检测到该值的页码。

类型: 整数

有效范围：最小值为 0。

必填项：否

Type

检测到的元素的隐含标签。与标签检测一起出现，以获取显式元素。

类型：ExpenseType 对象

必填项：否

ValueDetection

检测到的元素的值。存在于显式和隐含的元素中。

类型：ExpenseDetection 对象

必填项：否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包

- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

ExpenseType

用于存储有关 Amazon Textract 检测到的类型的信息的对象。

目录

Confidence

准确性的信心，以百分比表示。

类型: Float

有效范围：最小值为 0。最大值为 100。

必填项：否

Text

Amazon Textract 检测到的单词或行文本。

类型: 字符串

必填项：否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

Geometry

有关文档页面上的位置的信息：检测到的页、文、键值对、表、表单元和选择元素。

目录

BoundingBox

对齐轴对齐的粗略表示文档页面上已识别项目的位置。

类型：BoundingBox 对象

：必需 否

Polygon

在边界框内，识别物品周围有一个细粒度的多边形。

类型: 数组Point对象

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

HumanLoopActivationOutput

显示循环中的人类评估结果。如果没有 HumanLoopARN，则输入内容不会触发人工审查。

目录

HumanLoopActivationConditionsEvaluationResults

显示状况评估的结果，包括激活人工审查的那些条件。

类型: 字符串

长度约束：长度上限为 10240。

：必需 否

HumanLoopActivationReasons

显示是否需要以及为什么需要人工审查。

类型: 字符串数组

数组成员：最少 1 项。

：必需 否

HumanLoopArn

HumanLoop 的 Amazon 资源名称 (ARN) 已创建。

类型: 字符串

长度约束：长度上限为 256。

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包

- 适用于 Ruby V3 的 AWS 开发工具包

HumanLoopConfig

设置符合条件之一时将文档发送到的人工审阅工作流程。您还可以在审阅之前设置图像的某些属性。

目录

DataAttributes

设置输入数据的属性。

类型：HumanLoopDataAttributes 对象

：必需 否

FlowDefinitionArn

流定义的 Amazon 资源名称 (ARN)。

类型: 字符串

长度约束：长度上限为 256。

：必需 是

HumanLoopName

用于此图像的人工作流的名称。这在区域中应该唯一。

类型: 字符串

长度约束：最小长度为 1。长度上限为 63。

模式：`^[a-z0-9](-*[a-z0-9])*`

：必需 是

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包

- 适用于 Ruby V3 的 AWS 开发工具包

HumanLoopDataAttributes

允许您设置图像的属性。目前，您可以宣布图片不含个人身份信息和成人内容。

目录

ContentClassifiers

设置输入图片是不含个人身份信息还是成人内容。

类型: 字符串数组

数组成员：最多 256 项。

有效值: `FreeOfPersonallyIdentifiableInformation | FreeOfAdultContent`

必填项：否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

IdentityDocument

列出了在分析 ID 操作中处理的每个文档的结构。

目录

DocumentIndex

表示文档在身份文档列表中的位置。第一个文档标记为 1，第二个 2，依此类推。

类型: 整数

有效范围：最小值为 0。

：必需 否

IdentityDocumentFields

用于记录从身份证件中提取的信息的结构。包含标准化字段和提取文本的值。

类型: 数组IdentityDocumentField对象

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

IdentityDocumentField

包含已提取信息的标准化类型以及与之关联的文本的结构。它们分别作为类型和值提取。

目录

Type

用于包含 AnalyzeID 操作检测到的信息。

类型：AnalyzeIDDetections 对象

必填项: 否

ValueDetection

用于包含 AnalyzeID 操作检测到的信息。

类型：AnalyzeIDDetections 对象

必填项: 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

LineItemFields

保存有关文档表格中不同行的信息的结构。

目录

LineItemExpenseFields

ExpenseFields 用于显示表格上检测到的行中的信息。

类型: 数组ExpenseField对象

必填项：否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

LineItemGroup

一组包含 lineItem 的表格，其中每个表都由表格标识`LineItemGroupIndex`.

目录

LineItemGroupIndex

用于标识文档中特定表的数字。遇到的第一张表格的 lineItemGroupIndex 将为 1，第二个 2 等。

类型: 整数

有效范围：最小值为 0。

：必填项：否

LineItems

表格中特定行的信息细目。

类型: 数组LineItemFields对象

：必填项：否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

NormalizedValue

包含与文档中的日期相关的信息，包括值的类型和值。

目录

Value

日期的值，写为年-月-日小时：分钟：秒。

类型: 字符串

：必需 否

ValueType

检测到的值的标准化类型。在这种情况下，为 DATE。

类型: 字符串

有效值: DATE

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

NotificationChannel

Amazon Textract 向其发布异步文档操作的完成状态的 Amazon Simple Notification Service (Amazon SNS) 主题，例如StartDocumentTextDetection.

目录

RoleArn

向 Amazon Textract 授予 Amazon SNS 主题发布权限的 Amazon 资源名称 (ARN)。

类型: 字符串

长度约束：最小长度为 20。长度上限为 2048。

模式：`arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

必填项：是

SNSTopicArn

Amazon Textract 将完成状态发布到的 Amazon SNS 主题。

类型: 字符串

长度约束：最小长度为 20。长度上限为 1024。

模式：`(^arn:([a-z\d-]+):sns:[a-zA-Z\d-]{1,20}:\w{12}:.+$)`

必填项：是

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

OutputConfig

设置您的输出是否将转到用户创建的存储桶。用于设置存储桶的名称和输出文件上的前缀。

OutputConfig是一个可选参数，它允许你调整输出的放置位置。默认情况下，Amazon Textract 将在内部存储结果，并且只能通过获取 API 操作访问。启用 OutputConfig 后，您可以设置输出将发送到的存储桶的名称以及可以下载结果的结果的文件前缀。此外，您还可以设置KMSKeyID将参数转换为客户主密钥 (CMK)，以加密您的输出。如果不设置此参数，Amazon Textract 将使用适用于 Amazon S3 的 AWS 托管 CMK 加密服务器端。

Amazon Textract 处理文档时，必须解密买家内容。如果您的账户根据 AI 服务退出政策选择退出，那么在服务处理客户内容后，所有未加密的客户内容都将立即永久删除。Amazon Textract 不保留输出的副本。有关如何退出的信息，请参阅管理 AI 服务选择退出策略。

有关数据隐私的更多信息，请参阅常见问题.

目录

S3Bucket

输出将转到的存储桶的名称。

类型: 字符串

长度约束：最小长度为 3。长度上限为 255。

模式：[0-9A-Za-z\.\-_]*

必填项：是

S3Prefix

将保存输出的对象键的前缀。如果未启用，前缀将为 “textract_output”。

类型: 字符串

长度约束：最小长度为 1。长度上限为 1024。

模式：.*\S.*

必填项：否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

Point

文档页面上点的 X 和 Y 坐标。返回的 X 和 Y 值是整个文档页面大小的比率。例如，如果输入文档为 700 x 200，操作返回 X=0.5 且 Y=0.25，则该点位于文档页面上的 (350,50) 像素坐标处。

数组Point对象，Polygon作为Geometry返回的对象Block对象。一个Polygon对象表示围绕检测到的文本、键值对、表格、表格单元格或选择元素周围的细粒度多边形。

目录

X

X 坐标的值表示Polygon.

类型: Float

：必需 否

Y

A 上某个点的 Y 坐标值Polygon.

类型: Float

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

Relationship

关于区块之间如何相关的信息。一个Block对象包含 0 或更大值Relation列表中的对象，Relationships. 有关更多信息，请参阅 Block。

这些区域有：Type元素提供了中所有区块的关系类型IDs数组。

目录

Ids

相关区块的 ID 数组。您可以从Type元素。

类型: 字符串数组

模式：.*\S.*

：必填项 否

Type

ID 数组中的块与当前模块之间的关系类型。这种关系可能是VALUE要么CHILD. VALUE 类型的关系是一个列表，其中包含与键值对的 KEY 关联的 VALUE 块的 ID。孩子类型的关系是一个 ID 列表，用于在表格的情况下标识单元格块的单元格块，在选择元素的情况下标识 WORD 块。

类型: 字符串

有效值: VALUE | CHILD | COMPLEX_FEATURES

：必填项 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

S3Object

用于标识文档的 S3 存储桶名称和文件名。

包含文档的 S3 存储桶的 AWS 区域必须与您用于 Amazon Textract 操作的区域匹配。

要使 Amazon Textract 处理 S3 存储桶中的文件，用户必须拥有访问 S3 存储桶和文件的权限。

目录

Bucket

S3 存储桶的名称。请注意，文件名中的 # 字符无效。

类型: 字符串

长度约束：最小长度为 3。长度上限为 255。

模式：`[0-9A-Za-z\.\-_]*`

：必需 否

Name

输入文档的文件名。同步操作可以使用 JPEG 或 PNG 格式的图像文件。异步操作还支持 PDF 和 TIFF 格式文件。

类型: 字符串

长度约束：最小长度为 1。长度上限为 1024。

模式：`.*\S.*`

：必需 否

Version

如果存储桶已启用版本控制，则可指定对象版本。

类型: 字符串

长度约束：最小长度为 1。长度上限为 1024。

模式：`.*\S.*`

：必需 否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

Warning

关于异步文本分析期间发生的问题的警告 (StartDocumentAnalysis) 或异步文档文本检测 (StartDocumentTextDetection)。

目录

ErrorCode

警告的错误代码。

类型: 字符串

必填项：否

Pages

警告适用的页面列表。

类型: 整数

有效范围：最小值为 0。

必填项：否

另请参阅

有关在特定语言的AWS软件开发工具包中使用此 API 的更多信息，请参阅以下内容：

- 适用于 C++ 的AWS开发工具包
- 适用于 Go 的AWS开发工具包
- AWS适用于 Java V2 的开发工具包
- 适用于 Ruby V3 的 AWS 开发工具包

Amazon Textract 中的硬性限制

以下是无法更改的 Amazon Textract 中的硬性限制的列表。有关位置限制和限制的信息，可更改的信息，请参阅Amazon Textract 终端节点和配额. 有关可更改的限制的信息，请参阅 AWS 服务限制。要更改限制，请参阅创建案例。

Amazon Textract

限制	描述
接受的文件格式	操作支持 JPEG、PNG、PDF 和 TIFF 文件。（支持 PDF 中的 JPEG 2000 编码图像）..
文件大小和页数限制	对于同步操作，JPEG、PNG、PDF 和 TIFF 文件的大小限制为 10MB。PDF 和 TIFF 文件的页面限制为 1 页。对于异步操作，JPEG 和 PNG 文件的大小限制为 10MB。PDF 和 TIFF 文件有 500MB 的限制。PDF 和 TIFF 文件的页面限制为 3,000 页。
特定于 PDF 的限制	最大高度和宽度为 40 英寸和 2880 点。PDF 不能受密码保护。PDF 可以包含 JPEG 2000 格式化的图像。
文档旋转和图像大小	Amazon Textract 支持所有飞机内文档旋转，例如 45 度平面旋转。 Amazon Textract 支持四面分辨率小于或等于 10000 像素的图像。
对齐方式	文本可以在文档中水平对齐文本。Amazon Textract 不支持文档中的垂直文本对齐。
语言	Amazon Textract 支持英语、法语、德语、意大利语、葡萄牙语和西班牙语文本检测。Amazon Textract 不会返回输出中检测到的语言。
角色大小	要检测到的文本的最小高度为 15 像素。在 150 DPI 时，这将与 8 点字体相同。
字符类型	Amazon Textract 支持手写和印刷字符识别。

限制	描述
字符	Amazon Textract 检测到以下字符： • a-z • A-Z • 0-9 • ä ö ü ü ü ç é é Â â à à è ù ë ë è ù ë ë ï ü á é í í ü á é í ó ú ü • ！ “# $% '& () * +,-./:; =? @ [\] ^ _ `{|} ~ > <° € ¥ ¥ #ß ß ¿€ ¥ ¥ ##ø Ø©®™ § ¹ ² ³
AnalyZEd 特定于限制	AnalyZEID 仅支持美国护照和美国驾驶执照。

Amazon Textract 的文档历史记录

下表介绍每一个发行版中的重大更改。Amazon Textract 开发人员指南. 如需对此文档更新的通知，您可以订阅 RSS 源。

- 最近文档更新时间：2019 年 5 月 29 日

update-history-change	update-history-description	update-history-date
集成来自的代码示例AWSDocs SDK 代码示例 GitHub 仓库	Amazon Textract 指南现在包含其他代码示例。将之前的示例部分重命名为教程	2022 年 1 月 30 日
已将分析开支添加到	Amazon Textract 现在支持使用分析 AyazeFecments API 对发票和收据单据进行分析。本功能仅适用于我们的亚太地区（孟买）、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）、加拿大（中部）、欧洲（爱尔兰）、欧洲（伦敦）、美国东部（弗吉尼亚北部）、美国东部（俄亥俄）美国西部（加利福尼亚北部）和美国西部（俄勒冈）地区。	2021 年 7 月 26 日
Augmented AI Support	Amazon Textract 现在支持 Amazon Augmented AI 来实施人工审核。	2019 年 12 月 3 日
新服务和指南	Amazon Textract 现可用于一般用途。	2019 年 5 月 29 日
Support 选择元素	Amazon Textract 现在可以检测选择元素（单选按钮和复选框）。	2019 年 4 月 24 日

Amazon Textract 的发布时间	这是 Amazon Textract 文档的第一个版本。	2018 年 11 月 28 日

AWS词汇表

有关最新AWS术语，请参阅《AWS一般参考》中的AWS术语表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。