
Amazon Comprehend

Developer Guide



Amazon Comprehend: Developer Guide

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Amazon Comprehend?	1
Comprehend Custom	1
Document Clustering (Topic Modeling)	2
Examples	2
Benefits	2
Are You a First-time User of Amazon Comprehend?	3
How It Works	4
Supported Languages	5
Supported Languages	5
Languages Supported by Amazon Comprehend Features	5
Tutorial: Analyzing Insights from Reviews	7
Prerequisites	8
Step 1: Adding Documents to Amazon S3	9
Prerequisites	10
Download Sample Data	10
Create an Amazon S3 Bucket	10
(Console Only) Create Folders	11
Upload the Input Data	12
Step 2: (CLI Only) Creating an IAM Role	13
Prerequisites	13
Create an IAM Role	13
Attach an IAM Policy to the IAM Role	14
Step 3: Running Analysis Jobs	15
Prerequisites	15
Analyze Sentiment and Entities	16
Step 4: Preparing the Output	18
Prerequisites	18
Download the Output	18
Extract the Output Files	19
Upload the Extracted Files	20
Load the Data into an AWS Glue Data Catalog	21
Prepare the Data for Analysis	24
Step 5: Visualizing the Output	26
Prerequisites	26
Give Amazon QuickSight Access	26
Import the Datasets	27
Create a Sentiment Visualization	27
Create an Entities Visualization	28
Publish a Dashboard	29
Clean Up	30
Getting Started	31
Step 1: Set Up an Account	31
Sign Up for AWS	31
Create an IAM User	31
Next Step	32
Step 2: Set Up the AWS CLI	32
Next Step	33
Step 3: Getting Started Using the Console	33
Analyzing Documents Using the Console	33
Creating and Using Custom Entity Recognizer	40
Creating and Using Custom Classifiers	44
Model Versioning with Amazon Comprehend	49
Creating a Topic Modeling Job Using the Console	51
Creating an Events Detection Job Using the Console	53

Step 4: Getting Started Using the API	54
Detecting the Dominant Language	54
Detecting Named Entities	57
Detecting Key Phrases	59
Detecting PII	61
Labeling Documents with PII	62
Detecting Sentiment	63
Detecting Syntax	65
Using Custom Classification	68
Detecting Custom Entities	72
Detecting Events	76
Topic Modeling	78
Using the Batch APIs	84
Solution: Analyzing Text with Amazon Elasticsearch Service	89
Using S3 Object Lambda Access Points for PII	90
Controlling Access to Documents with PII	90
Creating an Amazon S3 Object Lambda Access Point to Control Access to Documents	91
Invoking an Amazon S3 Object Lambda Access Point to Control Access to Documents	91
Redacting PII from Documents	92
Creating an Amazon S3 Object Lambda Access Point to Redact PII from Documents	91
Invoking an Amazon S3 Object Lambda Access Point to Redact PII from Documents	91
Text Analysis APIs	94
Detect Entities	94
Detect Events	96
.....	96
Supported Types for Entities, Events, and Arguments	97
Detect Key Phrases	101
Detect the Dominant Language	102
Detect PII	106
PII Entity Types	106
Locate PII Entities	108
Redact PII Entities	109
Label Documents with PII	110
Label Documents with PII Entity Types	110
Determine Sentiment	111
Analyze Syntax	112
Topic Modeling	114
Document Processing Modes	117
Single-Document Processing	117
Asynchronous Batch Processing	117
Prerequisites	118
Starting an Analysis Job	118
Monitoring Analysis Jobs	119
Getting Analysis Results	119
Multiple Document Synchronous Processing	122
Comprehend Custom	125
Custom Classification	125
Multi-Class and Multi-Label Modes	125
Asynchronous Classification	126
Training a Custom Classifier	126
Running an Asynchronous Classification Job	132
Real-time Analysis with Custom Classification	135
Tagging	137
Metrics	142
Custom Entity Recognition	147
Training Custom Entity Recognizers	148
Detecting Custom Entities with a Batch Job	160

Detecting Custom Entities in Real Time	161
Tagging	163
Metrics	168
Managing Endpoints	170
Endpoints Overview	170
Monitoring an Endpoint	171
Updating an Endpoint	172
Using Trusted Advisor	174
Deleting an Endpoint	176
Auto Scaling with Endpoints	177
Security	182
Data Protection	182
KMS Encryption in Amazon Comprehend	183
Using a Virtual Private Cloud (VPC)	185
VPC endpoints (AWS PrivateLink)	188
Authentication and Access Control	190
Authentication	190
Access Control	191
Overview of Managing Access	191
Using Identity-Based Policies (IAM Policies) for Amazon Comprehend	193
Amazon Comprehend API Permissions Reference	198
AWS Managed Policies	199
Logging Amazon Comprehend API Calls with AWS CloudTrail	201
Amazon Comprehend Information in CloudTrail	201
Examples: Amazon Comprehend Log File Entries	203
Compliance Validation	210
Resilience	210
Infrastructure Security	210
Permissions Required for a Custom Asynchronous Analysis Job	211
Guidelines and Quotas	212
Supported Regions	212
Overall Quotas	212
Throttling When Using Single Transactions	212
Multiple Document Operations	212
Asynchronous Operations	213
Document Classification	213
Language Detection	214
Events	215
Topic Modeling	215
Entity Recognition	215
API Reference	218
Actions	218
BatchDetectDominantLanguage	220
BatchDetectEntities	223
BatchDetectKeyPhrases	226
BatchDetectSentiment	229
BatchDetectSyntax	232
ClassifyDocument	235
ContainsPiiEntities	238
CreateDocumentClassifier	240
CreateEndpoint	245
CreateEntityRecognizer	249
DeleteDocumentClassifier	254
DeleteEndpoint	256
DeleteEntityRecognizer	258
DescribeDocumentClassificationJob	260
DescribeDocumentClassifier	263

DescribeDominantLanguageDetectionJob	266
DescribeEndpoint	269
DescribeEntitiesDetectionJob	271
DescribeEntityRecognizer	274
DescribeEventsDetectionJob	277
DescribeKeyPhrasesDetectionJob	279
DescribePiiEntitiesDetectionJob	282
DescribeSentimentDetectionJob	285
DescribeTopicsDetectionJob	288
DetectDominantLanguage	291
DetectEntities	294
DetectKeyPhrases	298
DetectPiiEntities	301
DetectSentiment	303
DetectSyntax	306
ListDocumentClassificationJobs	309
ListDocumentClassifiers	312
ListDocumentClassifierSummaries	315
ListDominantLanguageDetectionJobs	317
ListEndpoints	320
ListEntitiesDetectionJobs	323
ListEntityRecognizers	326
ListEntityRecognizerSummaries	330
ListEventsDetectionJobs	332
ListKeyPhrasesDetectionJobs	335
ListPiiEntitiesDetectionJobs	338
ListSentimentDetectionJobs	341
ListTagsForResource	344
ListTopicsDetectionJobs	346
StartDocumentClassificationJob	349
StartDominantLanguageDetectionJob	354
StartEntitiesDetectionJob	359
StartEventsDetectionJob	364
StartKeyPhrasesDetectionJob	368
StartPiiEntitiesDetectionJob	373
StartSentimentDetectionJob	377
StartTopicsDetectionJob	382
StopDominantLanguageDetectionJob	387
StopEntitiesDetectionJob	389
StopEventsDetectionJob	391
StopKeyPhrasesDetectionJob	393
StopPiiEntitiesDetectionJob	395
StopSentimentDetectionJob	397
StopTrainingDocumentClassifier	399
StopTrainingEntityRecognizer	401
TagResource	403
UntagResource	405
UpdateEndpoint	407
Data Types	409
AugmentedManifestsListItem	411
BatchDetectDominantLanguageItemResult	413
BatchDetectEntitiesItemResult	414
BatchDetectKeyPhrasesItemResult	415
BatchDetectSentimentItemResult	416
BatchDetectSyntaxItemResult	417
BatchItemError	418
ClassifierEvaluationMetrics	419

ClassifierMetadata	421
DocumentClass	422
DocumentClassificationJobFilter	423
DocumentClassificationJobProperties	424
DocumentClassifierFilter	427
DocumentClassifierInputDataConfig	428
DocumentClassifierOutputDataConfig	430
DocumentClassifierProperties	431
DocumentClassifierSummary	435
DocumentLabel	437
DocumentReaderConfig	438
DominantLanguage	439
DominantLanguageDetectionJobFilter	440
DominantLanguageDetectionJobProperties	441
EndpointFilter	444
EndpointProperties	445
EntitiesDetectionJobFilter	448
EntitiesDetectionJobProperties	449
Entity	452
EntityLabel	454
EntityRecognizerAnnotations	455
EntityRecognizerDocuments	456
EntityRecognizerEntityList	457
EntityRecognizerEvaluationMetrics	458
EntityRecognizerFilter	459
EntityRecognizerInputDataConfig	460
EntityRecognizerMetadata	462
EntityRecognizerMetadataEntityTypeListListItem	463
EntityRecognizerProperties	464
EntityRecognizerSummary	467
EntityTypesEvaluationMetrics	469
EntityTypesListItem	470
EventsDetectionJobFilter	471
EventsDetectionJobProperties	472
InputDataConfig	475
KeyPhrase	477
KeyPhrasesDetectionJobFilter	478
KeyPhrasesDetectionJobProperties	479
OutputDataConfig	482
PartOfSpeechTag	483
PiiEntitiesDetectionJobFilter	484
PiiEntitiesDetectionJobProperties	485
PiiEntity	488
PiiOutputDataConfig	490
RedactionConfig	491
SentimentDetectionJobFilter	492
SentimentDetectionJobProperties	493
SentimentScore	496
SyntaxToken	497
Tag	498
TopicsDetectionJobFilter	499
TopicsDetectionJobProperties	500
VpcConfig	503
Common Errors	503
Common Parameters	505
Document History	508

What Is Amazon Comprehend?

Amazon Comprehend uses natural language processing (NLP) to extract insights about the content of documents. Amazon Comprehend processes any text file in UTF-8 format, and semi-structured documents, like PDF and Word documents. It develops insights by recognizing the entities, key phrases, language, sentiments, and other common elements in a document. Use Amazon Comprehend to create new products based on understanding the structure of documents. For example, using Amazon Comprehend you can search social networking feeds for mentions of products or scan an entire document repository for key phrases.

You work with one or more documents at a time to evaluate their content and gain insights about them. Some of the insights that Amazon Comprehend develops about a document include:

- **Entities** – Amazon Comprehend returns a list of entities, such as people, places, and locations, identified in a document. For more information, see [Detect Entities \(p. 94\)](#).
- **Key phrases** – Amazon Comprehend extracts key phrases that appear in a document. For example, a document about a basketball game might return the names of the teams, the name of the venue, and the final score. For more information, see [Detect Key Phrases \(p. 101\)](#).
- **PII** – Amazon Comprehend analyzes documents to detect personal data that could be used to identify an individual, such as an address, bank account number, or phone number. For more information, see [Detect Personally Identifiable Information \(PII\) \(p. 106\)](#).
- **Language** – Amazon Comprehend identifies the dominant language in a document. Amazon Comprehend can identify 100 languages. For more information, see [Detect the Dominant Language \(p. 102\)](#).
- **Sentiment** – Amazon Comprehend determines the emotional sentiment of a document. Sentiment can be positive, neutral, negative, or mixed. For more information, see [Determine Sentiment \(p. 111\)](#).
- **Syntax** – Amazon Comprehend parses each word in your document and determines the part of speech for the word. For example, in the sentence "It is raining today in Seattle," "it" is identified as a pronoun, "raining" is identified as a verb, and "Seattle" is identified as a proper noun. For more information, see [Analyze Syntax \(p. 112\)](#).

Comprehend Custom

Customize Comprehend for your specific requirements without the skillset required to build machine learning-based NLP solutions. Using automatic machine learning, or AutoML, Comprehend Custom builds customized NLP models on your behalf, using data you already have.

Custom Classification: Create custom document classifiers to organize your documents into your own categories. For each classification label, provide a set of documents that best represent that label and train your classifier on it. Once trained, a classifier can be used on any number of unlabeled document sets. You can use the console for a code-free experience or install the latest AWS SDK. For more information, see [Custom Classification \(p. 125\)](#).

Custom Entities: Create custom entity types that analyze text for your specific terms and noun-based phrases. You can train custom entities to extract terms like policy numbers, or phrases that imply a customer escalation. To train the model, you provide a list of the entities and a set of documents that contain them. Once the model is trained, you can submit analysis jobs against it to extract their custom entities. For more information, see [Custom Entity Recognition \(p. 147\)](#).

Document Clustering (Topic Modeling)

You can also use Amazon Comprehend to examine a corpus of documents to organize them based on similar keywords within them. Document clustering (topic modeling) is useful to organize a large corpus of documents into topics or clusters that are similar based on the frequency of words within them.

Topic modeling is a asynchronous process, you submit a set of documents for processing and then later get the results when processing is complete. Amazon Comprehend does topic modeling on large document sets, for best results you should include at least 1,000 documents when you submit a topic modeling job. For more information, see [Topic Modeling \(p. 114\)](#).

Examples

The following examples show how you might use the Amazon Comprehend operations in your applications.

Example 1: Find documents about a subject

Find the documents about a particular subject using Amazon Comprehend topic modeling. Scan a set of documents to determine the topics discussed, and to find the documents associated with each topic. You can specify the number of topics that Amazon Comprehend should return from the document set.

Example 2: Find out how customers feel about your products

If your company publishes a catalog, let Amazon Comprehend tell you what customers think of your products. Send each customer comment to the `DetectSentiment` operation and it will tell you whether customers feel positive, negative, neutral, or mixed about a product.

Example 3: Discover what matters to your customers

Use Amazon Comprehend topic modeling to discover the topics that your customers are talking about on your forums and message boards, then use entity detection to determine the people, places, and things that they associate with the topic. Finally, use sentiment analysis to determine how your customers feel about a topic.

Benefits

Some of the benefits of using Amazon Comprehend include:

- **Integrate powerful natural language processing into your apps**—Amazon Comprehend removes the complexity of building text analysis capabilities into your applications by making powerful and accurate natural language processing available with a simple API. You don't need textual analysis expertise to take advantage of the insights that Amazon Comprehend produces.
- **Deep learning based natural language processing**—Amazon Comprehend uses deep learning technology to accurately analyze text. Our models are constantly trained with new data across multiple domains to improve accuracy.
- **Scalable natural language processing**—Amazon Comprehend enables you to analyze millions of documents so that you can discover the insights that they contain.
- **Integrate with other AWS services**—Amazon Comprehend is designed to work seamlessly with other AWS services like Amazon S3, AWS KMS, and AWS Lambda. Store your documents in Amazon S3, or analyze real-time data with Kinesis Data Firehose. Support for AWS Identity and Access Management (IAM) makes it easy to securely control access to Amazon Comprehend operations. Using IAM, you can

create and manage AWS users and groups to grant the appropriate access to your developers and end users.

- **Encryption of output results and volume data**—Amazon S3 already enables you to encrypt your input documents, and Amazon Comprehend extends this even farther. By using your own KMS key, you can not only encrypt the output results of your job, but also the data on the storage volume attached to the compute instance that processes the analysis job. The result is significantly enhanced security.
- **Low cost**—With Amazon Comprehend, you only pay for the documents that you analyze. There are no minimum fees or upfront commitments.

Are You a First-time User of Amazon Comprehend?

If you are a first-time user of Amazon Comprehend, we recommend that you read the following sections in order:

1. [How It Works \(p. 4\)](#) – This section introduces Amazon Comprehend concepts.
2. [Getting Started with Amazon Comprehend \(p. 31\)](#) – In this section, you set up your account and test Amazon Comprehend.
3. [API Reference \(p. 218\)](#) – In this section you'll find reference documentation for Amazon Comprehend operations.

How It Works

Amazon Comprehend uses a pre-trained model to examine and analyze a document or set of documents to gather insights about it. This model is continuously trained on a large body of text so that there is no need for you to provide training data.

Amazon Comprehend can examine and analyze documents in a variety of languages, depending on the specific feature. For more information, see [Amazon Comprehend supported languages](#).

Additionally, Amazon Comprehend's [Detect the Dominant Language \(p. 102\)](#) operation can examine documents and determine the dominant language out of a far wider variety of different languages. For more information, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

With Amazon Comprehend, you can perform the following on your documents:

- [Detect the Dominant Language \(p. 102\)](#) — Examine text to determine the dominant language.
- [Detect Entities \(p. 94\)](#) — Detect textual references to the names of people, places, and items as well as references to dates and quantities.
- [Detect Key Phrases \(p. 101\)](#) — Find key phrases such as "good morning" in a document or set of documents.
- [Detect Personally Identifiable Information \(PII\) \(p. 106\)](#) — Analyze documents to detect personal data that could be used to identify an individual, such as an address, bank account number, or phone number.
- [Determine Sentiment \(p. 111\)](#) — Analyze documents and determine the dominant sentiment of the text.
- [Analyze Syntax \(p. 112\)](#) — Parse the words in your text and show the speech syntax for each word and enable you to understand the content of the document.
- [Topic Modeling \(p. 114\)](#) — Search the content of documents to determine common themes and topics.

Each operation can be processed in several ways:

- [Single-Document Processing \(p. 117\)](#) — You call Amazon Comprehend with a single document and receive a synchronous response.
- [Multiple Document Synchronous Processing \(p. 122\)](#) — You call Amazon Comprehend with a collection of up to 25 documents and receive a synchronous response.
- [Asynchronous Batch Processing \(p. 117\)](#) — You put a collection of documents into an Amazon S3 bucket and start an asynchronous operation to analyze the documents. The results of the analysis are returned in an S3 bucket.

Each operation can be encrypted both during communication and processing.

By using the integrated AWS KMS encryption, you maintain control over who can access to your encrypted data.

You can optionally provide a custom KMS key when you create your analysis job and your data will be encrypted on the storage volume attached to the ML compute instance processing the job. You can also provide a key to encrypt your output results as it's sent to the S3 bucket. If you have set up encryption on the S3 bucket that holds your input documents, this can provide you with end-to-end security.

For more information, see [KMS Encryption in Amazon Comprehend \(p. 183\)](#).

Languages Supported in Amazon Comprehend

Amazon Comprehend supports a wide variety of languages for its various features. The languages supported and the features that support them can be seen in the following tables.

Topics

- [Supported Languages \(p. 5\)](#)
- [Languages Supported by Amazon Comprehend Features \(p. 5\)](#)

Supported Languages

Amazon Comprehend (except the **Detect Dominant Language** feature) supports the following languages for one or more features.

Code	Language
de	German
en	English
es	Spanish
it	Italian
pt	Portuguese
fr	French
ja	Japanese
ko	Korean
hi	Hindi
ar	Arabic
zh	Chinese (simplified)
zh-TW	Chinese (traditional)

Note

Amazon Comprehend identifies the language using identifiers from RFC 5646 — if there is a 2-letter ISO 639-1 identifier, with a regional subtag if necessary, it uses that. Otherwise, it uses the ISO 639-2 3-letter code. For more information about RFC 5646, see the *IETF Tools* web site.

Languages Supported by Amazon Comprehend Features

Feature	Supported Languages
Detect the Dominant Language (p. 102)	See Detect the Dominant Language (p. 102) .

Feature	Supported Languages
Detect Entities (p. 94)	All supported languages.
Detect Key Phrases (p. 101)	All supported languages.
Detect Personally Identifiable Information (PII) (p. 106)	English
Label Documents with Personally Identifiable Information (PII) (p. 110)	English
Determine Sentiment (p. 111)	All supported languages.
Analyze Syntax (p. 112)	German (de), English (en), Spanish (es), French (fr), Italian (it), and Portuguese (pt).
Topic Modeling (p. 114)	Not dependent on the language used.
Custom Classification (p. 125)	German (de), English (en), Spanish (es), French (fr), Italian (it), and Portuguese (pt).
Custom Entity Recognition (p. 147)	German (de), English (en), Spanish (es), French (fr), Italian (it), and Portuguese (pt).

Tutorial: Analyzing Insights from Customer Reviews with Amazon Comprehend

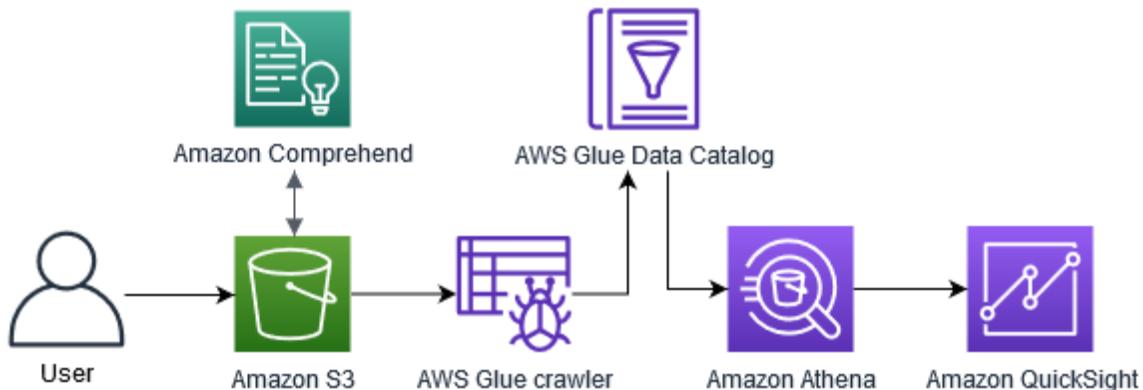
This tutorial explains how to use Amazon Comprehend with [Amazon Simple Storage Service](#), [AWS Glue](#), [Amazon Athena](#), and [Amazon QuickSight](#) to gain valuable insights into your documents. Amazon Comprehend can extract sentiment (the mood of a document) and entities (names of people, organizations, events, dates, products, places, quantities, and titles) from unstructured text.

For example, you can get actionable insights from customer reviews. In this tutorial, you analyze a sample dataset of customer reviews about a novel. You use Amazon Comprehend sentiment analysis to determine whether customers feel positive or negative about the novel. You also use Amazon Comprehend entities analysis to discover mentions of important entities, such as related novels or authors. After following this tutorial, you might discover that over 50% of the reviews are positive. You might also discover that customers are comparing authors and expressing interest in other classic novels.

In this tutorial, you accomplish the following:

- Store a sample dataset of reviews in [Amazon Simple Storage Service](#) (Amazon S3). Amazon Simple Storage Service is an object storage service.
- Use [Amazon Comprehend](#) to analyze the sentiment and entities in the review documents.
- Use an [AWS Glue](#) crawler to store the results of the analysis in a database. AWS Glue is an extract, transform, and load (ETL) service that lets you catalog and clean your data for analytics.
- Run [Amazon Athena](#) queries to clean your data. Amazon Athena is a serverless interactive query service.
- Create visualizations with your data in [Amazon QuickSight](#). Amazon QuickSight is a serverless business intelligence tool for extracting insights from your data.

The following diagram shows the workflow.



Estimated time to complete this tutorial: 1 hour

Estimated cost: Some of the actions in this tutorial incur charges on your AWS account. For information about the charges for each of these services, see the following pricing pages.

- [Amazon S3 pricing](#)
 - [Amazon Comprehend pricing](#)
 - [AWS Glue pricing](#)
 - [Amazon Athena pricing](#)
 - [Amazon QuickSight pricing](#)

Topics

- Prerequisites (p. 8)
 - Step 1: Adding Documents to Amazon S3 (p. 9)
 - Step 2: (CLI Only) Creating an IAM Role for Amazon Comprehend (p. 13)
 - Step 3: Running Analysis Jobs on Documents in Amazon S3 (p. 15)
 - Step 4: Preparing the Amazon Comprehend Output for Data Visualization (p. 18)
 - Step 5: Visualizing Amazon Comprehend Output in Amazon QuickSight (p. 26)

Prerequisites

To complete this tutorial, you need the following:

- An AWS account. If you don't have an AWS account, see the topic [Step 1: Set up an AWS Account and Create an Administrator User](#) in the *Amazon Comprehend User Guide* to set up a new account.
 - An [AWS Identity and Access Management \(IAM\)](#) user. We highly recommend that you use an IAM user to protect your root account. The root account has unrestricted access to AWS resources and billing information. Using an IAM user with restricted permissions limits how much access you have within your account. To learn how to set up an IAM user and group for your account, see the [Getting Started](#) tutorial in the *IAM User Guide*.
 - The following permissions policy attached to your IAM group or user. The policy grants your IAM user some of the permissions required to complete this tutorial. The next prerequisite describes the additional permissions you need.

```
"glue>CreateTable",
"glue>DeleteCrawler",
"glue>GetCrawlerMetrics",
"glue>GetDatabases",
"glue>GetSecurityConfigurations",
"glue>GetTable",
"glue>GetTables",
"glue>GetTags",
"glue>ListCrawlers",
"glue>ListCrawlers",
"glue>StartCrawler",
"iam:AttachRolePolicy",
"iam>CreatePolicy",
"iam>CreatePolicyVersion",
"iam>CreateRole",
"iam>DeletePolicyVersion",
"iam>DeleteRole",
"iam>DetachRolePolicy",
"iam>GetPolicy",
"iam>GetPolicyVersion",
"iam>GetRole",
"iam>ListAccountAliases",
"iam>ListAttachedRolePolicies",
"iam>ListEntitiesForPolicy",
"iam>ListPolicies",
"iam>ListPolicyVersions",
"iam>ListRoles",
"quicksight:*",
"s3:*",
>tag:GetResources"
],
"Resource": "*"
},
{
  "Action":
  [
    "iam:PassRole"
  ],
  "Effect": "Allow",
  "Resource":
  [
    "arn:aws:iam::*:role/*Comprehend*",
    "arn:aws:iam::*:role/*Glue*"
  ]
}
```

Use the previous policy to create an IAM policy and attach it to your group or user. For information about creating an IAM policy, see [Creating IAM policies](#) in the *IAM User Guide*. For information about attaching an IAM policy, see [Adding and removing IAM identity permissions](#) in the *IAM User Guide*.

- Managed policies attached to your IAM group or user. In addition to the previous policy, you must also attach the AWS managed policies `AWSGlueConsoleFullAccess` and `AWSQuicksightAthenaAccess` to your group or user. These managed policies give you permission to use AWS Glue, Amazon Athena, and Amazon QuickSight. For information about attaching an IAM policy, see [Adding and removing IAM identity permissions](#) in the *IAM User Guide*.

Step 1: Adding Documents to Amazon S3

Before starting the Amazon Comprehend analysis jobs, you need to store a sample dataset of customer reviews in Amazon Simple Storage Service (Amazon S3). Amazon S3 hosts your data in containers called

buckets. Amazon Comprehend can analyze documents stored in a bucket and it sends results of the analysis to a bucket. In this step, you create an S3 bucket, create input and output folders in the bucket, and upload a sample dataset to the bucket.

Topics

- [Prerequisites \(p. 10\)](#)
- [Download Sample Data \(p. 10\)](#)
- [Create an Amazon S3 Bucket \(p. 10\)](#)
- [\(Console Only\) Create Folders \(p. 11\)](#)
- [Upload the Input Data \(p. 12\)](#)

Prerequisites

Before you begin, review [Tutorial: Analyzing Insights from Customer Reviews with Amazon Comprehend \(p. 7\)](#) and complete the prerequisites.

Download Sample Data

The following sample dataset contains Amazon reviews taken from the larger dataset "Amazon reviews - Full", which was published with the article "Character-level Convolutional Networks for Text Classification" (Xiang Zhang et al., 2015). Download the dataset to your computer.

To get the sample data

1. Download the zip file [tutorial-reviews-data.zip](#) to your computer.
2. Extract the zip file on your computer. There are two files. The file `THIRD_PARTY_LICENSES.txt` is the open source license for the dataset published by Xiang Zhang et al. The file `amazon-reviews.csv` is the dataset you analyze in the tutorial.

Create an Amazon S3 Bucket

After downloading the sample dataset, create an Amazon S3 bucket to store your input and output data. You can create an S3 bucket using the Amazon S3 console or the AWS Command Line Interface (AWS CLI).

Create an Amazon S3 Bucket (Console)

In the Amazon S3 console, you create a bucket with a name that is unique in all of AWS.

To create an S3 bucket (console)

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In **Buckets**, choose **Create bucket**.
3. For **Bucket name**, enter a globally unique name that describes the bucket's purpose.
4. For **Region**, choose the AWS Region where you want to create the bucket. The Region you choose must support Amazon Comprehend. To reduce latency, choose the AWS Region closest to your geographic location that is supported by Amazon Comprehend. For a list of Regions that support Amazon Comprehend, see the [Region Table](#) in the *Global Infrastructure Guide*.
5. Leave the default settings for **Bucket settings for Block Public Access**, **Bucket Versioning**, and **Tags**.

6. For **Default encryption**, choose **Disable**.

Tip

While this tutorial does not use encryption, you might want to use encryption when analyzing important data. For end-to-end encryption, you can encrypt your data at rest in the bucket and also when you run analysis jobs. For more information about encryption with AWS, see [What is AWS Key Management Service?](#) in the *AWS Key Management Service Developer Guide*.

7. Review your bucket configurations and then choose **Create bucket**.

Create an Amazon S3 Bucket (AWS CLI)

After opening the AWS CLI, you run the `create-bucket` command to create a bucket that will store the input and output data.

To create an Amazon S3 bucket (AWS CLI)

1. To create your bucket, run the following command in the AWS CLI. Replace `DOC-EXAMPLE-BUCKET` with a name for the bucket that is unique in all of AWS.

```
aws s3api create-bucket --bucket DOC-EXAMPLE-BUCKET
```

By default, the `create-bucket` command creates a bucket in the `us-east-1` AWS Region. To create a bucket in an AWS Region other than `us-east-1`, add the `LocationConstraint` parameter to specify your Region. For example, the following command creates a bucket in the `us-west-2` Region.

```
aws s3api create-bucket --bucket DOC-EXAMPLE-BUCKET
--region us-west-2 --create-bucket-configuration LocationConstraint=us-west-2
```

Note that only certain Regions support Amazon Comprehend. For a list of Regions that support Amazon Comprehend, see the [Region Table](#) in the *Global Infrastructure Guide*.

2. To ensure that your bucket was created successfully, run the following command. The command lists all of the S3 buckets associated with your account.

```
aws s3 ls
```

(Console Only) Create Folders

Next, create two folders in your S3 bucket. The first folder is for your input data. The second folder is where Amazon Comprehend sends the analysis results. If you use the Amazon S3 console, you have to manually create the folders. If you use the AWS CLI, you can create folders when you upload the sample dataset or run an analysis job. For that reason, we provide a procedure for creating folders only for console users. If you are using the AWS CLI, you will create folders in [Upload the Input Data \(p. 12\)](#) and in [Step 3: Running Analysis Jobs on Documents in Amazon S3 \(p. 15\)](#).

To create folders in your S3 bucket (console)

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In **Buckets**, choose your bucket from the list of buckets.
3. In the **Overview** tab, choose **Create folder**.
4. For the new folder name, enter `input`.

5. For the encryption settings, choose **None (Use bucket settings)**.
6. Choose **Save**.
7. Repeat steps 3 through 6 to create another folder for the output of the analysis jobs, but in step 4, enter the new folder name **output**.

Upload the Input Data

Now that you have a bucket, upload the sample dataset `amazon-reviews.csv`. You can upload data to S3 buckets with the Amazon S3 console or the AWS CLI.

Upload Sample Documents to a Bucket (Console)

In the Amazon S3 console, upload the sample dataset file to the input folder.

To upload the sample documents (console)

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In **Buckets**, choose your bucket from the list of buckets.
3. Choose the **input** folder and then choose **Upload**.
4. Choose **Add files** and then choose the `amazon-reviews.csv` file on your computer.
5. Choose **Next**.
6. Leave the default settings for **Manage users**, **Access for other AWS accounts**, and **Manage public permissions**. Choose **Next**.
7. For **Storage class**, choose **Standard**.
8. For **Encryption**, choose **None**.
9. Leave **Metadata** and **Tag** blank.
10. Choose **Next** and review the configurations, and then choose **Upload**.

Upload Sample Documents to a Bucket (AWS CLI)

Create an input folder in your S3 bucket and upload the dataset file to the new folder with the `cp` command.

To upload the sample documents (AWS CLI)

1. To upload the `amazon-reviews.csv` file to a new folder in your bucket, run the following AWS CLI command. Replace **DOC-EXAMPLE-BUCKET** with the name of your bucket. By adding the path `/input/` at the end, Amazon S3 automatically creates a new folder called `input` in your bucket and uploads the dataset file to that folder.

```
aws s3 cp amazon-reviews.csv s3://DOC-EXAMPLE-BUCKET/input/
```

2. To ensure that your file was uploaded successfully, run the following command. The command lists the contents of your bucket's `input` folder.

```
aws s3 ls s3://DOC-EXAMPLE-BUCKET/input/
```

Now, you have an S3 bucket with the `amazon-reviews.csv` file in a folder called `input`. If you used the console, you also have an `output` folder in the bucket. If you used the AWS CLI, you will create the `output` folder when running the Amazon Comprehend analysis jobs.

Step 2: (CLI Only) Creating an IAM Role for Amazon Comprehend

This step is necessary only if you are using the AWS Command Line Interface (AWS CLI) to complete this tutorial. If you are using the Amazon Comprehend console to run the analysis jobs, skip to [Step 3: Running Analysis Jobs on Documents in Amazon S3 \(p. 15\)](#).

To run analysis jobs, Amazon Comprehend requires access to the Amazon S3 bucket that contains the sample dataset and will contain the jobs' output. IAM roles allow you to control the permissions of AWS services or users. In this step, you create an IAM role for Amazon Comprehend. Then, you create and attach to this role a resource-based policy that grants Amazon Comprehend access to your S3 bucket. By the end of this step, Amazon Comprehend will have the necessary permissions to access your input data, store your output, and run sentiment and entities analysis jobs.

For more information about using IAM with Amazon Comprehend, see [Overview of Managing Access Permissions to Amazon Comprehend Resources \(p. 191\)](#) and [Using Identity-Based Policies \(IAM Policies\) for Amazon Comprehend \(p. 193\)](#) in the *Amazon Comprehend User Guide*.

Topics

- [Prerequisites \(p. 13\)](#)
- [Create an IAM Role \(p. 13\)](#)
- [Attach an IAM Policy to the IAM Role \(p. 14\)](#)

Prerequisites

Before you begin, do the following:

- Complete [Step 1: Adding Documents to Amazon S3 \(p. 9\)](#).
- Have a code or text editor to save JSON policies and keep track of your Amazon Resource Names (ARNs).

Create an IAM Role

To access your Amazon Simple Storage Service (Amazon S3) bucket, Amazon Comprehend needs to assume an AWS Identity and Access Management (IAM) role. The IAM role declares Amazon Comprehend as a trusted entity. After Amazon Comprehend assumes the role and becomes a trusted entity, you can grant bucket access permissions to Amazon Comprehend. In this step, you create a role that labels Amazon Comprehend as a trusted entity. You can create a role with the IAM console or the AWS CLI. This topic explains how to create a role using the AWS CLI. To use the console, skip to [Step 3: Running Analysis Jobs on Documents in Amazon S3 \(p. 15\)](#).

To create an IAM role for Amazon Comprehend (AWS CLI)

1. Save the following trust policy as a JSON document called `comprehend-trust-policy.json` in a code or text editor on your computer. This trust policy declares Amazon Comprehend as a trusted entity and allows it to assume an IAM role.

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

```
        "Effect": "Allow",
        "Principal": {
            "Service": "comprehend.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
}
```

2. To create the IAM role, run the following AWS CLI command. The command creates an IAM role called `comprehend-access-role` and attaches the trust policy to the role. Replace `path/` with your local computer's path to the JSON document.

```
aws iam create-role --role-name comprehend-access-role
--assume-role-policy-document file://path/comprehend-trust-policy.json
```

Tip

If you get an Error parsing parameter message, the path to your JSON trust policy file is probably incorrect. Provide the relative path to the file based on your home directory.

3. Copy the Amazon Resource Name (ARN) and save it in a text editor. The ARN has a format similar to `arn:aws:iam::123456789012:role/comprehend-access-role`. You need this ARN to run Amazon Comprehend analysis jobs.

Attach an IAM Policy to the IAM Role

To access your Amazon S3 bucket, Amazon Comprehend needs permissions to list, read, and write. To give Amazon Comprehend the required permissions, create and attach an IAM policy to your IAM role. The IAM policy allows Amazon Comprehend to retrieve the input data from your bucket and write analysis results to the bucket. After creating the policy, you attach it to your IAM role.

To create an IAM policy (AWS CLI)

1. Save the following policy locally as a JSON document called `comprehend-access-policy.json`. It grants Amazon Comprehend access to the specified S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:PutObject"
            ]
        }
    ]
}
```

```
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ],
        "Effect": "Allow"
    }
}
```

2. To create the S3 bucket access policy, run the following AWS CLI command. Replace `path/` with your local computer's path to the JSON document.

```
aws iam create-policy --policy-name comprehend-access-policy
--policy-document file://path/comprehend-access-policy.json
```

3. Copy the access policy ARN and save it in a text editor. The ARN has a format similar to `arn:aws:iam::123456789012:policy/comprehend-access-policy`. You need this ARN to attach your access policy to your IAM role.

To attach the IAM policy to your IAM role (AWS CLI)

- Run the following command. Replace `policy-arn` with the access policy ARN that you copied in the previous step.

```
aws iam attach-role-policy --policy-arn policy-arn
--role-name comprehend-access-role
```

You now have an IAM role called `comprehend-access-role` that has a trust policy for Amazon Comprehend and an access policy that grants Amazon Comprehend access to your S3 bucket. You should also have the ARN for the IAM role copied to a text editor.

Step 3: Running Analysis Jobs on Documents in Amazon S3

After storing the data in Amazon S3, you can begin running Amazon Comprehend analysis jobs. A `sentiment` analysis job determines the overall mood of a document (positive, negative, neutral, or mixed). An `entities` analysis job extracts the names of real-world objects from a document. These objects include people, places, titles, events, dates, quantities, products, and organizations. In this step, you run two Amazon Comprehend analysis jobs to extract the sentiment and entities from the sample dataset.

Topics

- [Prerequisites \(p. 15\)](#)
- [Analyze Sentiment and Entities \(p. 16\)](#)

Prerequisites

Before you begin, do the following:

- Complete [Step 1: Adding Documents to Amazon S3 \(p. 9\)](#).
- (Optional) If you are using the AWS CLI, complete [Step 2: \(CLI Only\) Creating an IAM Role for Amazon Comprehend \(p. 13\)](#) and have your IAM role ARN ready.

Analyze Sentiment and Entities

The first job you run analyzes the sentiment of each customer review in the sample dataset. The second job extracts the entities in each customer review. You can perform Amazon Comprehend analysis jobs either using the Amazon Comprehend console or the AWS CLI.

Tip

Make sure that you are in an AWS Region that supports Amazon Comprehend. For more information, see the [Region Table](#) in the *Global Infrastructure Guide*.

Analyze Sentiments and Entities (Console)

When using the Amazon Comprehend console, you create one job at a time. You need to repeat the following steps in order to run both a sentiment and an entities analysis job. Note that for the first job, you create an IAM role, but for the second job, you can reuse the first job's IAM role. You can reuse the IAM role as long as you use the same S3 bucket and folders.

To run sentiment and entities analysis jobs (console)

1. Ensure that you're in the same Region in which you created your Amazon Simple Storage Service (Amazon S3) bucket. If you're in another Region, in the navigation bar, choose the AWS Region where you created your S3 bucket from the **Region selector**.
2. Open the Amazon Comprehend console at <https://console.aws.amazon.com/comprehend/>
3. Choose **Launch Amazon Comprehend**.
4. In the navigation pane, choose **Analysis jobs**.
5. Choose **Create job**.
6. In the **Job settings** section, do the following:
 - a. For **Name**, enter `reviews-sentiment-analysis`.
 - b. For **Analysis type**, choose **Sentiment**.
 - c. For **Language**, choose **English**.
 - d. Leave **Job encryption** turned off.
7. In the **Input data** section, do the following:
 - a. For **Data source**, choose **My documents**.
 - b. For **S3 location**, choose **Browse S3** and then choose your bucket from the list of buckets.
 - c. In your S3 bucket, for **Objects**, choose your input folder.
 - d. In the input folder, choose the sample dataset `amazon-reviews.csv` and then choose **Choose**.
 - e. For **Input format**, choose **One document per line**.
8. In the **Output data** section, do the following:
 - a. For **S3 location**, choose **Browse S3** and then choose your bucket from the list of buckets.
 - b. In your S3 bucket, for **Objects**, choose the output folder and then choose **Choose**.
 - c. Leave **Encryption** turned off.
9. In the **Access permissions** section, do the following:
 - a. For **IAM role**, choose **Create an IAM role**.
 - b. For **Permissions to access**, choose **Input and Output S3 buckets**.
 - c. For **Name suffix**, enter `comprehend-access-role`. This role provides access to your Amazon S3 bucket.
10. Choose **Create job**.
11. Repeat steps 1-10 to create an entities analysis job. Make the following changes:

- a. In **Job settings**, for **Name**, enter reviews-entities-analysis.
- b. In **Job settings**, for **Analysis type**, choose **Entities**.
- c. In **Access permissions**, choose **Use an existing IAM role**. For **Role name**, choose AmazonComprehendServiceRole-comprehend-access-role (this is the same role you created for the sentiment job).

Analyze Sentiments and Entities (AWS CLI)

You use the `start-sentiment-detection-job` and the `start-entities-detection-job` commands to run sentiment and entities analysis jobs. After you run each command, the AWS CLI shows a JSON object with a `JobId` value that allows you to access details about the job, including the output S3 location.

To run sentiment and entities analysis jobs (AWS CLI)

1. Start a sentiment analysis job by running the following command in the AWS CLI. Replace `arn:aws:iam::123456789012:role/comprehend-access-role` with the IAM role ARN that you previously copied to a text editor. If your default AWS CLI Region differs from the Region in which you created your Amazon S3 bucket, include the `--region` parameter and replace `us-east-1` with the Region where your bucket resides.

```
aws comprehend start-sentiment-detection-job
--input-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/input/
--output-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/output/
--data-access-role-arn arn:aws:iam::123456789012:role/comprehend-access-role
--job-name reviews-sentiment-analysis
--language-code en
[--region us-east-1]
```

2. After you submit the job, copy the `JobId` and save it to a text editor. You will need the `JobId` to find the output files from the analysis job.
3. Start an entities analysis job by running the following command.

```
aws comprehend start-entities-detection-job
--input-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/input/
--output-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/output/
--data-access-role-arn arn:aws:iam::123456789012:role/comprehend-access-role
--job-name reviews-entities-analysis
--language-code en
[--region us-east-1]
```

4. After you submit the job, copy the `JobId` and save it to a text editor.
5. Check the status of your jobs. You can view the progress of a job by tracking its `JobId`.

To track the progress of your sentiment analysis job, run the following command. Replace `sentiment-job-id` with the `JobId` that you copied after running your sentiment analysis.

```
aws comprehend describe-sentiment-detection-job
--job-id sentiment-job-id
```

To track your entities analysis job, run the following command. Replace `entities-job-id` with the `JobId` that you copied after running your entities analysis.

```
aws comprehend describe-entities-detection-job
--job-id entities-job-id
```

It takes several minutes for the JobStatus to show as COMPLETED.

You have completed sentiment and entities analysis jobs. Both of the jobs should be completed before you move on to the next step. It can take several minutes for the jobs to finish.

Step 4: Preparing the Amazon Comprehend Output for Data Visualization

To prepare the results of the sentiment and entities analysis jobs for creating data visualizations, you use AWS Glue and Amazon Athena. In this step, you extract the Amazon Comprehend results files. Then, you create an AWS Glue *crawler* that explores your data and automatically catalogs it in tables in the AWS Glue Data Catalog. Finally, you access and transform these tables using Amazon Athena, a serverless and interactive query service. When you have finished this step, your Amazon Comprehend results are cleaned and ready for visualization.

Topics

- [Prerequisites \(p. 18\)](#)
- [Download the Output \(p. 18\)](#)
- [Extract the Output Files \(p. 19\)](#)
- [Upload the Extracted Files \(p. 20\)](#)
- [Load the Data into an AWS Glue Data Catalog \(p. 21\)](#)
- [Prepare the Data for Analysis \(p. 24\)](#)

Prerequisites

Before you begin, complete [Step 3: Running Analysis Jobs on Documents in Amazon S3 \(p. 15\)](#).

Download the Output

The Amazon Comprehend uses Gzip compression to compress output files and save them as a tar archive. The simplest way to extract the output files is to download the `output.tar.gz` archives locally. In this step, you download the sentiment and entities output archives.

Download the Output Files (Console)

To find the output files for each job, return to the analysis job in the Amazon Comprehend console. The analysis job provides the S3 location for the output, where you can download the output file.

To download the output files (console)

1. In the [Amazon Comprehend console](#), in the navigation pane, return to **Analysis jobs**.
2. Choose your sentiment analysis job `reviews-sentiment-analysis`.
3. Under **Output**, choose the link displayed next to **Output data location**. This redirects you to the `output.tar.gz` archive in your S3 bucket.
4. In the **Overview** tab, choose **Download**.
5. On your computer, rename the archive as `sentiment-output.tar.gz`. Since all of the output files have the same name, this helps you keep track of the sentiment and entities files.
6. Repeat steps 1-4 to find and download the output from your `reviews-entities-analysis` job. On your computer, rename the archive as `entities-output.tar.gz`.

Download the Output Files (AWS CLI)

To find the output files for each job, use the `JobID` from the analysis job to find the output's S3 location. Then, use the `cp` command to download the output file to your computer.

To download the output files (AWS CLI)

1. To list details about your sentiment analysis job, run the following command. Replace `sentiment-job-id` with the sentiment `JobID` that you saved.

```
aws comprehend describe-sentiment-detection-job --job-id sentiment-job-id
```

If you lost track of your `JobID`, you can run the following command to list all of your sentiment jobs and filter for your job by name.

```
aws comprehend list-sentiment-detection-jobs  
--filter JobName="reviews-sentiment-analysis"
```

2. In the `OutputDataConfig` object, find the `S3Uri` value. The `S3Uri` value should be similar to the following format: `s3://DOC-EXAMPLE-BUCKET/.../output/output.tar.gz`. Copy this value to a text editor.
3. To download the sentiment output archive to your local directory, run the following command. Replace the S3 bucket path with the `S3Uri` you copied in the previous step. Replace `path/` with the folder path to your local directory. The name `sentiment-output.tar.gz` replaces the original archive name to help you keep track of the sentiment and entities files.

```
aws s3 cp s3://DOC-EXAMPLE-BUCKET/.../output/output.tar.gz  
path/sentiment-output.tar.gz
```

4. To list details about your entities analysis job, run the following command.

```
aws comprehend describe-entities-detection-job  
--job-id entities-job-id
```

If you don't know your `JobID`, run the following command to list all of your entities jobs and filter for your job by name.

```
aws comprehend list-entities-detection-jobs  
--filter JobName="reviews-entities-analysis"
```

5. From the `OutputDataConfig` object in your entities job description, copy the `S3Uri` value.
6. To download the entities output archive to your local directory, run the following command. Replace the S3 bucket path with the `S3Uri` you copied in the previous step. Replace `path/` with the folder path to your local directory. The name `entities-output.tar.gz` replaces the original archive name.

```
aws s3 cp s3://DOC-EXAMPLE-BUCKET/.../output/output.tar.gz  
path/entities-output.tar.gz
```

Extract the Output Files

Before you can access the Amazon Comprehend results, unpack the sentiment and entities archives. You can use either your local file system or a terminal to unpack the archives.

Extract the Output Files (GUI File System)

If you use macOS, double-click the archive in your GUI file system to extract the output file from the archive.

If you use Windows, you can use a third-party tool such as 7-Zip to extract the output files in your GUI file system. In Windows, you must perform two steps to access the output file in the archive. First decompress the archive, and then extract the archive.

Rename the sentiment file as `sentiment-output` and the entities file as `entities-output` to distinguish between the output files.

Extract the Output Files (Terminal)

If you use Linux or macOS, you can use your standard terminal. If you use Windows, you must have access to a Unix-style environment, such as Cygwin, to run tar commands.

To extract the sentiment output file from the sentiment archive, run the following command in your local terminal.

```
tar -xvf sentiment-output.tar.gz --transform 's,^,sentiment-,'
```

Note that the `--transform` parameter adds the prefix `sentiment-` to the output file inside of the archive, renaming the file as `sentiment-output`. This allows you to distinguish between the sentiment and entities output files and prevent overwriting.

To extract the entities output file from the entities archive, run the following command in your local terminal.

```
tar -xvf entities-output.tar.gz --transform 's,^,entities-,'
```

The `--transform` parameter adds the prefix `entities-` to the output file name.

Tip

To save storage costs in Amazon S3, you can compress the files again with Gzip before uploading them. It's important to decompress and unpack the original archives because AWS Glue can't automatically read data from a tar archive. However, AWS Glue can read from files in Gzip format.

Upload the Extracted Files

After extracting the files, upload them to your bucket. You must store the sentiment and entities output files in separate folders in order for AWS Glue to read the data properly. In your bucket, create a folder for the extracted sentiment results and a second folder for the extracted entities results. You can create folders either with the Amazon S3 console or the AWS CLI.

Upload the Extracted Files to Amazon S3 (Console)

In your S3 bucket, create one folder for the extracted sentiment results file and one folder for the entities results file. Then, upload the extracted results files to their respective folders.

To upload the extracted files to Amazon S3 (console)

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In **Buckets**, choose your bucket and then choose **Create folder**.
3. For the new folder name, enter `sentiment-results` and choose **Save**. This folder will contain the extracted sentiment output file.

4. In your bucket's **Overview** tab, from the list of bucket contents, choose the new folder **sentiment-results**. Choose **Upload**.
5. Choose **Add files**, choose the **sentiment-output** file from your local computer, and then choose **Next**.
6. Leave the options for **Manage users**, **Access for other AWS account**, and **Manage public permissions** as the defaults. Choose **Next**.
7. For **Storage class**, choose **Standard**. Leave the options for **Encryption**, **Metadata**, and **Tag** as the defaults. Choose **Next**.
8. Review the upload options and then choose **Upload**.
9. Repeat steps 1-8 to create a folder called **entities-results**, and upload the **entities-output** file to it.

Upload the Extracted Files to Amazon S3 (AWS CLI)

You can create a folder in your S3 bucket while uploading a file with the `cp` command.

To upload the extracted files to Amazon S3 (AWS CLI)

1. Create a sentiment folder and upload your sentiment file to it by running the following command. Replace `path/` with the local path to your extracted sentiment output file.

```
aws s3 cp path/sentiment-output s3://DOC-EXAMPLE-BUCKET/sentiment-results/
```

2. Create an entities output folder and upload your entities file to it by running the following command. Replace `path/` with the local path to your extracted entities output file.

```
aws s3 cp path/entities-output s3://DOC-EXAMPLE-BUCKET/entities-results/
```

Load the Data into an AWS Glue Data Catalog

To get the results into a database, you can use an AWS Glue *crawler*. An AWS Glue *crawler* scans files and discovers the schema of the data. It then arranges the data in tables in an AWS Glue Data Catalog (a serverless database). You can create a crawler with the AWS Glue console or the AWS CLI.

Load the Data into an AWS Glue Data Catalog (Console)

Create an AWS Glue crawler that scans your **sentiment-results** and **entities-results** folders separately. A new IAM role for AWS Glue gives the crawler permission to access your S3 bucket. You create this IAM role while setting up the crawler.

To load the data into an AWS Glue Data Catalog (console)

1. Ensure that you're in a Region which supports AWS Glue. If you're in another Region, in the navigation bar, choose a supported Region from the **Region selector**. For a list of Regions that support AWS Glue, see the [Region Table](#) in the [Global Infrastructure Guide](#).
2. Open the AWS Glue console at <https://console.aws.amazon.com/glue/>.
3. In the navigation pane, choose **Crawlers** and then choose **Add crawler**.
4. For **Crawler name**, enter `comprehend-analysis-crawler` and then choose **Next**.
5. For **Crawler source type**, choose **Data stores** and then choose **Next**.
6. For **Add a data store**, do the following:
 - a. For **Choose a data store**, choose **S3**.

- b. Leave **Connection** blank.
 - c. For **Crawl data in**, choose **Specified path in my account**.
 - d. For **Include path**, enter the full S3 path of the sentiment output folder: `s3://DOC-EXAMPLE-
BUCKET/sentiment-results/`.
 - e. Choose **Next**.
7. For **Add another data store**, choose **Yes** and then choose **Next**. Repeat Step 6, but enter the full S3 path of the entities output folder: `s3://DOC-EXAMPLE-
BUCKET/entities-results`.
 8. For **Add another data store**, choose **No** and then choose **Next**.
 9. For **Choose an IAM role**, do the following:
 - a. Choose **Create an IAM role**.
 - b. For **IAM role**, enter `glue-access-role` and then choose **Next**.
 10. For **Create a schedule for this crawler**, choose **Run on demand** and choose **Next**.
 11. For **Configure the crawler's output**, do the following:
 - a. For **Database**, choose **Add database**.
 - b. For **Database name**, enter `comprehend-results`. This database will store your Amazon Comprehend output tables.
 - c. Leave the other options on their default settings and choose **Next**.
 12. Review the crawler information and then choose **Finish**.
 13. In the Glue console, in **Crawlers**, choose `comprehend-analysis-crawler` and choose **Run crawler**. It can take a few minutes for the crawler to finish.

Load the Data into an AWS Glue Data Catalog (AWS CLI)

Create an IAM role for AWS Glue that provides permission to access your S3 bucket. Then, create a database in the AWS Glue Data Catalog. Finally, create and run a crawler that loads your data into tables in the database.

To load the data into an AWS Glue Data Catalog (AWS CLI)

1. To create an IAM role for AWS Glue, do the following:
 - a. Save the following trust policy as a JSON document called `glue-trust-policy.json` on your computer.

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

- b. To create an IAM role, run the following command. Replace `path/` with your local computer's path to the JSON document.

```
aws iam create-role --role-name glue-access-role  
--assume-role-policy-document file://path/glue-trust-policy.json
```

- c. When the AWS CLI lists the Amazon Resource Number (ARN) for the new role, copy and save it to a text editor.
- d. Save the following IAM policy as a JSON document called `glue-access-policy.json` on your computer. The policy grants AWS Glue permission to crawl your results folders.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/sentiment-results*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/entities-results*"
            ]
        }
    ]
}
```

- e. To create the IAM policy, run the following command. Replace `path/` with your local computer's path to the JSON document.

```
aws iam create-policy --policy-name glue-access-policy
--policy-document file://path/glue-access-policy.json
```

- f. When the AWS CLI lists the access policy's ARN, copy and save it to a text editor.
- g. Attach the new policy to the IAM role by running the following command. Replace `policy-arn` with the IAM policy ARN you copied in the previous step.

```
aws iam attach-role-policy --policy-arn policy-arn
--role-name glue-access-role
```

- h. Attach the AWS managed policy `AWSGlueServiceRole` to your IAM role by running the following command.

```
aws iam attach-role-policy --policy-arn
arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
--role-name glue-access-role
```

2. Create an AWS Glue database by running the following command.

```
aws glue create-database
--database-input Name="comprehend-results"
```

3. Create a new AWS Glue crawler by running the following command. Replace `glue-iam-role-arn` with the ARN of your AWS Glue IAM role.

```
aws glue create-crawler
--name comprehend-analysis-crawler
--role glue-iam-role-arn
--targets S3Targets=[
{Path="s3://DOC-EXAMPLE-BUCKET/sentiment-results"},
{Path="s3://DOC-EXAMPLE-BUCKET/entities-results"}]
--database-name comprehend-results
```

4. Start the crawler by running the following command.

```
aws glue start-crawler --name comprehend-analysis-crawler
```

It can take a few minutes for the crawler to finish.

Prepare the Data for Analysis

Now you have a database populated with the Amazon Comprehend results. However, the results are nested. To unnest them, you run a few SQL statements in Amazon Athena. Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage and it has a pay-per-query pricing model. In this step, you create new tables of cleaned data that you can use for analysis and visualization. You use the Athena console to prepare the data.

To prepare the data

1. Open the Athena console at <https://console.aws.amazon.com/athena/>.
2. In the navigation bar, choose **Settings**.
3. For **Query result location**, enter `s3://DOC-EXAMPLE-BUCKET/query-results/`. This creates a new folder called `query-results` in your bucket that stores the output of the Amazon Athena queries you run. Choose **Save**.
4. In the navigation bar, choose **Query editor**.
5. For **Database**, choose the AWS Glue database `comprehend-results` that you created.
6. In the **Tables** section, you should have two tables called `sentiment_results` and `entities_results`. Preview the tables to make sure that the crawler loaded the data. In each table's options, choose **Preview table**. A short query runs automatically. Check the **Results** pane to ensure that the tables contain data.

Tip

If the tables don't have any data, try checking the folders in your S3 bucket. Make sure that there is one folder for entities results and one folder for sentiment results. Then, try running a new AWS Glue crawler.

7. To unnest the `sentiment_results` table, enter the following query in the **Query editor** and choose **Run query**.

```
CREATE TABLE sentiment_results_final AS
SELECT file, line, sentiment,
sentimentscore.mixed AS mixed,
sentimentscore.negative AS negative,
sentimentscore.neutral AS neutral,
sentimentscore.positive AS positive
FROM sentiment_results
```

8. To begin unnesting the `entities` table, enter the following query in the **Query editor** and choose **Run query**.

```
CREATE TABLE entities_results_1 AS
SELECT file, line, nested FROM entities_results
CROSS JOIN UNNEST(entities) as t(nested)
```

9. To finish unnesting the `entities` table, enter the following query in the **Query editor** and choose **Run query**.

```
CREATE TABLE entities_results_final AS
SELECT file, line,
```

```

nested.beginoffset AS beginoffset,
nested.endoffset AS endoffset,
nested.score AS score,
nested.text AS entity,
nested.type AS category
FROM entities_results_1

```

Your **sentiment_results_final** table should look like the following, with columns named **file**, **line**, **sentiment**, **mixed**, **negative**, **neutral**, and **positive**. The table should have one value per cell. The **sentiment** column describes the most likely overall sentiment of a particular review. The **mixed**, **negative**, **neutral**, and **positive** columns give scores for each type of sentiment.

Results						
file	line	sentiment	mixed	negative	neutral	positive
1	amazon-reviews.csv	6	MIXED	0.9862896203994751	0.0015502438182011247	1.6660270921420306E-4
2	amazon-reviews.csv	8	POSITIVE	0.0012987082591280341	0.01186690479516983	0.174478679895401
3	amazon-reviews.csv	11	POSITIVE	6.5368581090297084E-6	0.0013866390800103545	0.007405391428619623
4	amazon-reviews.csv	13	POSITIVE	4.7155481297522783E-4	0.24615342915058136	0.017713148146867752
5	amazon-reviews.csv	14	POSITIVE	1.5821871784282848E-5	0.06828905642032623	0.014075091108679771
6	amazon-reviews.csv	16	MIXED	0.9864791035652161	8.548551704734564E-4	1.0789262159960344E-4
7	amazon-reviews.csv	20	NEGATIVE	1.1621621524682269E-4	0.9815887212753296	0.004688907880336046
8	amazon-reviews.csv	21	POSITIVE	4.663573781726882E-5	0.009533549658954144	0.0015825830632820725
9	amazon-reviews.csv	23	POSITIVE	1.7699007003102452E-4	0.40269607305526733	0.0018250439316034317
10	amazon-reviews.csv	25	POSITIVE	1.8434448065818287E-6	1.15832663141191E-4	0.0010993879986926913

Your **entities_results_final** table should look like the following, with columns named **file**, **line**, **beginoffset**, **endoffset**, **score**, **entity**, and **category**. The table should have one value per cell. The **score** column indicates Amazon Comprehend's confidence in the **entity** it detected. The **category** indicates what kind of entity Comprehend detected.

Results						
file	line	beginoffset	endoffset	score	entity	category
1	amazon-reviews.csv	0	15	22	0.9885989378545348	English
2	amazon-reviews.csv	2	24	28	0.9699371997593782	2 me
3	amazon-reviews.csv	2	94	95	0.6523066984191679	2
4	amazon-reviews.csv	2	125	126	0.713791396412543	2
5	amazon-reviews.csv	4	30	36	0.9957169942979278	kindle
6	amazon-reviews.csv	5	1	10	0.9979111763962706	Hawthorne
7	amazon-reviews.csv	5	135	142	0.5065408081314243	Puritan
8	amazon-reviews.csv	5	143	148	0.7702269458801602	Salem
9	amazon-reviews.csv	5	211	229	0.999675563687763	The Scarlet Letter
10	amazon-reviews.csv	5	233	236	0.8944631322676461	one

Now that you have the Amazon Comprehend results loaded into tables, you can visualize and extract meaningful insights from the data.

Step 5: Visualizing Amazon Comprehend Output in Amazon QuickSight

After storing the Amazon Comprehend results in tables, you can connect to and visualize the data with Amazon QuickSight. Amazon QuickSight is an AWS managed business intelligence (BI) tool for visualizing data. Amazon QuickSight makes it easy to connect to your data source and create powerful visuals. In this step, you connect Amazon QuickSight to your data, create visualizations that extract insights from the data, and publish a dashboard of visualizations.

Topics

- [Prerequisites \(p. 26\)](#)
- [Give Amazon QuickSight Access \(p. 26\)](#)
- [Import the Datasets \(p. 27\)](#)
- [Create a Sentiment Visualization \(p. 27\)](#)
- [Create an Entities Visualization \(p. 28\)](#)
- [Publish a Dashboard \(p. 29\)](#)
- [Clean Up \(p. 30\)](#)

Prerequisites

Before you begin, complete [Step 4: Preparing the Amazon Comprehend Output for Data Visualization \(p. 18\)](#).

Give Amazon QuickSight Access

To import the data, Amazon QuickSight requires access to your Amazon Simple Storage Service (Amazon S3) bucket and Amazon Athena tables. To give Amazon QuickSight access to your data, you must be signed in as a QuickSight administrator and have access to edit the resource permissions. If you are unable to complete the following steps, review the IAM prerequisites from the overview page [Tutorial: Analyzing Insights from Customer Reviews with Amazon Comprehend \(p. 7\)](#).

To give Amazon QuickSight access to your data

1. Open the [Amazon QuickSight console](#).
2. If this is the first time you have used Amazon QuickSight, the console prompts you to create a new administrator user by providing an email address. For **Email address**, enter the same email address as your AWS account. Choose **Continue**.
3. After signing in, choose your profile name in the navigation bar and choose **Manage QuickSight**. You must be signed in as an administrator to view the **Manage QuickSight** option.
4. Choose **Security and permissions**.
5. For **QuickSight access to AWS services**, choose **Add or remove**.
6. Choose **Amazon S3**.
7. From **Select Amazon S3 buckets**, choose your S3 bucket for both **S3 Bucket** and **Write permissions for Athena Workgroup**.
8. Choose **Finish**.
9. Choose **Update**.

Import the Datasets

Before creating visualizations, you must add the sentiment and entities datasets to Amazon QuickSight. You do this with the Amazon QuickSight console. You import your unnested sentiment and unnested entities tables from Amazon Athena.

To import your datasets

1. Open the [Amazon QuickSight console](#).
2. In the navigation bar, in **Datasets**, choose **New dataset**.
3. For **Create a Data Set**, choose **Athena**.
4. For **Data source name**, enter `reviews-sentiment-analysis` and choose **Create data source**.
5. For **Database**, choose the database `comprehend-results`.
6. For **Tables**, choose the sentiment table `sentiment_results_final` and then choose **Select**.
7. Choose **Import to SPICE for quicker analytics** and choose **Visualize**. SPICE is QuickSight's in-memory calculation engine that provides faster analyses than direct querying when creating visualizations.
8. Return to the Amazon QuickSight console and choose **Datasets**. Repeat steps 1-7 to create an entities dataset, but make the following changes:
 - a. For **Data source name**, enter `reviews-entities-analysis`.
 - b. For **Tables**, choose the entities table `entities_results_final`.

Create a Sentiment Visualization

Now that you can access your data in Amazon QuickSight, you can begin creating visualizations. You create a pie chart with the Amazon Comprehend sentiment data. The pie chart shows what proportion of the reviews are positive, neutral, mixed, and negative.

To visualize sentiment data

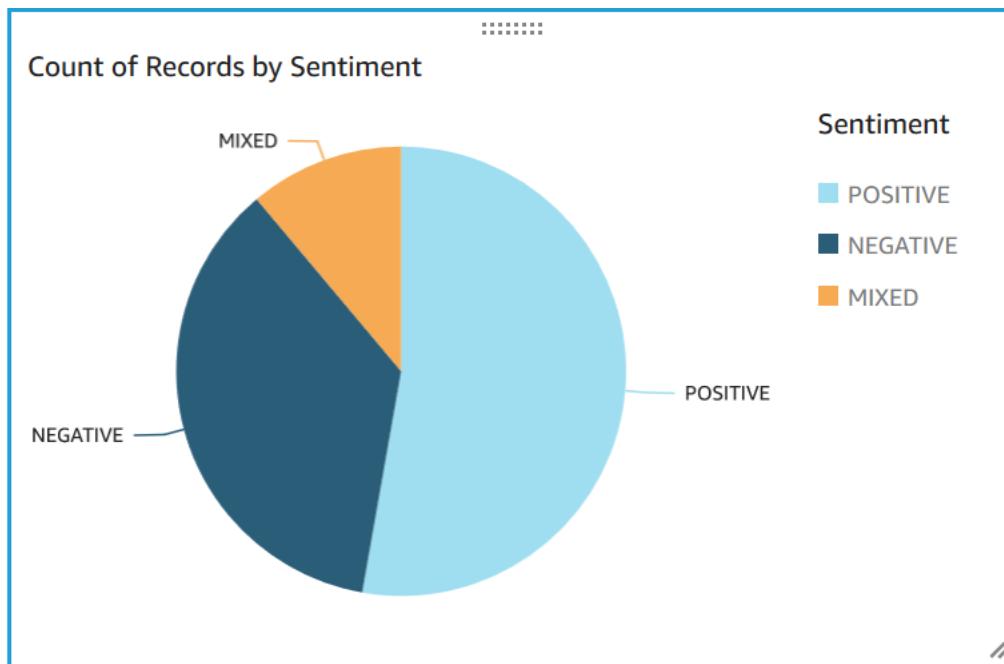
1. In the Amazon QuickSight console, choose **Analyses** and then choose **New analysis**.
2. From **Your Data Sets**, choose the sentiment dataset `sentiment_results_final` and then choose **Create analysis**.
3. In the visual editor, in **Fields list**, choose **sentiment**.

Note

The values in the **Fields list** depend on the column names you used to create the tables in Amazon Athena. If you changed the provided column names in the SQL queries, the **Fields list** names will be different than the names used in these visualization examples.

4. For **Visual types**, choose **Pie chart**.

A pie chart similar to the following with positive, neutral, mixed, and negative sections is displayed. To see the count and percentage of a section, hover over it.



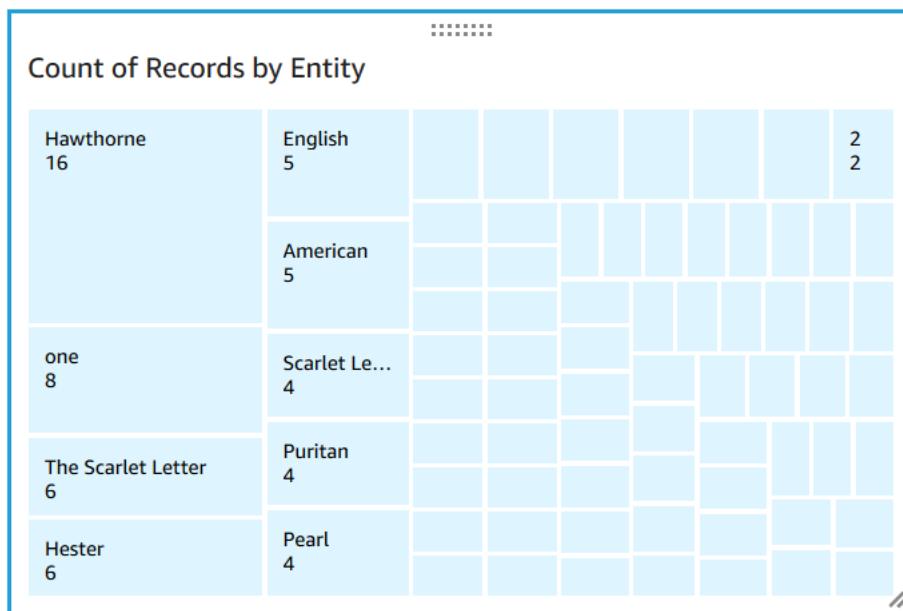
Create an Entities Visualization

Now create a second visualization with the entities dataset. You create a tree map of the distinct entities in the data. Each block in the tree map represents an entity, and the size of the block correlates to the number of times that the entity appears in the dataset.

To visualize entities data

1. In the **Visualize** control pane, next to **Data set**, choose the **Add, edit, replace, and remove data sets** icon.
2. Choose **Add data set**.
3. For **Choose data set to add**, choose your entities dataset `entities_results_final` from the list of datasets and choose **Select**.
4. In the **Visualize** control pane, choose the **Data set** drop down menu and choose the entities dataset `entities_results_final`.
5. In **Fields list**, choose **entity**.
6. For **Visual types**, choose **Tree map**.

A tree map similar to the following is displayed next to your pie chart. To see the count of a specific entity, hover over a block.



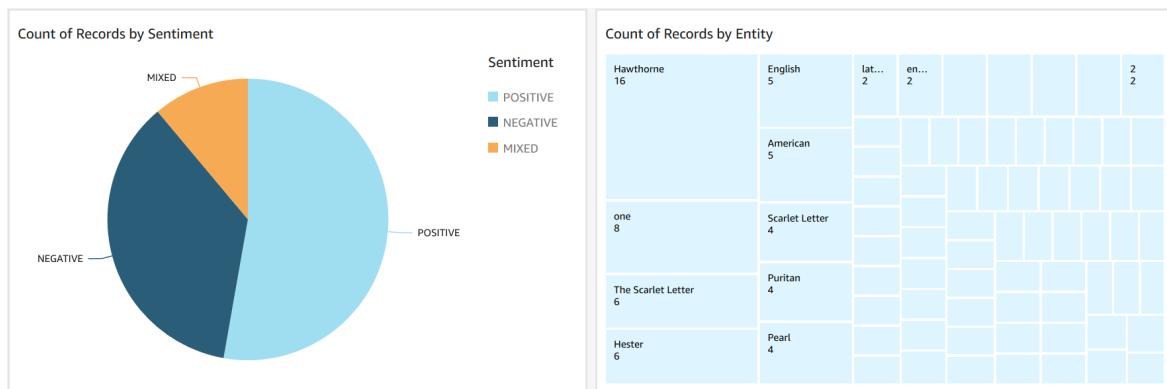
Publish a Dashboard

After creating the visualizations, you can publish them as a dashboard. You can perform various tasks with a dashboard, such as sharing it with users in your AWS account, saving it as a PDF, or emailing it as a report (limited to the Enterprise edition of Amazon QuickSight). In this step, you publish the visualizations as a dashboard in your account.

To publish your dashboard

1. In the navigation bar, choose **Share**.
2. Choose **Publish dashboard**.
3. Choose **Publish new dashboard as** and enter the name `comprehend-analysis-reviews` for the dashboard.
4. Choose **Publish dashboard**.
5. Close the **Share dashboard with users** pane by choosing the close button in the upper-right corner.
6. In the Amazon QuickSight console, in the navigation pane, choose **Dashboards**. A thumbnail of your new dashboard `comprehend-analysis-reviews` should appear under **Dashboards**. Choose the dashboard to view it.

You now have a dashboard with sentiment and entities visualizations that looks similar to the following example.



Tip

If you want to edit the visualizations in your dashboard, return to [Analyses](#) and edit the visualization that you want to update. Then, publish the dashboard again either as a new dashboard or as a replacement of the existing dashboard.

Clean Up

After completing this tutorial, you might want to clean up any AWS resources you no longer want to use. Active AWS resources can continue to incur charges in your account.

The following actions can help prevent incurring ongoing charges:

- Cancel your Amazon QuickSight subscription. Amazon QuickSight is a monthly subscription service. To cancel your subscription, see [Canceling your Subscription](#) in the *Amazon QuickSight User Guide*.
- Delete your Amazon S3 bucket. Amazon S3 charges you for storage. To clean up your Amazon S3 resources, delete your bucket. For information about deleting a bucket, see [How do I delete an S3 Bucket?](#) in the *Amazon Simple Storage Service User Guide*. Make sure that you save all of your important files before deleting your bucket.
- Clear your AWS Glue Data Catalog. The AWS Glue Data Catalog charges you monthly for storage. You can delete your databases to prevent incurring ongoing charges. For information about managing your AWS Glue Data Catalog databases, see [Working with Databases on the AWS Glue Console](#) in the *AWS Glue Developer Guide*. Make sure that you export your data before clearing any databases or tables.

Getting Started with Amazon Comprehend

To get started using Amazon Comprehend, set up an AWS account and create an AWS Identity and Access Management (IAM) user. To use the Amazon Comprehend (AWS CLI), download and configure it.

Topics

- [Step 1: Set Up an AWS Account and Create an Administrator User \(p. 31\)](#)
- [Step 2: Set Up the AWS Command Line Interface \(AWS CLI\) \(p. 32\)](#)
- [Step 3: Getting Started Using the Amazon Comprehend Console \(p. 33\)](#)
- [Step 4: Getting Started Using the Amazon Comprehend API \(p. 54\)](#)
- [Solution: Analyzing Text with Amazon Comprehend and Amazon Elasticsearch Service \(p. 89\)](#)

Step 1: Set Up an AWS Account and Create an Administrator User

Before you use Amazon Comprehend for the first time, complete the following tasks:

1. [Sign Up for AWS \(p. 31\)](#)
2. [Create an IAM User \(p. 31\)](#)

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all AWS services, including Amazon Comprehend. You are charged only for the services that you use.

With Amazon Comprehend, you pay only for the resources that you use. If you are a new AWS customer, you can get started with Amazon Comprehend for free. For more information, see [AWS Free Usage Tier](#).

If you already have an AWS account, skip to the next section.

To create an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Record your AWS account ID because you'll need it for the next task.

Create an IAM User

Services in AWS, such as Amazon Comprehend, require that you provide credentials when you access them. This allows the service to determine whether you have permissions to access the service's resources.

We strongly recommend that you access AWS using AWS Identity and Access Management (IAM), not the credentials for your AWS account. To use IAM to access AWS, create an IAM user, add the user to an IAM group with administrative permissions, and then grant administrative permissions to the IAM user. You can then access AWS using a special URL and the IAM user's credentials.

The Getting Started exercises in this guide assume that you have a user with administrator privileges, `adminuser`.

To create an administrator user and sign in to the console

1. Create an administrator user called `adminuser` in your AWS account. For instructions, see [Creating Your First IAM User and Administrators Group](#) in the *IAM User Guide*.
2. Sign in to the AWS Management Console using a special URL. For more information, see [How Users Sign In to Your Account](#) in the *IAM User Guide*.

For more information about IAM, see the following:

- [AWS Identity and Access Management \(IAM\)](#)
- [Getting started](#)
- [IAM User Guide](#)

Next Step

[Step 2: Set Up the AWS Command Line Interface \(AWS CLI\) \(p. 32\)](#)

Step 2: Set Up the AWS Command Line Interface (AWS CLI)

You don't need the AWS CLI to perform the steps in the Getting Started exercises. However, some of the other exercises in this guide do require it. If you prefer, you can skip this step and go to [Step 3: Getting Started Using the Amazon Comprehend Console \(p. 33\)](#), and set up the AWS CLI later.

To set up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:
 - [Getting Set Up with the AWS Command Line Interface](#)
 - [Configuring the AWS Command Line Interface](#)
2. In the AWS CLI config file, add a named profile for the administrator user:

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

You use this profile when executing the AWS CLI commands. For more information about named profiles, see [Named Profiles](#) in the *AWS Command Line Interface User Guide*. For a list of AWS Regions, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

3. Verify the setup by typing the following help command at the command prompt:

```
aws help
```

Next Step

[Step 3: Getting Started Using the Amazon Comprehend Console \(p. 33\)](#)

Step 3: Getting Started Using the Amazon Comprehend Console

The easiest way to get started using Amazon Comprehend is to use the console to analyze a short text file. If you haven't reviewed the concepts and terminology in [How It Works \(p. 4\)](#), we recommend that you do that before proceeding.

Topics

- [Analyzing Documents Using the Console \(p. 33\)](#)
- [Creating and Using Custom Entity Recognizer \(p. 40\)](#)
- [Creating and Using Custom Classifiers \(p. 44\)](#)
- [Model Versioning with Amazon Comprehend \(p. 49\)](#)
- [Creating a Topic Modeling Job Using the Console \(p. 51\)](#)
- [Creating an Events Detection Job Using the Console \(p. 53\)](#)

Analyzing Documents Using the Console

The Amazon Comprehend console enables you to analyze the contents of documents up to 5,000 characters long. The results are shown in the console so that you can review the analysis.

To start analyzing documents, sign in to the AWS Management Console and open the [Amazon Comprehend console](#).

You can replace the sample text with your own text either in English or one of the other languages supported by Amazon Comprehend and then choose **Analyze** to get an analysis of your text. Below the text being analyzed, the **Results** pane shows more information about the text.

Entities

The **Entities** tab lists each entity, its category, and the level of confidence that Amazon Comprehend has detected in the input text. The results are color-coded to indicate different entity types such as organizations, locations, dates, and persons. For more information, see [Detect Entities \(p. 94\)](#).

The screenshot shows the 'Entities' tab selected in the 'Insights' interface. The analyzed text is a credit card statement. The results table lists entities with their types and confidence levels:

Entity	Type	Confidence
Zhang Wei	Person	0.99+
AnyCompany Financial Services, LLC	Organization	0.99+
1111-0000-1111-0000	Other	0.87
\$24.53	Quantity	0.99+
July 31st	Date	0.99+
XXXXXX1111	Other	0.85
XXXXX0000	Other	0.83
100 Main Street, Anytown, WA 98121	Location	0.99+
206-555-0100	Other	0.99+
AnyCompany	Organization	0.97

Key phrases

The **Key phrases** tab lists key noun phrases that Amazon Comprehend detected in the input text and the associated confidence level. For more information, see [Detect Key Phrases \(p. 101\)](#).

The screenshot shows the 'Key phrases' tab selected in the 'Insights' interface. Below the tab, the analyzed text is displayed, containing several underlined words and phrases. A search bar and navigation controls are visible above the results table. The results table lists key phrases and their confidence levels.

Key phrases	Confidence
Zhang Wei	0.83
Your AnyCompany Financial Services	0.99+
LLC credit card	0.99+
1111-0000-1111-0000	0.72
a minimum payment	0.99+
\$24.53	0.99+
July 31st	0.99+
your autopay settings	0.99+
your payment	0.99+
the due date	0.99+

Language

The **Language** tab shows the dominant language of the text and Amazon Comprehend's level of confidence that it has detected the dominant language correctly. Amazon Comprehend can recognize 100 languages. For more information, see [Detect the Dominant Language \(p. 102\)](#).

The screenshot shows the 'Language' tab selected in the Insights console. The analyzed text is a credit card statement. The Language section shows 'English, en' with 0.98 confidence.

PII

The **PII** tab lists entities in your input text that contain personally identifiable information (PII). A PII entity is a textual reference to personal data that could be used to identify an individual, such as an address, bank account number, or phone number. For more information, see [Detect Personally Identifiable Information \(PII\) \(p. 106\)](#).

The **PII** tab provides two analysis modes:

- Offsets
- Labels

Offsets

The **Offsets** analysis mode identifies the location of PII in your text documents. For more information, see [Locate PII Entities \(p. 108\)](#).

Insights Info

Entities | Key phrases | Language | **PII** | Sentiment | Syntax

Personally identifiable information (PII) analysis mode

Offsets
Identify the location of PII in your text documents.

Labels
Label text documents with PII.

Analyzed text

Hello Zhang Wei. Your AnyCompany Financial Services, LLC credit card account 1111-0000-1111-0000 has a minimum payment of \$24.53 that is due by July 31st. Based on your autopay settings, we will withdraw your payment on the due date from your bank account XXXXXX1111 with the routing number XXXXX0000.

Your latest statement was mailed to 100 Main Street, Anytown, WA 98121. After your payment is received, you will receive a confirmation text message at 206-555-0100. If you have questions about your bill, AnyCompany Customer Service is available by phone at 206-555-0199 or email at support@anycompany.com.

▼ Results

Entity	Type	Confidence
Zhang Wei	Name	0.99+
1111-0000-1111-0000	Bank account number	0.93
July 31st	Date time	0.99+
XXXXXX1111	Bank account number	0.99+
XXXXX0000	Bank routing	0.99+
100 Main Street, Anytown, WA 98121	Address	0.99+
206-555-0100	Phone	0.99+
206-555-0199	Phone	0.99+
support@anycompany.com	Email	0.99+

► Application integration

Labels

The **Labels** analysis mode checks for the presence of PII in your text document and returns the labels of identified PII entity types. For more information, see [Label Documents with PII Entity Types \(p. 110\)](#).

The screenshot shows the 'PII' tab selected in the navigation bar. It displays a table of results with columns for Type and Confidence. The results are:

Type	Confidence
Email	0.99+
Address	0.99+
Bank routing	0.76
Phone	0.76

Below the table, there is a link labeled 'Application integration'.

Sentiment

The **Sentiment** tab shows the overall emotional sentiment of the text. Sentiment can be rated neutral, positive, negative, or mixed. In this case, each emotional sentiment has a confidence rating, providing an estimate by Amazon Comprehend for that sentiment being dominant. For more information, see [Determine Sentiment \(p. 111\)](#).

The screenshot shows the Amazon Comprehend Insights interface. At the top, there's a navigation bar with tabs: Entities, Key phrases, Language, PII, Sentiment (which is highlighted in orange), and Syntax. Below the navigation bar, the text being analyzed is displayed:

Hello Zhang Wei. Your AnyCompany Financial Services, LLC credit card account 1111-0000-1111-0000 has a minimum payment of \$24.53 that is due by July 31st. Based on your autopay settings, we will withdraw your payment on the due date from your bank account XXXXXX1111 with the routing number XXXXX0000.

Your latest statement was mailed to 100 Main Street, Anytown, WA 98121.
After your payment is received, you will receive a confirmation text message at 206-555-0100.
If you have questions about your bill, AnyCompany Customer Service is available by phone at 206-555-0199 or email at support@anycompany.com.

Below the text, there's a section titled "Results" with a "Sentiment" heading. It shows four categories: Neutral (0.99 confidence), Positive (0.00 confidence), Negative (0.00 confidence), and Mixed (0.00 confidence). There's also a link labeled "Application integration".

Syntax

The **Syntax** tab shows a breakdown of each element in the text, along with its part of speech and the associated confidence score. For more information, see [Analyze Syntax \(p. 112\)](#).

The screenshot shows the 'Syntax' tab selected in the 'Insights' interface. The 'Analyzed text' section contains a sample credit card statement. The 'Results' section displays a table of tokens and their properties.

Word	Part of speech	Confidence
Hello	Interjection	0.98
Zhang	Proper noun	0.99+
Wei	Proper noun	0.99+
.	Punctuation	0.99+
Your	Pronoun	0.99+
AnyCompany	Proper noun	0.99+
Financial	Proper noun	0.99+
Services	Proper noun	0.99+
,	Punctuation	0.99+
LLC	Proper noun	0.98

Creating and Using Custom Entity Recognizer

You can create custom entity recognizers using the Amazon Comprehend console. This section shows you how to create and train a custom entity recognizer and then how to create an entity recognizer job.

Creating a Custom Entity Recognizer Using the Console - CSV Format

To create the custom entity recognizer, first provide a dataset to train your model. With this dataset, include one of the following: a set of annotated documents or a list of entities and their type label, along with a set of documents containing those entities. For more information, see [Custom Entity Recognition \(p. 147\)](#)

To train a custom entity recognizer with a CSV file

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).

2. From the left menu, choose **Customization** and then choose **Custom entity recognition**.
3. Choose **Create new model**.
4. Give the recognizer a name. The name must be unique within the Region and account.
5. Select the language.
6. Under **Custom entity type**, enter a custom label that you want the recognizer to find in the dataset.

The entity type must be uppercase, and if it consists of more than one word, separate the words with an underscore.

7. Choose **Add type**.
8. If you want to add an additional entity type, enter it, and then choose **Add type**. If you want to remove one of the entity types you've added, choose **Remove type** and then choose the entity type to remove from the list. A maximum of 25 entity types can be listed.
9. To encrypt your training job, choose **Recognizer encryption** and then choose whether to use a KMS key associated with the current account, or one from another account.
 - If you are using a key associated with the current account, for **KMS key ID** choose the key ID.
 - If you are using a key associated with a different account, for **KMS key ARN** enter the ARN for the key ID.

Note

For more information on creating and using KMS keys and the associated encryption, see [Key Management Service \(KMS\)](#).

10. Under **Data specifications**, choose the format of your training documents:

- **CSV file** — A CSV file that supplements your training documents. The CSV file contains information about the custom entities that your trained model will detect. The required format of the file depends on whether you are providing annotations or an entity list.
- **Augmented manifest** — A labeled dataset that is produced by Amazon SageMaker Ground Truth. This file is in JSON lines format. Each line is a complete JSON object that contains a training document and its labels. Each label annotates a named entity in the training document. You can provide up to 5 augmented manifest files.

For more information about available formats, and for examples, see [Training Custom Entity Recognizers \(p. 148\)](#).

11. Under **Training type**, choose the training type to use:

- **Using annotations and training docs**
- **Using entity list and training docs**

If choosing annotations, enter the URL of the annotations file in Amazon S3. You can also navigate to the bucket or folder in Amazon S3 where the annotation files are located and choose **Browse S3**.

If choosing entity list, enter the URL of the entity list in Amazon S3. You can also navigate to the bucket or folder in Amazon S3 where the entity list is located and choose **Browse S3**.

12. Enter the URL of an input dataset containing the training documents in Amazon S3. You can also navigate to the bucket or folder in Amazon S3 where the training documents are located and choose **Select folder**.
13. Under **Test dataset** select how you want to evaluate the performance of your trained model - you can do this for both annotations and entity list training types.
 - **Autosplit**: Autosplit automatically selects 10% of your provided training data to use as testing data

- (Optional) **Customer provided:** When you select customer provided, you can specify exactly what test data you want to use.
14. If you select Customer provided test dataset, enter the URL of the annotations file in Amazon S3. You can also navigate to the bucket or folder in Amazon S3 where the annotation files are located and choose **Select folder**.
 15. In the **Choose an IAM role** section, either select an existing IAM role or create a new one.
 - **Choose an existing IAM role** – Select this option if you already have an IAM role with permissions to access the input and output Amazon S3 buckets.
 - **Create a new IAM role** – Select this option when you want to create a new IAM role with the proper permissions for Amazon Comprehend to access the input and output buckets.
- Note**
If the input documents are encrypted, the IAM role used must have `kms:Decrypt` permission. For more information, see [Permissions Required to Use KMS Encryption \(p. 194\)](#).
16. (Optional) To launch your resources into Amazon Comprehend from a VPC, enter the VPC ID under **VPC** or choose the ID from the drop-down list.
 1. Choose the subnet under **Subnet(s)**. After you select the first subnet, you can choose additional ones.
 2. Under **Security Group(s)**, choose the security group to use if you specified one. After you select the first security group, you can choose additional ones.

Note

When you use a VPC with your custom entity recognition job, the `DataAccessRole` used for the Create and Start operations must have permissions to the VPC from which the input documents and the output bucket are accessed.

17. (Optional) To add a tag to the custom entity recognizer, enter a key-value pair under **Tags**. Choose **Add tag**. To remove this pair before creating the recognizer, choose **Remove tag**.
18. Choose **Train**.

The new recognizer will then appear in the list, showing its status. It will first show as **Submitted**. It will then show **Training** for a classifier that is processing training documents, **Trained** for a classifier that is ready to use, and **In error** for a classifier that has an error. You can click on a job to get more information about the recognizer, including any error messages.

[Creating a Custom Entity Recognizer Using the Console - Augmented Manifest](#)

To train a custom entity recognizer with a Plain text, PDF, or Word Document

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Customization** and then choose **Custom entity recognition**.
3. Choose **Train recognizer**.
4. Give the recognizer a name. The name must be unique within the Region and account.
5. Select the language. Note: If you're training a PDF or Word document, English is the supported language.
6. Under **Custom entity type**, enter a custom label that you want the recognizer to find in the dataset.

The entity type must be uppercase, and if it consists of more than one word, separate the words with an underscore.

7. Choose **Add type**.

8. If you want to add an additional entity type, enter it, and then choose **Add type**. If you want to remove one of the entity types you've added, choose **Remove type** and then choose the entity type to remove from the list. A maximum of 25 entity types can be listed.
9. To encrypt your training job, choose **Recognizer encryption** and then choose whether to use a KMS key associated with the current account, or one from another account.
 - If you are using a key associated with the current account, for **KMS key ID** choose the key ID.
 - If you are using a key associated with a different account, for **KMS key ARN** enter the ARN for the key ID.

Note

For more information on creating and using KMS keys and the associated encryption, see [Key Management Service \(KMS\)](#).

10. Under **Training data**, choose **Augmented manifest** as your data format:

- **Augmented manifest** — is a labeled dataset that is produced by Amazon SageMaker Ground Truth. This file is in JSON lines format. Each line in the file is a complete JSON object that contains a training document and its labels. Each label annotates a named entity in the training document. You can provide up to 5 augmented manifest files. If you are using PDF documents for training data, you must select **Augmented manifest**. You can provide up to 5 augmented manifest files. For each file, you can name up to 5 attributes to use as training data.

For more information about available formats, and for examples, see [Training Custom Entity Recognizers \(p. 148\)](#).

11. Select the training model type.

If you selected **Plain text documents**, under **Input location**, enter the Amazon S3URL of the Amazon SageMakerGround Truth augmented manifest file. You can also navigate to the bucket or folder in Amazon S3 where the augmented manifest(s) is located and choose **Select folder**.

12. Under **Attribute name**, enter the name of the attribute that contains your annotations. If the file contains annotations from multiple chained labeling jobs, add an attribute for each job. In this case, each attribute contains the set of annotations from a labeling job. Note: You can provide up to 5 attribute names for each file.

13. Select **Add**.

14. If you selected **PDF, Word documents** under **Input location**, enter the Amazon S3URL of the Amazon SageMaker Ground Truth augmented manifest file. You can also navigate to the bucket or folder in Amazon S3 where the augmented manifest(s) is located and choose **Select folder**.

15. Enter the S3 prefix for your **Annotation** data files. These are the PDF documents that you labeled.

16. Enter the S3 prefix for your **Source** documents. These are the original PDF documents (data objects) that you provided to Ground Truth for your labeling job.

17. Enter the attribute names that contain your annotations. Note: You can provide up to 5 attribute names for each file. Any attributes in your file that you don't specify are ignored.

18. In the IAM role section, either select an existing IAM role or create a new one.

- **Choose an existing IAM role** – Select this option if you already have an IAM role with permissions to access the input and output Amazon S3 buckets.
- **Create a new IAM role** – Select this option when you want to create a new IAM role with the proper permissions for Amazon Comprehend to access the input and output buckets.

Note

If the input documents are encrypted, the IAM role used must have `kms:Decrypt` permission. For more information, see [Permissions Required to Use KMS Encryption \(p. 194\)](#).

19. (Optional) To launch your resources into Amazon Comprehend from a VPC, enter the VPC ID under **VPC** or choose the ID from the drop-down list.

1. Choose the subnet under **Subnet(s)**. After you select the first subnet, you can choose additional ones.
2. Under **Security Group(s)**, choose the security group to use if you specified one. After you select the first security group, you can choose additional ones.

Note

When you use a VPC with your custom entity recognition job, the `DataAccessRole` used for the Create and Start operations must have permissions to the VPC from which the input documents and the output bucket are accessed.

20. (Optional) To add a tag to the custom entity recognizer, enter a key-value pair under **Tags**. Choose **Add tag**. To remove this pair before creating the recognizer, choose **Remove tag**.

21. Choose **Train**.

The new recognizer will then appear in the list, showing its status. It will first show as `Submitted`. It will then show `Training` for a classifier that is processing training documents, `Trained` for a classifier that is ready to use, and `In error` for a classifier that has an error. You can click on a job to get more information about the recognizer, including any error messages.

Creating and Using Custom Classifiers

You can create and train custom classifiers using the console, and then run asynchronous classification jobs to analyze your documents. You can also use the same custom model and add an endpoint to it to run custom classification requests to gain real-time (synchronous) insights about your text. This section shows you how to create a classifier using the console and then both how to use it to run an asynchronous classification job, or how to create an endpoint for it and run a real-time classification request.

Topics

- [Creating a Custom Classifier \(Console\) \(p. 44\)](#)
- [Running an Asynchronous Custom Classification Job \(p. 46\)](#)
- [Creating a Real-time Custom Classification Request \(p. 48\)](#)

Creating a Custom Classifier (Console)

Create a custom document classifier to identify the categories of a set of documents.

To train the classifier, you need a set of training documents. You label these documents with the categories that you want the document classifier to recognize. For more information on these training documents, see [Custom Classification \(p. 125\)](#).

To train a document classifier

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Customization** and then choose **Custom Classification**.
3. Choose **Create new model**.
4. Give the classifier a name. The name must be unique within your account and current Region.
5. Select the language of the training documents. You can train a document classifier using any of the languages that work with Amazon Comprehend. However, you can only train the classifier in one language. To learn more, see [Languages Supported by Amazon Comprehend. \(p. 5\)](#)

6. (Optional) If you want to encrypt the data in the storage volume while your training job is being processed, choose **Classifier encryption** and then choose whether to use a KMS key associated with your current account, or one from another account.
 - If you are using a key associated with the current account, choose the key ID for **KMS key ID**.
 - If you are using a key associated with a different account, enter the ARN for the key ID under **KMS key ARN**.

Note

For more information on creating and using KMS keys and the associated encryption, see [Key Management Service \(KMS\)](#).

7. Under **Data specifications**, choose which classifier mode to use.
 - **Single-label mode:** Choose this option if the categories you are assigning to documents are mutually exclusive and you are training your classifier to assign one and only one label to each document.
 - **Multi-label mode:** Choose this option if multiple categories can be applied to a document at the same time and you are training your classifier to assign one, many, all, or no label to each document.
8. If you chose **Multi-label mode**, choose the character delimiter you want to use to separate labels when there are more than one label per line from **Delimiter for labels**.
9. Under **Data format**, choose the format of your training documents:
 - **CSV file** — A two-column CSV file, where labels are provided in the first column, and documents are provided in the second.
 - **Augmented manifest** — A labeled dataset that is produced by Amazon SageMaker Ground Truth. This file is in JSON lines format. Each line is a complete JSON object that contains a training document and its associated labels.

For more information about these formats, and for examples, see [Training a Custom Classifier \(p. 126\)](#).

10. Under **Training dataset**, enter the location of the Amazon S3 bucket that contains your training documents or navigate to it by choosing **Select folder**. The IAM role you're using for access permissions for the training job must have reading permissions for the S3 bucket.
11. Under **Test dataset** select how you want to evaluate the performance of your trained model - you can do this for both annotations and entity list training types.
 - **Autosplit:** Autosplit automatically selects 10% of your provided training data to use as testing data
 - (Optional) **Customer provided:** When you select customer provided, you can specify exactly what test data you want to use. If you select Customer provided test dataset, enter the URL of the annotations file in Amazon S3. You can also navigate to the bucket or folder in Amazon S3 where the annotation files are located and choose **Select folder**.
12. (Optional) If you want Amazon Comprehend to create a confusion matrix that provides metrics on how well the classifier performed during training, enter the location of an Amazon S3 bucket where it will be saved. For more information, see [Confusion Matrix \(p. 145\)](#).

(Optional) If you choose to encrypt the output result from your training job, choose **Encryption** and then choose whether to use a KMS key associated with the current account, or one from another account.

- If you are using a key associated with the current account, choose the key alias for **KMS key ID**.
- If you are using a key associated with a different account, enter the ARN for the key alias or ID under **KMS key ID**.

13. Choose **Choose an existing IAM role**, and then choose an existing IAM role that has read permissions for the S3 bucket that contains your training documents. Only roles that have a trust policy that begins with comprehend.amazonaws.com are valid.

If you don't already have an IAM role with these permissions, choose **Create an IAM role** to make one. Choose the access permissions to grant this role, and then choose a name suffix to distinguish the role from IAM roles in your account.

Note

If the input documents are encrypted, the IAM role used must also have `kms:Decrypt` permission. For more information, see [Permissions Required to Use KMS Encryption \(p. 194\)](#).

14. (Optional) To launch your resources into Amazon Comprehend from a VPC, enter the VPC ID under **VPC** or choose the ID from the drop-down list.

1. Choose the subnet under **Subnets(s)**. After you select the first subnet, you can choose additional ones.
2. Under **Security Group(s)**, choose the security group to use if you specified one. After you select the first security group, you can choose additional ones.

Note

When you use a VPC with your classification job, the `DataAccessRole` used for the Create and Start operations must have permissions to the VPC from which the input documents and the output bucket are accessed.

15. (Optional) To add a tag to the custom classifier, enter a key-value pair under **Tags**. Choose **Add tag**. To remove this pair before creating the classifier, choose **Remove tag**. For more information, see [Tagging Custom Classifiers \(p. 137\)](#).
16. Choose **Create**.

The new classifier will then appear in the list, showing its status. It will first show as Submitted. It will then show Training for a classifier that is processing training documents, Trained for a classifier that is ready to use, and In error for a classifier that has an error. You can click on a job to get more information about the classifier, including any error messages.

Name	Training started	Training ended	Status
classifiertags5-copy	7/22/2019, 3:48:38 PM	7/22/2019, 3:57:18 PM	Trained
classifiertags5	6/24/2019, 3:40:28 PM	6/24/2019, 3:47:26 PM	Trained
classifiertags:	6/3/2019, 6:33:16 PM	6/3/2019, 6:33:35 PM	In error
hk-classifier-output-2	4/9/2019, 11:28:26 AM	4/9/2019, 11:28:29 AM	In error

Running an Asynchronous Custom Classification Job

Once you have created a custom document classifier, you can use it to categorize a group of documents.

To create a custom asynchronous classification job

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Customization** and then choose **Custom classification**.
3. Choose **Create job**.
4. Give the classification job a name. The name must be unique your account and current Region.
5. Under **Analysis type**, choose **Custom classification**.
6. From **Select classifier**, choose the custom classifier to use.
7. (Optional) If you choose to encrypt the data in the storage volume while your classification job is processed, choose **Job encryption** and then choose whether to use a KMS key associated with the current account, or one from another account.
 - If you are using a key associated with the current account, choose the key ID for **KMS key ID**.
 - If you are using a key associated with a different account, enter the ARN for the key ID under **KMS key ARN**.

Note

For more information on creating and using KMS keys and the associated encryption, see [Key Management Service \(KMS\)](#).

8. Under **Input data**, enter the location of the Amazon S3 bucket that contains your input documents or navigate to it by choosing **Select folder**. This bucket must be in the same region as the API that you are calling. The IAM role you're using for access permissions for the classification job must have reading permissions for the S3 bucket.
9. (Optional) Choose the format of the documents to be classified under **Input format**. These can be one document per file, or one document per line in a single file.
10. Under **Output data**, enter the location of the Amazon S3 bucket where Amazon Comprehend should write the job's output data or navigate to it by choosing **Select folder**. This bucket must be in the same region as the API that you are calling. The IAM role you're using for access permissions for the classification job must have write permissions for the S3 bucket.
11. (Optional) If you choose to encrypt the output result from your job, choose **Encryption** and then choose whether to use a KMS key associated with the current account, or one from another account.
 - If you are using a key associated with the current account, choose the key alias or ID for **KMS key ID**.
 - If you are using a key associated with a different account, enter the ARN for the key alias or ID under **KMS key ID**.
12. (Optional) To launch your resources into Amazon Comprehend from a VPC, enter the VPC ID under **VPC** or choose the ID from the drop-down list.
 1. Choose the subnet under **Subnet(s)**. After you select the first subnet, you can choose additional ones.
 2. Under **Security Group(s)**, choose the security group to use if you specified one. After you select the first security group, you can choose additional ones.
13. Choose **Create job** to create the document classification job.

Note

When you use a VPC with your classification job, the `DataAccessRole` used for the Create and Start operations must have permissions to the VPC from which the output bucket are accessed.

Creating a Real-time Custom Classification Request

In addition to using the custom document classifier to run asynchronous jobs, you can also use it to run synchronous custom classification requests to gain real-time insight into the categories in your document. This requires first that you create an endpoint and set the level of data throughput for it, and then to run the real-time analysis.

Note

Using real-time analysis will result in additional cost to your account. This cost is determined by how long the endpoint is operating and the level of throughput you determine.

The level of throughput assigned to an endpoint is measured in *Inference units*, each of which represents data throughput of 100 characters per second. You can provision the endpoint with up to 10 inference units. This level of throughput can be adjusted to meet your needs by updating the endpoint.

Once you have completed your real-time analysis, you should delete the endpoint because the charge for it will continue as long as it's active. You can easily create another endpoint whenever you need it.

To create an endpoint

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Customization** and then choose **Custom Classification**.
3. From the **Classifiers** list, choose the name of the custom model for which you want to create the endpoint and follow the link. The **Endpoints** list on the custom model details page is displayed.

Note

Previously created endpoints are shown on the models detail page, along with the model with which they're associated.

4. Under **Endpoints**, choose **Create endpoint**.
5. Give the endpoint a name. The name must be unique within the AWS Region and account.
6. Enter the number of inference units to assign to the endpoint. Each unit represents a throughput of 100 characters per second. You can assign up to a maximum of 10 inference units per endpoint.
7. (Optional) To add a tag to the endpoint, enter a key-value pair under **Tags** and choose **Add tag**. To remove this pair before creating the endpoint, choose **Remove tag**.
8. Choose **Create endpoint**. The **Endpoints** list is displayed, with the new endpoint showing **Creating**. Once it shows **Ready**, the endpoint can be used for real-time analysis.

To run a real-time custom classification request

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Real-time analysis**.
3. Under **Input type**, choose **Custom** for **Analysis type**.
4. For **Select endpoint**, choose the endpoint that you want to use. This endpoint is linked to a specific custom model.
5. Enter the text you want to analyze.
6. Choose **Analyze**. The text analysis based on your custom model is displayed, along with a confidence assessment of the analysis.

To update your endpoint

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).

2. From the left menu, choose **Customization** and then choose **Custom classification**.
3. From the **Classifiers** list, choose the name of the custom model for which you want to update the endpoint and follow the link. The custom model details page is displayed.
4. Navigate to the **Endpoints** list, choose the name of the endpoint you want to update and follow the link.
5. Choose **Edit**.
6. Enter the updated number of inference units to assign to the endpoint. Each unit represents a throughput of 100 characters per second. You can assign up to a maximum of 10 inference units per endpoint.

Note

The cost of using an endpoint is based on the amount of time operating and the throughput (based on the number of inference units). Increasing the number of inference units will thus increase the cost of operation. For more information, see [Amazon Comprehend Pricing](#).

7. Choose **Edit endpoint**. The endpoint details page is displayed.
8. Confirm that the endpoint is updating by choosing the model name from the breadcrumbs at the top of the page. On the custom model details page, navigate to the **Endpoints** list and verify that it shows **Updating** next to the endpoint. When the update is complete, it will show **Ready**.

To delete your endpoint

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Customization** and then choose **Custom classification**.
3. From the **Classifiers** list, choose the name of the custom model associated with the endpoint you want to delete and follow the link. The custom model details page is displayed.
4. Navigate to the **Endpoints** list, choose the name of the endpoint to delete and follow the link.
5. Choose **Delete**.

Note

All endpoints associated with a custom model must be deleted before that model itself can be removed.

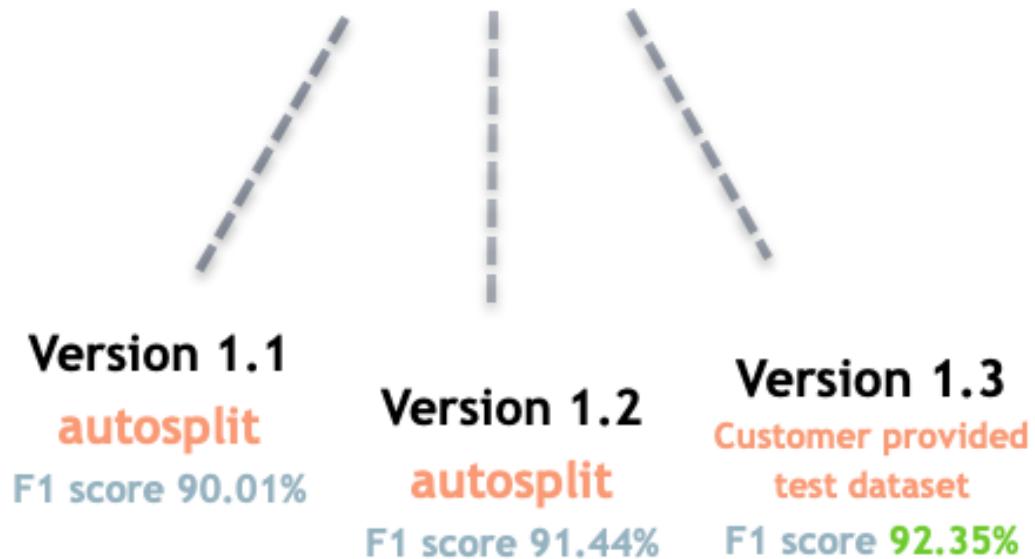
6. Choose **Delete** again to confirm the deletion. The custom model details page is displayed. Confirm that the endpoint you deleted shows **deleting** next to it. When it's deleted, the endpoint is removed from the **Endpoints** list.

Model Versioning with Amazon Comprehend

Artificial intelligence and machine learning (AI/ML) is all about rapid experimentation. With Amazon Comprehend, you train and build out models which you use to gain insight on your data. With model versioning you can keep track of your modeling history and scores associated with running results of your models as you provide more or different sets of data. You can use versioning with your custom classification models or your custom entity recognition models. Taking a look at your different versions over time you can gain insight on how successful they've performed and gain insight on what parameters you used to get to your state of success.

When you train a new version of an existing custom classifier model or entity recognition model, all you need to do is create a new version from the model details page and all the details populate for you. The new version will have the same name as your earlier model — what we call the `versionID` — although you will give it a unique version name during creation. As you add new versions to a model, you can see all the previous versions and their details in one view from the model details page. With versioning, you can see how model performance changes as you make changes to your training dataset.

Model 1.0



Create a new Custom classifier version (console)

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Customization** and then choose **Custom classification**.
3. From the **Classifiers** list, choose the name of the custom model from which you want to create a new version. The custom model details page is displayed.
4. On the top right, select **Create new model**. A screen opens with prepopulated details from the parent custom classification model.
5. Under **Version name** add a unique name to the new version.
6. Under version details, you can change the language and number of labels associated with your new model.
7. Under the **Data specifications** section configure how you want to provide the data to your new version—make sure to provide full data, which includes documents from your previous model and your new documents. You can change the **Classifier mode** (single-label, or multi-label), **Data format** (CSV file, Augmented manifest), your **Training dataset**, and your **Test dataset** (autosplit, or your custom test data configuration).
8. (Optional) update the S3 location for your output data
9. Under **Access permissions**, create or use an existing IAM role.
10. (Optional) Update your VPC settings

11. (Optional) Add tags to your new version to help keep track of the details.

For more information about creating custom classifiers, see [Custom Classification \(p. 125\)](#) and [Creating and Using Custom Classifiers \(p. 44\)](#)

Create a new Custom entity recognizer version (console)

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Customization** and then choose **Custom entity recognition**.
3. From the **Recognizer model** list, choose the name of the recognizer from which you want to create a new version. The details page is displayed.
4. On the top right, select **Train new version**. A screen opens with prepopulated details from the parent entity recognizer.
5. Under **Version name** add a unique name to the new version.
6. Under Custom entity type, add the custom labels or label you want the recognizer to identify in your dataset and select **Add type**. Choose a custom entity type from the annotations or entity list you've provided. The recognizer will then use all of the included entity types to identify entities in the data set when running your job. Each entity type must be upper-case and separated by and underscore if it uses multiple words. A maximum of 25 types are allowed.
7. (Optional) Select **Recognizer encryption** to encrypt the data in the storage volume while your job is being processed.
8. Under the Training data section, specify the **Annotation and data format** details (CSV file, Augmented manifest)single-label, or multi-label), **Data format** (CSV, Augmented manifest), your **Training dataset**, and your **Test dataset** (autosplit, or your custom test data configuration).
9. (Optional) update the S3 location for your output data
10. Under **Access permissions**, create or use an existing IAM role.
11. (Optional) Update your VPC settings
12. (Optional) Add tags to your new version to help keep track of the details.

To learn more about custom entity recognizers, see [Custom Entity Recognition \(p. 147\)](#) and [Creating a Custom Entity Recognizer Using the Console \(p. 40\)](#).

Creating a Topic Modeling Job Using the Console

You can use the Amazon Comprehend console to create and manage asynchronous topic detection jobs.

To create a topic modeling job

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Analysis Jobs** and then choose **Create**.
3. Under **Job settings**, give the job a name. The name must be unique within the region and account.
4. For **Analysis Type**, choose **Topic Modeling**.
5. (Optional) If you choose to encrypt the data in the storage volume while your job is processed, choose **Job encryption** and then choose whether to use a KMS key associated with the current account, or one from another account.
 - If you are using a key associated with the current account, for **KMS key ID**choose the key ID.
 - If you are using a key associated with a different account, for **KMS key ARN** enter the ARN for the key ID.

Note

For more information on creating and using KMS keys and the associated encryption, see [Key Management Service \(KMS\)](#).

6. Choose the data source to use. You can use either sample data or you can analyze your own data stored in an Amazon S3 bucket.

If you choose to use your own data, provide the following information:

- **S3 data location** – An Amazon S3 data bucket that contains the documents to analyze. You can choose the folder icon to browse to the location of your data. The bucket must be in the same region as the API that you are calling.
 - **Input format** – Optionally choose whether input data is contained in one document per file, or if there is one document per line in a file.
 - **Number of topics** – The number of topics to return.
7. (Optional) If you choose to encrypt the output result from your job, choose **Encryption** and then choose whether to use a KMS key associated with the current account, or one from another account.
 - If you are using a key associated with the current account, for **KMS key ID** choose the key alias or ID.
 - If you are using a key associated with a different account, for **KMS key ID** enter the ARN for the key alias or ID.
 8. In the **Choose an IAM role** section, either select an existing IAM role or create a new one.
 - **Choose an existing IAM role** – Choose this option if you already have an IAM role with permissions to access the input and output Amazon S3 buckets.
 - **Create a new IAM role** – Choose this option when you want to create a new IAM role with the proper permissions for Amazon Comprehend to access the input and output buckets. For more information about the permissions given to the IAM role, see [Role-Based Permissions Required for Asynchronous Operations \(p. 196\)](#).

Note

If the input documents are encrypted, the IAM role used must have `KMS:Decrypt` permission. For more information, see [Permissions Required to Use KMS Encryption \(p. 194\)](#).

9. (Optional) To launch your resources into Amazon Comprehend from a VPC, enter the VPC ID under **VPC** or choose the ID from the drop-down list.
 1. Choose the subnet under **Subnet(s)**. After you select the first subnet, you can choose additional ones.
 2. Under **Security Group(s)**, choose the security group to use if you specified one. After you select the first security group, you can choose additional ones.

Note

When you use a VPC with your topic modeling job, the `DataAccessRole` that is used for the Create and Start operations must have permissions to the VPC from which the input documents and the output bucket are accessed.

10. When you have finished filling out the form, choose **Create job** to create and start the topic detection job.

The new job appears in the job list with the status field showing the status of the job. The field can be `IN_PROGRESS` for a job that is processing, `COMPLETED` for a job that has finished successfully,

and **FAILED** for a job that has an error. You can click on a job to get more information about the job, including any error messages.

Creating an Events Detection Job Using the Console

You can use the Amazon Comprehend console to create and manage asynchronous events detection jobs.

To create an events detection job

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Analysis jobs** and then choose **Create job**.
3. Under **Job settings**, give the analysis job a unique name.
4. For **Analysis type**, choose **Events**.
5. For **Language**, choose the language of your input documents.
6. For **Target event types**, select the types of events to detect in your input documents. For more information about supported event types, see [Event Types \(p. 98\)](#).
7. Under **Input data**, specify where the input documents are located in Amazon S3:
 - To analyze your own documents, choose **My documents**, and choose **Browse S3** to provide the path to the bucket or folder that contains your files.
 - To analyze samples that are provided by Amazon Comprehend, choose **Example documents**. In this case, Amazon Comprehend uses a bucket that is managed by AWS, and you don't specify the location.
8. (Optional) For **Input format**, specify one of the following formats for your input files:
 - **One document per file** – Each file contains one input document. This is best for collections of large documents.
 - **One document per line** – The input is one or more files. Each line in a file is considered a document. This is best for short documents, such as social media postings. Each line must end with a line feed (LF, \n), a carriage return (CR, \r), or both (CRLF, \r\n). You can't use the UTF-8 line separator (u+2028) to end a line.
9. Under **Output data**, choose **Browse S3**. Choose the Amazon S3 bucket or folder where you want Amazon Comprehend to write the output data that is produced by the analysis.
10. (Optional) To encrypt the output result from your job, choose **Encryption**. Then, choose whether to use a KMS key associated with the current account or one from another account:
 - If you are using a key associated with the current account, choose the key alias or ID for **KMS key ID**.
 - If you are using a key associated with a different account, enter the ARN for the key alias or ID under **KMS key ID**.

Note

For more information on creating and using KMS keys and the associated encryption, see [Key Management Service \(KMS\)](#).

11. Under **Access permissions**, provide an IAM role that:
 - Grants read access to the Amazon S3 location of your input documents.
 - Grants write access to the Amazon S3 location of your output documents.
 - Includes a trust policy that allows the `comprehend.amazonaws.com` service principal to assume the role and gain its permissions.

If you don't already have an IAM role with these permissions and an appropriate trust policy, choose [Create an IAM role](#) to create one.

12. When you have finished filling out the form, choose **Create job** to create and start the topic detection job.

The new job appears in the job list with the status field showing the status of the job. The field can be `IN_PROGRESS` for a job that is processing, `COMPLETED` for a job that has finished successfully, and `FAILED` for a job that has an error. You can click on a job to get more information about the job, including any error messages.

Next Step

[Step 4: Getting Started Using the Amazon Comprehend API \(p. 54\)](#)

Step 4: Getting Started Using the Amazon Comprehend API

The following examples demonstrate how to use Amazon Comprehend operations using the AWS CLI, Java, and Python. Use them to learn about Amazon Comprehend operations and as building blocks for your own applications.

To run the AWS CLI and Python examples, you need to install the AWS CLI. For more information, see [Step 2: Set Up the AWS Command Line Interface \(AWS CLI\) \(p. 32\)](#).

To run the Java examples, you need to install the AWS SDK for Java. For instructions for installing the SDK for Java, see [Set up the AWS SDK for Java](#).

Topics

- [Detecting the Dominant Language \(p. 54\)](#)
- [Detecting Named Entities \(p. 57\)](#)
- [Detecting Key Phrases \(p. 59\)](#)
- [Detecting Personally Identifiable Information \(PII\) \(p. 61\)](#)
- [Labeling Documents with Personally Identifiable Information \(PII\) \(p. 62\)](#)
- [Detecting Sentiment \(p. 63\)](#)
- [Detecting Syntax \(p. 65\)](#)
- [Using Custom Classification \(p. 68\)](#)
- [Detecting Custom Entities \(p. 72\)](#)
- [Detecting Events \(p. 76\)](#)
- [Topic Modeling \(p. 78\)](#)
- [Using the Batch APIs \(p. 84\)](#)

Detecting the Dominant Language

To determine the dominant language used in text, use the Amazon Comprehend [DetectDominantLanguage \(p. 291\)](#) operation. To detect the dominant language in up to 25 documents in a batch, use the [BatchDetectDominantLanguage \(p. 220\)](#) operation. For more information, see [Using the Batch APIs \(p. 84\)](#).

Topics

- [Detecting the Dominant Language Using the AWS Command Line Interface \(p. 55\)](#)
- [Detecting the Dominant Language Using the AWS SDK for Java \(p. 55\)](#)
- [Detecting the Dominant Language Using the AWS SDK for Python \(Boto\) \(p. 56\)](#)

- Detecting the Dominant Language Using the AWS SDK for .NET (p. 56)

Detecting the Dominant Language Using the AWS Command Line Interface

The following example demonstrates using the `DetectDominantLanguage` operation with the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend detect-dominant-language \
--region region \
--text "It is raining today in Seattle."
```

Amazon Comprehend responds with the following:

```
{
    "Languages": [
        {
            "LanguageCode": "en",
            "Score": 0.9793661236763
        }
    ]
}
```

Detecting the Dominant Language Using the AWS SDK for Java

The following example uses the `DetectDominantLanguage` operation with Java.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.comprehend.AmazonComprehend;
import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;
import com.amazonaws.services.comprehend.model.DetectDominantLanguageRequest;
import com.amazonaws.services.comprehend.model.DetectDominantLanguageResult;

public class App
{
    public static void main( String[] args )
    {

        String text = "It is raining today in Seattle";

        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWSCredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();

        AmazonComprehend comprehendClient =
            AmazonComprehendClientBuilder.standard()
                .withCredentials(awsCreds)
                .withRegion("region")
                .build();

        // Call detectDominantLanguage API
        System.out.println("Calling DetectDominantLanguage");
        DetectDominantLanguageRequest detectDominantLanguageRequest = new
        DetectDominantLanguageRequest().withText(text);
        DetectDominantLanguageResult detectDominantLanguageResult =
            comprehendClient.detectDominantLanguage(detectDominantLanguageRequest);
```

```
    detectDominantLanguageResult.getLanguages().forEach(System.out::println);
    System.out.println("Calling DetectDominantLanguage\n");
    System.out.println("Done");
}
```

Detecting the Dominant Language Using the AWS SDK for Python (Boto)

The following example demonstrates using the `DetectDominantLanguage` operation with Python.

```
import boto3
import json

comprehend = boto3.client(service_name='comprehend', region_name='region')
text = "It is raining today in Seattle"

print('Calling DetectDominantLanguage')
print(json.dumps(comprehend.detect_dominant_language(Text = text), sort_keys=True,
    indent=4))
print("End of DetectDominantLanguage\n")
```

Detecting the Dominant Language Using the AWS SDK for .NET

The .NET example in this section uses the [AWS SDK for .NET](#). You can use the [AWS Toolkit for Visual Studio](#) to develop AWS applications using .NET. It includes helpful templates and the AWS Explorer for deploying applications and managing services. For a .NET developer perspective of AWS, see the [AWS Guide for .NET Developers](#).

```
using System;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

namespace Comprehend
{
    class Program
    {
        static void Main(string[] args)
        {
            String text = "It is raining today in Seattle";

            AmazonComprehendClient comprehendClient = new
            AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectDominantLanguage API
            Console.WriteLine("Calling DetectDominantLanguage\n");
            DetectDominantLanguageRequest detectDominantLanguageRequest = new
            DetectDominantLanguageRequest()
            {
                Text = text
            };
            DetectDominantLanguageResponse detectDominantLanguageResponse =
            comprehendClient.DetectDominantLanguage(detectDominantLanguageRequest);
            foreach (DominantLanguage dl in detectDominantLanguageResponse.Languages)
                Console.WriteLine("Language Code: {0}, Score: {1}", dl.LanguageCode,
                    dl.Score);
            Console.WriteLine("Done");
        }
    }
}
```

Detecting Named Entities

To determine the named entities in a document, use the Amazon Comprehend [DetectEntities](#) (p. 294) operation. To detect entities in up to 25 documents in a batch, use the [BatchDetectEntities](#) (p. 223) operation. For more information, see [Using the Batch APIs](#) (p. 84).

Topics

- [Detecting Named Entities Using the AWS Command Line Interface](#) (p. 57)
- [Detecting Named Entities Using the AWS SDK for Java](#) (p. 57)
- [Detecting Named Entities Using the AWS SDK for Python \(Boto\)](#) (p. 58)
- [Detecting Entities Using the AWS SDK for .NET](#) (p. 58)

Detecting Named Entities Using the AWS Command Line Interface

The following example demonstrates using the `DetectEntities` operation using the AWS CLI. You must specify the language of the input text.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend detect-entities \
    --region region \
    --language-code "en" \
    --text "It is raining today in Seattle."
```

Amazon Comprehend responds with the following:

```
{
  "Entities": [
    {
      "Text": "today",
      "Score": 0.97,
      "Type": "DATE",
      "BeginOffset": 14,
      "EndOffset": 19
    },
    {
      "Text": "Seattle",
      "Score": 0.95,
      "Type": "LOCATION",
      "BeginOffset": 23,
      "EndOffset": 30
    }
  ],
  "LanguageCode": "en"
}
```

Detecting Named Entities Using the AWS SDK for Java

The following example uses the `DetectEntities` operation with Java. You must specify the language of the input text.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.comprehend.AmazonComprehend;
```

```

import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;
import com.amazonaws.services.comprehend.model.DetectEntitiesRequest;
import com.amazonaws.services.comprehend.model.DetectEntitiesResult;

public class App
{
    public static void main( String[] args )
    {

        String text = "It is raining today in Seattle";

        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWS CredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();

        AmazonComprehend comprehendClient =
            AmazonComprehendClientBuilder.standard()
                .withCredentials(awsCreds)
                .withRegion("region")
                .build();

        // Call detectEntities API
        System.out.println("Calling DetectEntities");
        DetectEntitiesRequest detectEntitiesRequest = new
        DetectEntitiesRequest().withText(text)

        .withLanguageCode("en");
        DetectEntitiesResult detectEntitiesResult =
        comprehendClient.detectEntities(detectEntitiesRequest);
        detectEntitiesResult.getEntities().forEach(System.out::println);
        System.out.println("End of DetectEntities\n");
    }
}

```

Detecting Named Entities Using the AWS SDK for Python (Boto)

The following example uses the `DetectEntities` operation with Python. You must specify the language of the input text.

```

import boto3
import json

comprehend = boto3.client(service_name='comprehend', region_name='region')
text = "It is raining today in Seattle"

print('Calling DetectEntities')
print(json.dumps(comprehend.detect_entities(Text=text, LanguageCode='en'), sort_keys=True,
    indent=4))
print('End of DetectEntities\n')

```

Detecting Entities Using the AWS SDK for .NET

The .NET example in this section uses the [AWS SDK for .NET](#). You can use the [AWS Toolkit for Visual Studio](#) to develop AWS applications using .NET. It includes helpful templates and the AWS Explorer for deploying applications and managing services. For a .NET developer perspective of AWS, see the [AWS Guide for .NET Developers](#).

```

using System;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

namespace Comprehend

```

```
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            String text = "It is raining today in Seattle";  
  
            AmazonComprehendClient comprehendClient = new  
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);  
  
            // Call DetectEntities API  
            Console.WriteLine("Calling DetectEntities\n");  
            DetectEntitiesRequest detectEntitiesRequest = new DetectEntitiesRequest()  
            {  
                Text = text,  
                LanguageCode = "en"  
            };  
            DetectEntitiesResponse detectEntitiesResponse =  
comprehendClient.DetectEntities(detectEntitiesRequest);  
foreach (Entity e in detectEntitiesResponse.Entities)  
            Console.WriteLine("Text: {1}, Type: {2}, Score: {3}, BeginOffset: {4},  
EndOffset: {5}",  
                e.Text, e.Type, e.Score, e.BeginOffset, e.EndOffset);  
            Console.WriteLine("Done");  
        }  
    }  
}
```

Detecting Key Phrases

To determine the key noun phrases used in text, use the Amazon Comprehend [DetectKeyPhrases \(p. 298\)](#) operation. To detect the key noun phrases in up to 25 documents in a batch, use the [BatchDetectKeyPhrases \(p. 226\)](#) operation. For more information, see [Using the Batch APIs \(p. 84\)](#).

Topics

- [Detecting Key Phrases Using the AWS Command Line Interface \(p. 59\)](#)
- [Detecting Key Phrases Using the AWS SDK for Java \(p. 60\)](#)
- [Detecting Key Phrases Using the AWS SDK for Python \(Boto\) \(p. 60\)](#)
- [Detecting Key Phrases Using the AWS SDK for .NET \(p. 61\)](#)

Detecting Key Phrases Using the AWS Command Line Interface

The following example demonstrates using the `DetectKeyPhrases` operation with the AWS CLI. You must specify the language of the input text.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend detect-key-phrases \  
--region region \  
--language-code "en" \  
--text "It is raining today in Seattle."
```

Amazon Comprehend responds with the following:

```
{  
    "LanguageCode": "en",  
    "KeyPhrases": [  
        {
```

```
        "Text": "today",
        "Score": 0.89,
        "BeginOffset": 14,
        "EndOffset": 19
    },
    {
        "Text": "Seattle",
        "Score": 0.91,
        "BeginOffset": 23,
        "EndOffset": 30
    }
]
```

Detecting Key Phrases Using the AWS SDK for Java

The following example uses the `DetectKeyPhrases` operation with Java. You must specify the language of the input text.

```
import com.amazonaws.auth.AWS CredentialsProvider;
import com.amazonaws.auth.DefaultAWS CredentialsProviderChain;
import com.amazonaws.services.comprehend.AmazonComprehend;
import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;
import com.amazonaws.services.comprehend.model.DetectKeyPhrasesRequest;
import com.amazonaws.services.comprehend.model.DetectKeyPhrasesResult;

public class App
{
    public static void main( String[] args )
    {

        String text = "It is raining today in Seattle";

        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWS CredentialsProvider awsCreds = DefaultAWS CredentialsProviderChain.getInstance();

        AmazonComprehend comprehendClient =
            AmazonComprehendClientBuilder.standard()
                .withCredentials(awsCreds)
                .withRegion("region")
                .build();

        // Call detectKeyPhrases API
        System.out.println("Calling DetectKeyPhrases");
        DetectKeyPhrasesRequest detectKeyPhrasesRequest = new
        DetectKeyPhrasesRequest().withText(text)

        .withLanguageCode("en");
        DetectKeyPhrasesResult detectKeyPhrasesResult =
        comprehendClient.detectKeyPhrases(detectKeyPhrasesRequest);
        detectKeyPhrasesResult.getKeyPhrases().forEach(System.out::println);
        System.out.println("End of DetectKeyPhrases\n");
    }
}
```

Detecting Key Phrases Using the AWS SDK for Python (Boto)

The following example uses the `DetectKeyPhrases` operation with Python. You must specify the language of the input text.

```
import boto3
```

```
import json

comprehend = boto3.client(service_name='comprehend', region_name='region')

text = "It is raining today in Seattle"

print('Calling DetectKeyPhrases')
print(json.dumps(comprehend.detect_key_phrases(Text=text, LanguageCode='en'),
    sort_keys=True, indent=4))
print('End of DetectKeyPhrases\n')
```

Detecting Key Phrases Using the AWS SDK for .NET

The .NET example in this section uses the [AWS SDK for .NET](#). You can use the [AWS Toolkit for Visual Studio](#) to develop AWS applications using .NET. It includes helpful templates and the AWS Explorer for deploying applications and managing services. For a .NET developer perspective of AWS, see the [AWS Guide for .NET Developers](#).

```
using System;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

namespace Comprehend
{
    class Program
    {
        static void Main(string[] args)
        {
            String text = "It is raining today in Seattle";

            AmazonComprehendClient comprehendClient = new
            AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectKeyPhrases");
            DetectKeyPhrasesRequest detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()
            {
                Text = text,
                LanguageCode = "en"
            };
            DetectKeyPhrasesResponse detectKeyPhrasesResponse =
            comprehendClient.DetectKeyPhrases(detectKeyPhrasesRequest);
            foreach (KeyPhrase kp in detectKeyPhrasesResponse.KeyPhrases)
                Console.WriteLine("Text: {1}, Type: {1}, BeginOffset: {2}, EndOffset: {3}",
                    kp.Text, kp.Text, kp.BeginOffset, kp.EndOffset);
            Console.WriteLine("Done");
        }
    }
}
```

Detecting Personally Identifiable Information (PII)

To detect entities that contain personally identifiable information (PII) in a document, use the Amazon Comprehend [DetectPiiEntities](#) ([p. 301](#)) operation.

Detecting PII Using the AWS Command Line Interface

The following example uses the `DetectPiiEntities` operation with the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend detect-pii-entities \
--text "Hello Paul Santos. The latest statement for your credit card \
account 1111-0000-1111-0000 was mailed to 123 Any Street, Seattle, WA \
98109." \
--language-code en
```

Amazon Comprehend responds with the following:

```
{  
    "Entities": [  
        {  
            "Score": 0.9999669790267944,  
            "Type": "NAME",  
            "BeginOffset": 6,  
            "EndOffset": 18  
        },  
        {  
            "Score": 0.8905550241470337,  
            "Type": "CREDIT_DEBIT_NUMBER",  
            "BeginOffset": 69,  
            "EndOffset": 88  
        },  
        {  
            "Score": 0.9999889731407166,  
            "Type": "ADDRESS",  
            "BeginOffset": 103,  
            "EndOffset": 138  
        }  
    ]  
}
```

Labeling Documents with Personally Identifiable Information (PII)

To label documents with personally identifiable information (PII), use the Amazon Comprehend [ContainsPiiEntities](#) (p. 238) operation.

Labeling Documents with PII Using the AWS Command Line Interface

The following example uses the `ContainsPiiEntities` operation with the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend contains-pii-entities \
--text "Hello Paul Santos. The latest statement for your credit card \
account 1111-0000-1111-0000 was mailed to 123 Any Street, Seattle, WA \
98109." \
--language-code en
```

Amazon Comprehend responds with the following:

```
{  
    "Labels": [  
        {  
            "Name": "NAME",  
            "Score": 0.9149109721183777
```

```
        },
        {
            "Name": "CREDIT_DEBIT_NUMBER",
            "Score": 0.8905550241470337
        }
    ],
    {
        "Name": "ADDRESS",
        "Score": 0.9951046109199524
    }
}
```

Detecting Sentiment

To determine the overall emotional tone of text, use the [DetectSentiment \(p. 303\)](#) operation. To detect the sentiment in up to 25 documents in a batch, use the [BatchDetectSentiment \(p. 229\)](#) operation. For more information, see [Using the Batch APIs \(p. 84\)](#).

Topics

- [Detecting Sentiment Using the AWS Command Line Interface \(p. 63\)](#)
- [Detecting Sentiment Using the AWS SDK for Java \(p. 63\)](#)
- [Detecting Sentiment Using the AWS SDK for Python \(Boto\) \(p. 64\)](#)
- [Detecting Sentiment Using the AWS SDK for .NET \(p. 64\)](#)

Detecting Sentiment Using the AWS Command Line Interface

The following example demonstrates using the `DetectSentiment` operation with the AWS CLI. This example specifies the language of the input text.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend detect-sentiment \
--region region \
--language-code "en" \
--text "It is raining today in Seattle."
```

Amazon Comprehend responds with the following:

```
{
    "SentimentScore": {
        "Mixed": 0.014585512690246105,
        "Positive": 0.31592071056365967,
        "Neutral": 0.5985543131828308,
        "Negative": 0.07093945890665054
    },
    "Sentiment": "NEUTRAL",
    "LanguageCode": "en"
}
```

Detecting Sentiment Using the AWS SDK for Java

The following example Java program detects the sentiment of input text. You must specify the language of the input text.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
```

```
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.comprehend.AmazonComprehend;
import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;
import com.amazonaws.services.comprehend.model.DetectSentimentRequest;
import com.amazonaws.services.comprehend.model.DetectSentimentResult;

public class App
{
    public static void main( String[] args )
    {

        String text = "It is raining today in Seattle";

        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWSCredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();

        AmazonComprehend comprehendClient =
            AmazonComprehendClientBuilder.standard()
                .withCredentials(awsCreds)
                .withRegion("region")
                .build();

        // Call detectSentiment API
        System.out.println("Calling DetectSentiment");
        DetectSentimentRequest detectSentimentRequest = new
        DetectSentimentRequest().withText(text)

        .withLanguageCode("en");
        DetectSentimentResult detectSentimentResult =
        comprehendClient.detectSentiment(detectSentimentRequest);
        System.out.println(detectSentimentResult);
        System.out.println("End of DetectSentiment\n");
        System.out.println("Done" );
    }
}
```

Detecting Sentiment Using the AWS SDK for Python (Boto)

The following Python program detects the sentiment of input text. You must specify the language of the input text.

```
import boto3
import json

comprehend = boto3.client(service_name='comprehend', region_name='region')

text = "It is raining today in Seattle"

print('Calling DetectSentiment')
print(json.dumps(comprehend.detect_sentiment(Text=text, LanguageCode='en'), sort_keys=True,
    indent=4))
print('End of DetectSentiment\n')
```

Detecting Sentiment Using the AWS SDK for .NET

The .NET example in this section uses the [AWS SDK for .NET](#). You can use the [AWS Toolkit for Visual Studio](#) to develop AWS applications using .NET. It includes helpful templates and the AWS Explorer for deploying applications and managing services. For a .NET developer perspective of AWS, see the [AWS Guide for .NET Developers](#).

The .NET example in this section uses the AWS SDK for .NET.

```
using System;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

namespace Comprehend
{
    class Program
    {
        static void Main(string[] args)
        {
            String text = "It is raining today in Seattle";

            AmazonComprehendClient comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectSentiment");
            DetectSentimentRequest detectSentimentRequest = new DetectSentimentRequest()
            {
                Text = text,
                LanguageCode = "en"
            };
            DetectSentimentResponse detectSentimentResponse =
comprehendClient.DetectSentiment(detectSentimentRequest);
            Console.WriteLine(detectSentimentResponse.Sentiment);
            Console.WriteLine("Done");
        }
    }
}
```

Detecting Syntax

To parse text to extract the individual words and determine the parts of speech for each word, use the [DetectSyntax \(p. 306\)](#) operation. To parse the syntax of up to 25 documents in a batch, use the [BatchDetectSyntax \(p. 232\)](#) operation. For more information, see [Using the Batch APIs \(p. 84\)](#).

Topics

- [Detecting Syntax Using the AWS Command Line Interface. \(p. 65\)](#)
- [Detecting Syntax Using the AWS SDK for Java \(p. 67\)](#)
- [Detecting Parts of Speech Using the AWS SDK for Python \(Boto\) \(p. 67\)](#)
- [Detecting Syntax Using the AWS SDK for .NET \(p. 68\)](#)

Detecting Syntax Using the AWS Command Line Interface.

The following example demonstrates using the `DetectSyntax` operation with the AWS CLI. This example specifies the language of the input text.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend detect-syntax \
--region region \
--language-code "en" \
--text "It is raining today in Seattle."
```

Amazon Comprehend responds with the following:

```
{
```

```
"SyntaxTokens": [
    {
        "Text": "It",
        "EndOffset": 2,
        "BeginOffset": 0,
        "PartOfSpeech": {
            "Tag": "PRON",
            "Score": 0.8389829397201538
        },
        "TokenId": 1
    },
    {
        "Text": "is",
        "EndOffset": 5,
        "BeginOffset": 3,
        "PartOfSpeech": {
            "Tag": "AUX",
            "Score": 0.9189288020133972
        },
        "TokenId": 2
    },
    {
        "Text": "raining",
        "EndOffset": 13,
        "BeginOffset": 6,
        "PartOfSpeech": {
            "Tag": "VERB",
            "Score": 0.9977611303329468
        },
        "TokenId": 3
    },
    {
        "Text": "today",
        "EndOffset": 19,
        "BeginOffset": 14,
        "PartOfSpeech": {
            "Tag": "NOUN",
            "Score": 0.9993606209754944
        },
        "TokenId": 4
    },
    {
        "Text": "in",
        "EndOffset": 22,
        "BeginOffset": 20,
        "PartOfSpeech": {
            "Tag": "ADP",
            "Score": 0.9999061822891235
        },
        "TokenId": 5
    },
    {
        "Text": "Seattle",
        "EndOffset": 30,
        "BeginOffset": 23,
        "PartOfSpeech": {
            "Tag": "PROPN",
            "Score": 0.9940338730812073
        },
        "TokenId": 6
    },
    {
        "Text": ".",
        "EndOffset": 31,
        "BeginOffset": 30,
        "PartOfSpeech": {

```

```
        "Tag": "PUNCT",
        "Score": 0.9999997615814209
    },
    "TokenId": 7
}
]
```

Detecting Syntax Using the AWS SDK for Java

The following Java program detects the syntax of the input text. You must specify the language of the input text.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.comprehend.AmazonComprehend;
import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;
import com.amazonaws.services.comprehend.model.DetectSyntaxRequest;
import com.amazonaws.services.comprehend.model.DetectSyntaxResult;

public class App
{
    public static void main( String[] args )
    {

        String text = "It is raining today in Seattle.";
        String region = "region"

        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWSCredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();

        AmazonComprehend comprehendClient =
            AmazonComprehendClientBuilder.standard()
                .withCredentials(awsCreds)
                .withRegion(region)
                .build();

        // Call detectSyntax API
        System.out.println("Calling DetectSyntax");
        DetectSyntaxRequest detectSyntaxRequest = new DetectSyntaxRequest()
            .withText(text)
            .withLanguageCode("en");
        DetectSyntaxResult detectSyntaxResult =
            comprehendClient.detectSyntax(detectSyntaxRequest);
        detectSyntaxResult.getSyntaxTokens().forEach(System.out::println);
        System.out.println("End of DetectSyntax\n");
        System.out.println( "Done" );
    }
}
```

Detecting Parts of Speech Using the AWS SDK for Python (Boto)

The following Python program detects the parts of speech in the input text. You must specify the language of the input text.

```
import boto3
import json

comprehend = boto3.client(service_name='comprehend', region_name='region')
text = "It is raining today in Seattle"
```

```
print('Calling DetectSyntax')
print(json.dumps(comprehend.detect_syntax(Text=text, LanguageCode='en'), sort_keys=True,
    indent=4))
print('End of DetectSyntax\n')
```

Detecting Syntax Using the AWS SDK for .NET

The .NET example in this section uses the [AWS SDK for .NET](#). You can use the [AWS Toolkit for Visual Studio](#) to develop AWS applications using .NET. It includes helpful templates and the AWS Explorer for deploying applications and managing services. For a .NET developer perspective of AWS, see the [AWS Guide for .NET Developers](#).

```
using System;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

namespace Comprehend
{
    class Program
    {
        static void Main(string[] args)
        {
            String text = "It is raining today in Seattle";

            AmazonComprehendClient comprehendClient = new
                AmazonComprehendClient(Amazon.RegionEndpoint.region);

            // Call DetectSyntax API
            Console.WriteLine("Calling DetectSyntax\n");
            DetectSyntaxRequest detectSyntaxRequest = new DetectSyntaxRequest()
            {
                Text = text,
                LanguageCode = "en"
            };
            DetectSyntaxResponse detectSyntaxResponse =
                comprehendClient.DetectSyntax(detectSyntaxRequest);
            foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
            {
                Console.WriteLine("Text: {0}, PartOfSpeech: {1}, Score: {2}, BeginOffset: {3},
                    EndOffset: {4}",
                    s.Text, s.PartOfSpeech, s.Score, s.BeginOffset, s.EndOffset);
                Console.WriteLine("Done");
            }
        }
    }
}
```

Using Custom Classification

To create and train a custom classifier, use the Amazon Comprehend [the section called “CreateDocumentClassifier” \(p. 240\)](#). To identify custom classifiers in a corpus of documents, use the [the section called “StartDocumentClassificationJob” \(p. 349\)](#) operation.

Topics

- [Using Custom Classification With the AWS Command Line Interface \(p. 69\)](#)
- [Using Custom Classification Using the AWS SDK for Java \(p. 70\)](#)
- [Using Custom Classification Using the AWS SDK for Python \(Boto\) \(p. 71\)](#)

Using Custom Classification With the AWS Command Line Interface

The following examples demonstrates using the `CreateDocumentClassifier` operation, `StartDocumentClassificationJob` operation, and other custom classifier APIs with the AWS CLI.

The examples are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

Create a custom classifier using the `create-document-classifier` operation.

```
aws comprehend create-document-classifier \
    --region region \
    --document-classifier-name testDelete \
    --language-code en \
    --input-data-config S3Uri=s3://S3Bucket/docclass/file name \
    --data-access-role-arn arn:aws:iam::account number:role/testDeepInsightDataAccess
```

Get information on a custom classifier with the document classifier arn using the `DescribeDocumentClassifier` operation.

```
aws comprehend describe-document-classifier \
    --region region \
    --document-classifier-arn arn:aws:comprehend:region:account number:document-
    classifier/file name
```

Delete a custom classifier using the `DeleteDocumentClassifier` operation.

```
aws comprehend delete-document-classifier \
    --region region \
    --document-classifier-arn arn:aws:comprehend:region:account number:document-
    classifier/testDelete
```

List all custom classifiers in the account using the `ListDocumentClassifiers` operation.

```
aws comprehend list-document-classifiers
    --region region
```

Run a custom classification job using the `StartDocumentClassificationJob` operation.

```
aws comprehend start-document-classification-job \
    --region region \
    --document-classifier-arn arn:aws:comprehend:region:account number:document-
    classifier/testDelete \
    --input-data-config S3Uri=s3://S3Bucket/docclass/file
    name,InputFormat=ONE_DOC_PER_LINE \
    --output-data-config S3Uri=s3://S3Bucket/output \
    --data-access-role-arn arn:aws:iam::account number:role/resource name
```

Get information on a custom classifier with the job id using the `DescribeDocumentClassificationJob` operation.

```
aws comprehend describe-document-classification-job \
    --region region \
```

```
--job-id job id
```

List all custom classification jobs in your account using the `ListDocumentClassificationJobs` operation.

```
aws comprehend list-document-classification-jobs  
--region region
```

Create an endpoint associated with a specific custom model using the `CreateEndpoint` operation.

```
aws comprehend create-endpoint \  
--desired-inference-units number of inference units \  
--endpoint-name endpoint name \  
--model-arn arn:aws:comprehend:region:account-id:model/example \  
--tags Key=My1stTag,Value=Value1
```

Run a custom classification request using an endpoint by using the `ClassifyDocument` operation.

```
aws comprehend classify-document \  
--endpoint-arn arn:aws:comprehend:region:account-id:endpoint/endpoint name \  
--text 'text.'
```

Update an endpoint using the `UpdateEndpoint` operation.

```
aws comprehend update-endpoint \  
--desired-inference-units updated number of inference units \  
--endpoint-arn arn:aws:comprehend:region:account-id:endpoint/endpoint name
```

Delete an endpoint using the `DeleteEndpoint` operation.

```
aws comprehend delete-endpoint \  
--endpoint-arn arn:aws:comprehend:region:account-id:endpoint/endpoint name
```

Get information on an endpoint with the endpoint Arn using the `DescribeEndpoint` operation.

```
aws comprehend describe-endpoint \  
--endpoint-arn arn:aws:comprehend:region:account-id:endpoint/endpoint name
```

List all endpoints in your account using the `ListEndpoints` operation.

```
aws comprehend list-endpoint \  
--filter status=Ready \  
--max-results 50
```

Using Custom Classification Using the AWS SDK for Java

This example creates a custom classifier and trains it using Java

```
import com.amazonaws.services.comprehend.AmazonComprehend;  
  
import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;  
import com.amazonaws.services.comprehend.model.CreateDocumentClassifierRequest;  
import com.amazonaws.services.comprehend.model.CreateDocumentClassifierResult;
```

```
import com.amazonaws.services.comprehend.model.DescribeDocumentClassifierRequest;
import com.amazonaws.services.comprehend.model.DescribeDocumentClassifierResult;
import com.amazonaws.services.comprehend.model.DocumentClassifierInputDataConfig;
import com.amazonaws.services.comprehend.model.LanguageCode;
import com.amazonaws.services.comprehend.model.ListDocumentClassifiersRequest;
import com.amazonaws.services.comprehend.model.ListDocumentClassifiersResult;

public class DocumentClassifierDemo {

    public static void main(String[] args) {
        final AmazonComprehend comprehendClient =
            AmazonComprehendClientBuilder.standard()
                .withRegion("us-west-2")
                .build();

        final String dataAccessRoleArn = "arn:aws:iam::account number:role/resource name";

        final CreateDocumentClassifierRequest createDocumentClassifierRequest = new
CreateDocumentClassifierRequest()
            .withDocumentClassifierName("SampleCodeClassifier")
            .withDataAccessRoleArn(dataAccessRoleArn)
            .withLanguageCode(LanguageCode.En)
            .withInputDataConfig(new DocumentClassifierInputDataConfig()
                .withS3Uri("s3://$3Bucket/docclass/file name"));

        final CreateDocumentClassifierResult createDocumentClassifierResult =
            comprehendClient.createDocumentClassifier(createDocumentClassifierRequest);
        final String documentClassifierArn =
createDocumentClassifierResult.getDocumentClassifierArn();

        System.out.println("Document Classifier ARN: " + documentClassifierArn);

        final DescribeDocumentClassifierRequest describeDocumentClassifierRequest = new
DescribeDocumentClassifierRequest()
            .withDocumentClassifierArn(documentClassifierArn);
        final DescribeDocumentClassifierResult describeDocumentClassifierResult =
comprehendClient.describeDocumentClassifier(describeDocumentClassifierRequest);
        System.out.println("DescribeDocumentClassifierResult: " +
describeDocumentClassifierResult);

        final ListDocumentClassifiersRequest listDocumentClassifiersRequest = new
ListDocumentClassifiersRequest();

        final ListDocumentClassifiersResult listDocumentClassifiersResult =
comprehendClient
            .listDocumentClassifiers(listDocumentClassifiersRequest);
        System.out.println("ListDocumentClassifierResult: " +
listDocumentClassifiersResult );
    }
}
```

Using Custom Classification Using the AWS SDK for Python (Boto)

This example creates a custom classifier and trains it using Python

```
import boto3

# Instantiate Boto3 SDK:
client = boto3.client('comprehend', region_name='region')

# Create a document classifier
```

```
create_response = client.create_document_classifier(
    InputDataConfig={
        'S3Uri': 's3://$3Bucket/docclass/file name'
    },
    DataAccessRoleArn='arn:aws:iam::account number:role/resource name',
    DocumentClassifierName='SampleCodeClassifier1',
    LanguageCode='en'
)
print("Create response: %s\n", create_response)

# Check the status of the classifier
describe_response = client.describe_document_classifier(
    DocumentClassifierArn=create_response['DocumentClassifierArn'])
print("Describe response: %s\n", describe_response)

# List all classifiers in account
list_response = client.list_document_classifiers()
print("List response: %s\n", list_response)
```

This example runs a custom classifier job using Python

```
import boto3

# Instantiate Boto3 SDK:
client = boto3.client('comprehend', region_name='region')

start_response = client.start_document_classification_job(
    InputDataConfig={
        'S3Uri': 's3://srikad-us-west-2-input/docclass/file name',
        'InputFormat': 'ONE_DOC_PER_LINE'
    },
    OutputDataConfig={
        'S3Uri': 's3://$3Bucket/output'
    },
    DataAccessRoleArn='arn:aws:iam::account number:role/resource name',
    DocumentClassifierArn=
        'arn:aws:comprehend:region:account number:document-classifier/SampleCodeClassifier1'
)
print("Start response: %s\n", start_response)

# Check the status of the job
describe_response =
    client.describe_document_classification_job(JobId=start_response['JobId'])
print("Describe response: %s\n", describe_response)

# List all classification jobs in account
list_response = client.list_document_classification_jobs()
print("List response: %s\n", list_response)
```

Detecting Custom Entities

To create the custom entities in a document, use the Amazon Comprehend [CreateEntityRecognizer](#) (p. 249) to create an entity recognizer. To identify those custom entities, use the [StartEntitiesDetectionJob](#) (p. 359) operation.

Topics

- [Creating and Detecting Custom Entities Using the AWS Command Line Interface \(p. 73\)](#)
- [Detecting Custom Entities Using the AWS SDK for Java \(p. 73\)](#)
- [Detecting Custom Entities Using the AWS SDK for Python \(Boto3\) \(p. 75\)](#)

Creating and Detecting Custom Entities Using the AWS Command Line Interface

The following examples demonstrates using the `CreateEntityRecognizer` operation, `StartEntitiesDetectionJob` operation and other associated APIs with the AWS CLI.

The examples are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

Creating a custom entity recognizer using the `CreateEntityRecognizer` operation.

```
aws comprehend create-entity-recognizer \
    --language-code en \
    --recognizer-name test-6 \
    --data-access-role-arn "arn:aws:iam::account number:role/service-role/
AmazonComprehendServiceRole-role" \
    --input-data-config "EntityType=[{Type=PERSON}],Documents={S3Uri=s3://Bucket
Name/Bucket Path/documents},
Annotations={S3Uri=s3://Bucket Name/Bucket Path/annotations}" \
    --region region
```

Listing all entity recognizers in a region using the `ListEntityRecognizers` operation.

```
aws comprehend list-entity-recognizers \
    --region region
```

Checking Job Status of custom entity recognizers using the `DescribeEntityRecognizer` operation.

```
aws comprehend describe-entity-recognizer \
    --entity-recognizer-arn arn:aws:comprehend:region:account number:entity-recognizer/
test-6 \
    --region region
```

Starting a custom entities recognition job using the `StartEntitiesDetectionJob` operation.

```
aws comprehend start-entities-detection-job \
    --entity-recognizer-arn "arn:aws:comprehend:region:account number:entity-recognizer/
test-6" \
    --job-name infer-1 \
    --data-access-role-arn "arn:aws:iam::account number:role/service-role/
AmazonComprehendServiceRole-role" \
    --language-code en \
    --input-data-config "S3Uri=s3://Bucket Name/Bucket Path" \
    --output-data-config "S3Uri=s3://Bucket Name/Bucket Path/" \
    --region region
```

Detecting Custom Entities Using the AWS SDK for Java

This example creates a custom entity recognizer, trains the model, and then runs it in an entity recognizer job using Java

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.comprehend.AmazonComprehend;
import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;
import com.amazonaws.services.comprehend.model.CreateEntityRecognizerRequest;
```

```

import com.amazonaws.services.comprehend.model.CreateEntityRecognizerResult;
import com.amazonaws.services.comprehend.model.DescribeEntityRecognizerRequest;
import com.amazonaws.services.comprehend.model.DescribeEntityRecognizerResult;
import com.amazonaws.services.comprehend.model.EntityRecognizerAnnotations;
import com.amazonaws.services.comprehend.model.EntityRecognizerDocuments;
import com.amazonaws.services.comprehend.model.EntityRecognizerInputDataConfig;
import com.amazonaws.services.comprehend.model.EntityTypesListItem;
import com.amazonaws.services.comprehend.model.InputDataConfig;
import com.amazonaws.services.comprehend.model.LanguageCode;
import com.amazonaws.services.comprehend.model.OutputDataConfig;
import com.amazonaws.services.comprehend.model.StartEntitiesDetectionJobRequest;
import com.amazonaws.services.comprehend.model.StartEntitiesDetectionJobResult;

public class CustomEntityRecognizerDemo {

    public static void main(String[] args) {
        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWS CredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();

        AmazonComprehend comprehendClient =
            AmazonComprehendClientBuilder.standard()
                .withCredentials(awsCreds)
                .withRegion("region")
                .build();

        final String dataAccessRoleArn = "arn:aws:iam::account number:role/service-role/
AmazonComprehendServiceRole-role";

        final CreateEntityRecognizerRequest createEntityRecognizerRequest = new
CreateEntityRecognizerRequest()
            .withRecognizerName("recognizer name")
            .withDataAccessRoleArn(dataAccessRoleArn)
            .withLanguageCode(LanguageCode.En)
            .withInputDataConfig(new EntityRecognizerInputDataConfig()
                .withEntityTypes(new EntityTypesListItem().withType("PERSON"))
                .withDocuments(new EntityRecognizerDocuments()
                    .withS3Uri("s3://Bucket Name/Bucket Path/documents"))
                .withAnnotations(new EntityRecognizerAnnotations()
                    .withS3Uri("s3://Bucket Name/Bucket Path/annotations"))));

        final CreateEntityRecognizerResult createEntityRecognizerResult =
comprehendClient.createEntityRecognizer(createEntityRecognizerRequest);
        final String entityRecognizerArn =
createEntityRecognizerResult.getEntityRecognizerArn();
        System.out.println("Entity Recognizer ARN: " + entityRecognizerArn);

        DescribeEntityRecognizerRequest describeEntityRecognizerRequest = new
DescribeEntityRecognizerRequest()
            .withEntityRecognizerArn(entityRecognizerArn);
        final DescribeEntityRecognizerResult describeEntityRecognizerResult =
comprehendClient.describeEntityRecognizer(describeEntityRecognizerRequest);
        System.out.println("describeEntityRecognizerResult: " +
describeEntityRecognizerResult);

        if
("TRAINED".equals(describeEntityRecognizerResult.getEntityRecognizerProperties().getStatus()))
{
    // After model gets trained, launch an job to extract entities.
    final StartEntitiesDetectionJobRequest startEntitiesDetectionJobRequest = new
StartEntitiesDetectionJobRequest()
        .withJobName("Inference Job Name")
        .withEntityRecognizerArn(entityRecognizerArn)
        .withDataAccessRoleArn(dataAccessRoleArn)
        .withLanguageCode(LanguageCode.En)
        .withInputDataConfig(new InputDataConfig())
}
}

```

```
        .withS3Uri("s3://Bucket Name/Bucket Path"))
    .withOutputDataConfig(new OutputDataConfig()
        .withS3Uri("s3://Bucket Name/Bucket Path")));

    final StartEntitiesDetectionJobResult startEntitiesDetectionJobResult =
comprehendClient.startEntitiesDetectionJob(startEntitiesDetectionJobRequest);
    System.out.println("startEntitiesDetectionJobResult: " +
startEntitiesDetectionJobResult);
}
}
```

Detecting Custom Entities Using the AWS SDK for Python (Boto3)

Instantiate Boto3 SDK:

```
import boto3
import uuid
comprehend = boto3.client("comprehend", region_name="recognizer name")
```

Create entity recognizer:

```
response = comprehend.create_entity_recognizer(
    RecognizerName="Recognizer-Name- Goes-Here-{}".format(str(uuid.uuid4())),
    LanguageCode="en",
    DataAccessRoleArn="Role ARN",
    InputDataConfig={
        "EntityTypes": [
            {
                "Type": "ENTITY_TYPE"
            }
        ],
        "Documents": {
            "S3Uri": "s3://Bucket Name/Bucket Path/documents"
        },
        "Annotations": {
            "S3Uri": "s3://Bucket Name/Bucket Path/annotations"
        }
    }
)
recognizer_arn = response["EntityRecognizerArn"]
```

List all recognizers:

```
response = comprehend.list_entity_recognizers()
```

Wait for recognizer to reach TRAINED status:

```
while True:
    response = comprehend.describe_entity_recognizer(
        EntityRecognizerArn=recognizer_arn
    )

    status = response["EntityRecognizerProperties"]["Status"]
    if "IN_ERROR" == status:
        sys.exit(1)
    if "TRAINED" == status:
```

```
    break
    time.sleep(10)
```

Start entities detection job:

```
response = comprehend.start_entities_detection_job(
    EntityRecognizerArn=recognizer_arn,
    JobName="Detection-Job-Name-{}".format(str(uuid.uuid4())),
    LanguageCode="en",
    DataAccessRoleArn="Role ARN",
    InputDataConfig={
        "InputFormat": "ONE_DOC_PER_LINE",
        "S3Uri": "s3://Bucket Name/Bucket Path/documents"
    },
    OutputDataConfig={
        "S3Uri": "s3://Bucket Name/Bucket Path/output"
    }
)
```

Detecting Events

To detect events in a document set, use the [StartEventsDetectionJob \(p. 364\)](#) to start an asynchronous job.

Before You Start

Before you start, make sure that you have:

- **Input and output buckets**—Identify the Amazon S3 buckets that you want to use for input and output. The buckets must be in the same region as the API that you are calling.
- **IAM service role**—You must have an IAM service role with permission to access your input and output buckets. For more information, see [Role-Based Permissions Required for Asynchronous Operations \(p. 196\)](#).

Detecting Events Using the AWS Command Line Interface

The following example demonstrates using the [StartEventsDetectionJob \(p. 364\)](#) operation with the AWS CLI

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend start-events-detection-job \
--region region \
--job-name job name \
--cli-input-json file://path to JSON input file
```

For the cli-input-json parameter you supply the path to a JSON file that contains the request data, as shown in the following example.

```
{
    "InputDataConfig": {
        "S3Uri": "s3://input bucket/input path",
        "InputFormat": "ONE_DOC_PER_LINE"
    },
}
```

```

    "OutputDataConfig": {
        "S3Uri": "s3://output bucket/output path"
    },
    "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role"
    "LanguageCode": "en",
    "TargetEventTypes": [
        "BANKRUPTCY",
        "EMPLOYMENT",
        "CORPORATE_ACQUISITION",
        "INVESTMENT_GENERAL",
        "CORPORATE_MERGER",
        "IPO",
        "RIGHTS_ISSUE",
        "SECONDARY_OFFERING",
        "SHELF_OFFERING",
        "TENDER_OFFERING",
        "STOCK_SPLIT"
    ]
}

```

If the request to start the events detection job was successful, you will receive the following response:

```
{
    "JobStatus": "SUBMITTED",
    "JobId": "job ID"
}
```

Use the [ListEventsDetectionJobs \(p. 332\)](#) operation to see a list of the events detection jobs that you have submitted. The list includes information about the input and output locations that you used as well as the status of each of the detection jobs. The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend list-events-detection-jobs --region region
```

You will get JSON similar to the following in response:

```
{
    "EventsDetectionJobPropertiesList": [
        {
            "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role",
            "EndTime": timestamp,
            "InputDataConfig": {
                "InputFormat": "ONE_DOC_PER_LINE",
                "S3Uri": "s3://input bucket/input path"
            },
            "JobId": "job ID",
            "JobName": "job name",
            "JobStatus": "COMPLETED",
            "LanguageCode": "en",
            "Message": "message",
            "OutputDataConfig": {
                "S3Uri": "s3://output bucket/ouput path"
            },
            "SubmitTime": timestamp,
            "TargetEventTypes": [
                "BANKRUPTCY",
                "EMPLOYMENT",
                "CORPORATE_ACQUISITION",
                "INVESTMENT_GENERAL",
                "CORPORATE_MERGER",
                "IPO",
                "RIGHTS_ISSUE",

```

```
        "SECONDARY_OFFERING",
        "SHELF_OFFERING",
        "TENDER_OFFERING",
        "STOCK_SPLIT"
    ],
},
"NextToken": "next token"
}
```

You can use the [DescribeEventsDetectionJob \(p. 277\)](#) operation to get the status of an existing job. The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend describe-events-detection-job \
--region region \
--job-id job ID
```

You will get the following JSON in response:

```
{
    "EventsDetectionJobProperties": {
        "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role",
        "EndTime": timestamp,
        "InputDataConfig": {
            "InputFormat": "ONE_DOC_PER_LINE",
            "S3Uri": "S3Uri": "s3://input bucket/input path"
        },
        "JobId": "job ID",
        "JobName": "job name",
        "JobStatus": "job status",
        "LanguageCode": "en",
        "Message": "message",
        "OutputDataConfig": {
            "S3Uri": "s3://output bucket/output path"
        },
        "SubmitTime": timestamp,
        "TargetEventTypes": [
            "BANKRUPTCY",
            "EMPLOYMENT",
            "CORPORATE_ACQUISITION",
            "INVESTMENT_GENERAL",
            "CORPORATE_MERGER",
            "IPO",
            "RIGHTS_ISSUE",
            "SECONDARY_OFFERING",
            "SHELF_OFFERING",
            "TENDER_OFFERING",
            "STOCK_SPLIT"
        ]
    }
}
```

Topic Modeling

To determine the topics in a document set, use the [StartTopicsDetectionJob \(p. 382\)](#) to start an asynchronous job. You can monitor topics in documents written in English or Spanish.

Topics

- [Before You Start \(p. 79\)](#)

- [Topic Modeling Using the AWS Command Line Interface \(p. 79\)](#)
- [Topic Modeling Using the AWS SDK for Java \(p. 81\)](#)
- [Topic Modeling Using the AWS SDK for Python \(Boto\) \(p. 82\)](#)
- [Topic Modeling Using the AWS SDK for .NET \(p. 83\)](#)

Before You Start

Before you start, make sure that you have:

- **Input and output buckets**—Identify the Amazon S3 buckets that you want to use for input and output. The buckets must be in the same region as the API that you are calling.
- **IAM service role**—You must have an IAM service role with permission to access your input and output buckets. For more information, see [Role-Based Permissions Required for Asynchronous Operations \(p. 196\)](#).

Topic Modeling Using the AWS Command Line Interface

The following example demonstrates using the `StartTopicsDetectionJob` operation with the AWS CLI

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend start-topics-detection-job \
    --number-of-topics topics to return \
    --job-name "job name" \
    --region region \
    --cli-input-json file:///path to JSON input file
```

For the `cli-input-json` parameter you supply the path to a JSON file that contains the request data, as shown in the following example.

```
{
    "InputDataConfig": {
        "S3Uri": "s3://input bucket/input path",
        "InputFormat": "ONE_DOC_PER_FILE"
    },
    "OutputDataConfig": {
        "S3Uri": "s3://output bucket/output path"
    },
    "DataAccessRoleArn": "arn:aws:iam::account ID:role/data access role"
}
```

If the request to start the topic detection job was successful, you will receive the following response:

```
{
    "JobStatus": "SUBMITTED",
    "JobId": "job ID"
}
```

Use the [ListTopicsDetectionJobs \(p. 346\)](#) operation to see a list of the topic detection jobs that you have submitted. The list includes information about the input and output locations that you used as well as the status of each of the detection jobs. The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend list-topics-detection-jobs \-- region
```

You will get JSON similar to the following in response:

```
{  
    "TopicsDetectionJobPropertiesList": [  
        {  
            "InputDataConfig": {  
                "S3Uri": "s3://input bucket/input path",  
                "InputFormat": "ONE_DOC_PER_LINE"  
            },  
            "NumberOfTopics": topics to return,  
            "JobId": "job ID",  
            "JobStatus": "COMPLETED",  
            "JobName": "job name",  
            "SubmitTime": timestamp,  
            "OutputDataConfig": {  
                "S3Uri": "s3://output bucket/output path"  
            },  
            "EndTime": timestamp  
        },  
        {  
            "InputDataConfig": {  
                "S3Uri": "s3://input bucket/input path",  
                "InputFormat": "ONE_DOC_PER_LINE"  
            },  
            "NumberOfTopics": topics to return,  
            "JobId": "job ID",  
            "JobStatus": "RUNNING",  
            "JobName": "job name",  
            "SubmitTime": timestamp,  
            "OutputDataConfig": {  
                "S3Uri": "s3://output bucket/output path"  
            }  
        }  
    ]  
}
```

You can use the [DescribeTopicsDetectionJob \(p. 288\)](#) operation to get the status of an existing job. The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend describe-topics-detection-job --job-id job ID
```

You will get the following JSON in response:

```
{  
    "TopicsDetectionJobProperties": {  
        "InputDataConfig": {  
            "S3Uri": "s3://input bucket/input path",  
            "InputFormat": "ONE_DOC_PER_LINE"  
        },  
        "NumberOfTopics": topics to return,  
        "JobId": "job ID",  
        "JobStatus": "COMPLETED",  
        "JobName": "job name",  
        "SubmitTime": timestamp,  
        "OutputDataConfig": {  
            "S3Uri": "s3://output bucket/output path"  
        },  
        "EndTime": timestamp  
    }  
}
```

```
}
```

Topic Modeling Using the AWS SDK for Java

The following Java program detects the topics in a document collection. It uses the [StartTopicsDetectionJob \(p. 382\)](#) operation to start detecting topics. Next, it uses the [DescribeTopicsDetectionJob \(p. 288\)](#) operation to check the status of the topic detection. Finally, it calls [ListTopicsDetectionJobs \(p. 346\)](#) to show a list of all jobs submitted for the account.

```
import com.amazonaws.auth.AWS CredentialsProvider;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.comprehend.AmazonComprehend;
import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;
import com.amazonaws.services.comprehend.model.DescribeTopicsDetectionJobRequest;
import com.amazonaws.services.comprehend.model.DescribeTopicsDetectionJobResult;
import com.amazonaws.services.comprehend.model.InputDataConfig;
import com.amazonaws.services.comprehend.model.InputFormat;
import com.amazonaws.services.comprehend.model.ListTopicsDetectionJobsRequest;
import com.amazonaws.services.comprehend.model.ListTopicsDetectionJobsResult;
import com.amazonaws.services.comprehend.model.StartTopicsDetectionJobRequest;
import com.amazonaws.services.comprehend.model.StartTopicsDetectionJobResult;

public class App
{
    public static void main( String[] args )
    {
        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWS CredentialsProvider awsCreds =
DefaultAWSCredentialsProviderChain.getInstance();

        AmazonComprehend comprehendClient =
            AmazonComprehendClientBuilder.standard()
                .withCredentials(awsCreds)
                .withRegion("region")
                .build();

        final String inputS3Uri = "s3://<input bucket>/<input path>";
        final InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
        final String outputS3Uri = "s3://<output bucket>/<output path>";
        final String dataAccessRoleArn = "arn:aws:iam::<account ID>:role/<data access role>";
        final int numberOfTopics = 10;

        final StartTopicsDetectionJobRequest startTopicsDetectionJobRequest = new
StartTopicsDetectionJobRequest()
            .withInputDataConfig(new InputDataConfig()
                .withS3Uri(inputS3Uri)
                .withInputFormat(inputDocFormat))
            .withOutputDataConfig(new OutputDataConfig()
                .withS3Uri(outputS3Uri))
            .withDataAccessRoleArn(dataAccessRoleArn)
            .withNumberOfTopics(numberOfTopics);

        final StartTopicsDetectionJobResult startTopicsDetectionJobResult =
comprehendClient.startTopicsDetectionJob(startTopicsDetectionJobRequest);

        final String jobId = startTopicsDetectionJobResult.getJobId();
        System.out.println("JobId: " + jobId);

        final DescribeTopicsDetectionJobRequest describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
```

```
        .withJobId(jobId);

    final DescribeTopicsDetectionJobResult describeTopicsDetectionJobResult =
comprehendClient.describeTopicsDetectionJob(describeTopicsDetectionJobRequest);
    System.out.println("describeTopicsDetectionJobResult: " +
describeTopicsDetectionJobResult);

    ListTopicsDetectionJobsResult listTopicsDetectionJobsResult =
comprehendClient.listTopicsDetectionJobs(new ListTopicsDetectionJobsRequest());
    System.out.println("listTopicsDetectionJobsResult: " +
listTopicsDetectionJobsResult);

}
}
```

Topic Modeling Using the AWS SDK for Python (Boto)

The following Python program detects the topics in a document collection. It uses the [StartTopicsDetectionJob](#) (p. 382) operation to start detecting topics. Next, it uses the [DescribeTopicsDetectionJob](#) (p. 288) operation to check the status of the topic detection. Finally, it calls [ListTopicsDetectionJobs](#) (p. 346) to show a list of all jobs submitted for the account.

```
import boto3
import json
from bson import json_util

comprehend = boto3.client(service_name='comprehend', region_name='region')

input_s3_url = "s3:///input path"
input_doc_format = "ONE_DOC_PER_FILE"
output_s3_url = "s3:///output path"
data_access_role_arn = "arn:aws:iam::account ID:role/data access role"
number_of_topics = 10

input_data_config = {"S3Uri": input_s3_url, "InputFormat": input_doc_format}
output_data_config = {"S3Uri": output_s3_url}

start_topics_detection_job_result =
comprehend.start_topics_detection_job(NumberOfTopics=number_of_topics,
                                      InputDataConfig=input_data_config,
                                      OutputDataConfig=output_data_config,
                                      DataAccessRoleArn=data_access_role_arn)

print('start_topics_detection_job_result: ' +
      json.dumps(start_topics_detection_job_result))

job_id = start_topics_detection_job_result["JobId"]

print('job_id: ' + job_id)

describe_topics_detection_job_result =
comprehend.describe_topics_detection_job(JobId=job_id)

print('describe_topics_detection_job_result: ' +
      json.dumps(describe_topics_detection_job_result, default=json_util.default))

list_topics_detection_jobs_result = comprehend.list_topics_detection_jobs()

print('list_topics_detection_jobs_result: ' + json.dumps(list_topics_detection_jobs_result,
                                                       default=json_util.default))
```

Topic Modeling Using the AWS SDK for .NET

The following C# program detects the topics in a document collection. It uses the [StartTopicsDetectionJob](#) (p. 382) operation to start detecting topics. Next, it uses the [DescribeTopicsDetectionJob](#) (p. 288) operation to check the status of the topic detection. Finally, it calls [ListTopicsDetectionJobs](#) (p. 346) to show a list of all jobs submitted for the account.

The .NET example in this section uses the [AWS SDK for .NET](#). You can use the [AWS Toolkit for Visual Studio](#) to develop AWS applications using .NET. It includes helpful templates and the AWS Explorer for deploying applications and managing services. For a .NET developer perspective of AWS, see the [AWS Guide for .NET Developers](#).

```
using System;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

namespace Comprehend
{
    class Program
    {
        // Helper method for printing properties
        static private void PrintJobProperties(TopicsDetectionJobProperties props)
        {
            Console.WriteLine("JobId: {0}, JobName: {1}, JobStatus: {2}, NumberOfTopics: {3}\nInputS3Uri: {4}, InputFormat: {5}, OutputS3Uri: {6}",
                props.JobId, props.JobName, props.JobStatus, props.NumberOfTopics,
                props.InputDataConfig.S3Uri, props.InputDataConfig.InputFormat,
                props.OutputDataConfig.S3Uri);
        }

        static void Main(string[] args)
        {
            String text = "It is raining today in Seattle";

            AmazonComprehendClient comprehendClient = new
            AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            String inputS3Uri = "s3://input bucket/input path";
            InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
            String outputS3Uri = "s3://output bucket/output path";
            String dataAccessRoleArn = "arn:aws:iam::account ID:role/data access role";
            int numberOfTopics = 10;

            StartTopicsDetectionJobRequest startTopicsDetectionJobRequest = new
            StartTopicsDetectionJobRequest()
            {
                InputDataConfig = new InputDataConfig()
                {
                    S3Uri = inputS3Uri,
                    InputFormat = inputDocFormat
                },
                OutputDataConfig = new OutputDataConfig()
                {
                    S3Uri = outputS3Uri
                },
                DataAccessRoleArn = dataAccessRoleArn,
                NumberOfTopics = numberOfTopics
            };

            StartTopicsDetectionJobResponse startTopicsDetectionJobResponse =
            comprehendClient.StartTopicsDetectionJob(startTopicsDetectionJobRequest);

            String jobId = startTopicsDetectionJobResponse.JobId;
```

```
Console.WriteLine("JobId: " + jobId);

DescribeTopicsDetectionJobRequest describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
{
    JobId = jobId
};

DescribeTopicsDetectionJobResponse describeTopicsDetectionJobResponse =
comprehendClient.DescribeTopicsDetectionJob(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);

ListTopicsDetectionJobsResponse listTopicsDetectionJobsResponse =
comprehendClient.ListTopicsDetectionJobs(new ListTopicsDetectionJobsRequest());
foreach (TopicsDetectionJobProperties props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
    PrintJobProperties(props);
}
}
```

Using the Batch APIs

To send batches of up to 25 documents, you can use the Amazon Comprehend batch operations. Calling a batch operation is identical to calling the single document APIs for each document in the request. Using the batch APIs can result in better performance for your applications. For more information, see [Multiple Document Synchronous Processing \(p. 122\)](#).

Topics

- [Batch Processing With the SDK for Java \(p. 84\)](#)
- [Batch Processing With the AWS SDK for .NET \(p. 85\)](#)
- [Batch Processing With the AWS CLI \(p. 87\)](#)

Batch Processing With the SDK for Java

The following sample program shows how to use the [BatchDetectEntities \(p. 223\)](#) operation with the SDK for Java. The response from the server contains a [BatchDetectEntitiesItemResult \(p. 414\)](#) object for each document that was successfully processed. If there is an error processing a document there will be a record in the error list in the response. The example gets each of the documents with an error and resends them.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.comprehend.AmazonComprehend;
import com.amazonaws.services.comprehend.AmazonComprehendClientBuilder;
import com.amazonaws.services.comprehend.model.BatchDetectEntitiesItemResult;
import com.amazonaws.services.comprehend.model.BatchDetectEntitiesRequest;
import com.amazonaws.services.comprehend.model.BatchDetectEntitiesResult;
import com.amazonaws.services.comprehend.model.BatchItemError;

public class App
{
    public static void main( String[] args )
    {

        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWSCredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();
```

```

AmazonComprehend comprehendClient =
    AmazonComprehendClientBuilder.standard()
        .withCredentials(awsCreds)
        .withRegion("region")
        .build();

String[] textList = {"I love Seattle", "Today is Sunday", "Tomorrow is Monday", "I
love Seattle"};

// Call detectEntities API
System.out.println("Calling BatchDetectEntities");
BatchDetectEntitiesRequest batchDetectEntitiesRequest = new
BatchDetectEntitiesRequest().withTextList(textList)

.withLanguageCode("en");
BatchDetectEntitiesResult batchDetectEntitiesResult =
client.batchDetectEntities(batchDetectEntitiesRequest);

for(BatchDetectEntitiesItemResult item : batchDetectEntitiesResult.getResultList())
{
    System.out.println(item);
}

// check if we need to retry failed requests
if (batchDetectEntitiesResult.getErrorList().size() != 0)
{
    System.out.println("Retrying Failed Requests");
    ArrayList<String> textToRetry = new ArrayList<String>();
    for(BatchItemError errorItem : batchDetectEntitiesResult.getErrorList())
    {
        textToRetry.add(textList[errorItem.getIndex()]);
    }

    batchDetectEntitiesRequest = new
BatchDetectEntitiesRequest().withTextList(textToRetry).withLanguageCode("en");
    batchDetectEntitiesResult =
client.batchDetectEntities(batchDetectEntitiesRequest);

    for(BatchDetectEntitiesItemResult item :
batchDetectEntitiesResult.getResultList()) {
        System.out.println(item);
    }
}

System.out.println("End of DetectEntities");
}
}

```

Batch Processing With the AWS SDK for .NET

The following sample program shows how to use the [BatchDetectEntities \(p. 223\)](#) operation with the AWS SDK for .NET. The response from the server contains a [BatchDetectEntitiesItemResult \(p. 414\)](#) object for each document that was successfully processed. If there is an error processing a document there will be a record in the error list in the response. The example gets each of the documents with an error and resends them.

The .NET example in this section uses the [AWS SDK for .NET](#). You can use the [AWS Toolkit for Visual Studio](#) to develop AWS applications using .NET. It includes helpful templates and the AWS Explorer for deploying applications and managing services. For a .NET developer perspective of AWS, see the [AWS Guide for .NET Developers](#).

```
using System;
```

```
using System.Collections.Generic;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

namespace Comprehend
{
    class Program
    {
        // Helper method for printing properties
        static private void PrintEntity(Entity entity)
        {
            Console.WriteLine("      Text: {0}, Type: {1}, Score: {2}, BeginOffset: {3}
EndOffset: {4}",
                entity.Text, entity.Type, entity.Score, entity.BeginOffset,
                entity.EndOffset);
        }

        static void Main(string[] args)
        {
            AmazonComprehendClient comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            List<String> textList = new List<String>()
            {
                { "I love Seattle" },
                { "Today is Sunday" },
                { "Tomorrow is Monday" },
                { "I love Seattle" }
            };

            // Call detectEntities API
            Console.WriteLine("Calling BatchDetectEntities");
            BatchDetectEntitiesRequest batchDetectEntitiesRequest = new
BatchDetectEntitiesRequest()
            {
                TextList = textList,
                LanguageCode = "en"
            };
            BatchDetectEntitiesResponse batchDetectEntitiesResponse =
comprehendClient.BatchDetectEntities(batchDetectEntitiesRequest);

            foreach (BatchDetectEntitiesItemResult item in
batchDetectEntitiesResponse.ResultList)
            {
                Console.WriteLine("Entities in {0}:", textList[item.Index]);
                foreach (Entity entity in item.Entities)
                    PrintEntity(entity);
            }

            // check if we need to retry failed requests
            if (batchDetectEntitiesResponse.ErrorList.Count != 0)
            {
                Console.WriteLine("Retrying Failed Requests");
                List<String> textToRetry = new List<String>();
                foreach(BatchItemError errorItem in batchDetectEntitiesResponse.ErrorList)
                    textToRetry.Add(textList[errorItem.Index]);

                batchDetectEntitiesRequest = new BatchDetectEntitiesRequest()
                {
                    TextList = textToRetry,
                    LanguageCode = "en"
                };

                batchDetectEntitiesResponse =
comprehendClient.BatchDetectEntities(batchDetectEntitiesRequest);
            }
        }
    }
}
```

```
        foreach(BatchDetectEntitiesItemResult item in
batchDetectEntitiesResponse.ResultList)
    {
        Console.WriteLine("Entities in {0}:", textList[item.Index]);
        foreach (Entity entity in item.Entities)
            PrintEntity(entity);
    }
    Console.WriteLine("End of DetectEntities");
}
}
```

Batch Processing With the AWS CLI

These examples show how to use the batch API operations using the AWS Command Line Interface. All of the operations except `BatchDetectDominantLanguage` use the following JSON file called `process.json` as input. For that operation the `LanguageCode` entity is not included.

The third document in the JSON file ("\$\$\$\$\$\$\$\$") will cause an error during batch processing. It is included so that the operations will include a [BatchItemError \(p. 418\)](#) in the response.

```
{
    "LanguageCode": "en",
    "TextList": [
        "I have been living in Seattle for almost 4 years",
        "It is raining today in Seattle",
        "$$$$$$$$"
    ]
}
```

The examples are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

Topics

- [Detect the Dominant Language Using a Batch \(AWS CLI\) \(p. 87\)](#)
- [Detect Entities Using a Batch \(AWS CLI\) \(p. 88\)](#)
- [Detect Key Phrases Using a Batch \(AWS CLI\) \(p. 88\)](#)
- [Detect Sentiment Using a Batch \(AWS CLI\) \(p. 88\)](#)

Detect the Dominant Language Using a Batch (AWS CLI)

The [BatchDetectDominantLanguage \(p. 220\)](#) operation determines the dominant language of each document in a batch. For a list of the languages that Amazon Comprehend can detect, see [Detect the Dominant Language \(p. 102\)](#). The following AWS CLI command calls the `BatchDetectDominantLanguage` operation.

```
aws comprehend batch-detect-dominant-language \
--endpoint endpoint \
--region region \
--cli-input-json file:///path to input file/process.json
```

The following is the response from the `BatchDetectDominantLanguage` operation:

```
{
```

```
"ResultList": [
    {
        "Index": 0,
        "Languages": [
            {
                "LanguageCode": "en",
                "Score": 0.99
            }
        ]
    },
    {
        "Index": 1,
        "Languages": [
            {
                "LanguageCode": "en",
                "Score": 0.82
            }
        ]
    }
],
"ErrorList": [
    {
        "Index": 2,
        "ErrorCode": "InternalServerException",
        "ErrorMessage": "Unexpected Server Error. Please try again."
    }
]
}
```

Detect Entities Using a Batch (AWS CLI)

Use the [BatchDetectEntities \(p. 223\)](#) operation to find the entities present in a batch of documents. For more information about entities, see [Detect Entities \(p. 94\)](#). The following AWS CLI command calls the `BatchDetectEntities` operation.

```
aws comprehend batch-detect-entities \
--endpoint endpoint \
--region region \
--cli-input-json file:///path to input file/process.json
```

Detect Key Phrases Using a Batch (AWS CLI)

The [BatchDetectKeyPhrases \(p. 226\)](#) operation returns the key noun phrases in a batch of documents. The following AWS CLI command calls the `BatchDetectKeyNounPhrases` operation.

```
aws comprehend batch-detect-key-phrases
--endpoint endpoint
--region region
--cli-input-json file:///path to input file/process.json
```

Detect Sentiment Using a Batch (AWS CLI)

Detect the overall sentiment of a batch of documents using the [BatchDetectSentiment \(p. 229\)](#) operation. The following AWS CLI command calls the `BatchDetectSentiment` operation.

```
aws comprehend batch-detect-sentiment \
--endpoint endpoint \
--region region \
--cli-input-json file:///path to input file/process.json
```

Solution: Analyzing Text with Amazon Comprehend and Amazon Elasticsearch Service

The amount of unstructured data that we work with daily is growing rapidly. Analysis of customer feedback, business documents, and more has become a very important part of making intelligent business decisions.

To support customers who need these large scale text analysis solutions, Amazon Comprehend combines with Amazon Elasticsearch Service on the AWS Cloud, to provide an end to end solution. Amazon Comprehend provides text analysis and Amazon Elasticsearch Service provides document indexing, searching, and visualization. Together they enable you to analyze text using the scale of Amazon Elasticsearch Service with the semantic analysis ability of Amazon Comprehend.

For more information, see [Analyzing Text with Amazon Elasticsearch Service and Amazon Comprehend](#).

Using Amazon S3 Object Lambda Access Points for Personally Identifiable Information (PII)

Use Amazon S3 Object Lambda Access Points for personally identifiable information (PII) to configure how documents are retrieved from your Amazon S3 bucket. You can control access to documents that contain PII and redact PII from documents. For more information on how Amazon Comprehend can detect PII in your documents, see [Detect Personally Identifiable Information \(PII\) \(p. 106\)](#). Amazon S3 Object Lambda Access Points use AWS Lambda functions to automatically transform the output of a standard Amazon S3 GET request. For more information see, [Transforming objects with S3 Object Lambda](#) in the *Amazon Simple Storage Service User Guide*.

When you create an Amazon S3 Object Lambda Access Point for PII, documents are processed using Amazon Comprehend Lambda functions to control access of documents that contain PII and redact PII from documents.

When you create an Amazon S3 Object Lambda Access Point for PII, documents are processed using the following Amazon Comprehend Lambda functions:

- `ComprehendPiiAccessControlS3ObjectLambda` - Controls access to documents with PII stored in your S3 bucket. For more information about this Lambda function, sign in to the AWS Management Console to view the [ComprehendPiiAccessControlS3ObjectLambda](#) function in the AWS Serverless Application Repository.
- `ComprehendPiiRedactionS3ObjectLambda` - Redacts PII from documents in your Amazon S3 bucket. For more information about this Lambda function, sign in to the AWS Management Console to view the [ComprehendPiiRedactionS3ObjectLambda](#) function in the AWS Serverless Application Repository.

For information about how to deploy serverless applications from the AWS Serverless Application Repository, see [Deploying Applications](#) in the *AWS Serverless Application Repository Developer Guide*.

Topics

- [Controlling Access to Documents with Personally Identifiable Information \(PII\) \(p. 90\)](#)
- [Redacting Personally Identifiable Information \(PII\) from Documents \(p. 92\)](#)

Controlling Access to Documents with Personally Identifiable Information (PII)

You can use an Amazon S3 Object Lambda Access Point to control access to documents with personally identifiable information (PII).

To ensure that only authorized users have access to documents that contain PII stored in your Amazon S3 bucket, you use the `ComprehendPiiAccessControlS3ObjectLambda` function. This Lambda function uses the [ContainsPiiEntities \(p. 238\)](#) operation when processing a standard Amazon S3 GET request on document objects.

For example, if you have documents in your S3 bucket that include PII such as credit card numbers or bank account information, you can configure the `ComprehendPiiAccessControlS3ObjectLambda` function to detect these PII entity types and restrict access to unauthorized users. For more information about supported PII entity types, see [PII Entity Types \(p. 106\)](#).

For more information about this Lambda function, sign in to the AWS Management Console to view the `ComprehendPiiAccessControlS3ObjectLambda` function in the AWS Serverless Application Repository.

Creating an Amazon S3 Object Lambda Access Point to Control Access to Documents

The following example creates an Amazon S3 Object Lambda Access Point to control access to documents that contain social security numbers.

Creating an Amazon S3 Object Lambda Access Point Using the AWS Command Line Interface

Create an Amazon S3 Object Lambda Access Point configuration and save the configuration in a file called `config.json`.

```
{  
    "SupportingAccessPoint": "s3-default-access-point-name-arn",  
    "TransformationConfigurations": [  
        {  
            "Actions": [  
                "s3:GetObject"  
            ],  
            "ContentTransformation": {  
                "AwsLambda": {  
                    "FunctionArn": "comprehend-pii-access-control-s3-object-lambda-arn",  
                    "FunctionPayload": "{\"pii_entities_types\": \"$SSN\"}"  
                }  
            }  
        }  
    ]  
}
```

The following example creates an Amazon S3 Object Lambda Access Point based on the configuration defined in the `config.json` file.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws s3control create-banner-access-point \  
  --region region \  
  --account-id account-id \  
  --name s3-object-lambda-access-point \  
  --configuration file://config.json
```

Invoking an Amazon S3 Object Lambda Access Point to Control Access to Documents

The following example invokes an Amazon S3 Object Lambda Access Point to control access to documents.

Invoking an Amazon S3 Object Lambda Access Point Using the AWS Command Line Interface

The following example invokes an Amazon S3 Object Lambda Access Point using the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws s3api get-object \
--region region \
--bucket s3-object-lambda-access-point-name-arn \
--key object-prefix-key output-file-name
```

Redacting Personally Identifiable Information (PII) from Documents

You can use an Amazon S3 Object Lambda Access Point to redact personally identifiable information (PII) from documents.

To redact PII entity types from documents stored in an S3 bucket, you use the `ComprehendPiiRedactionS3ObjectLambda` function. This Lambda function uses the [ContainsPiiEntities \(p. 238\)](#) and [DetectPiiEntities \(p. 301\)](#) operations when processing a standard Amazon S3 GET request on document objects.

For example, if documents in your S3 bucket include PII such as credit card numbers or bank account information, you can configure the `ComprehendPiiRedactionS3ObjectLambda` function to detect PII and then return a copy of these documents in which PII entity types are redacted. For more information about supported PII entity types, see [PII Entity Types \(p. 106\)](#).

For more information about this Lambda function, sign in to the AWS Management Console to view the `ComprehendPiiRedactionS3ObjectLambda` function in the AWS Serverless Application Repository.

Creating an Amazon S3 Object Lambda Access Point to Redact PII from Documents

The following example creates an Amazon S3 Object Lambda Access Point to redact credit card numbers from documents.

Creating an Amazon S3 Object Lambda Access Point Using the AWS Command Line Interface

Create an Amazon S3 Object Lambda Access Point configuration and save the configuration in a file called `config.json`.

```
{  
    "SupportingAccessPoint": "s3-default-access-point-name-arn",  
    "TransformationConfigurations": [  
        {  
            "Actions": [  
                "s3:GetObject"  
            ],  
            "Resource": "s3-object-lambda-access-point-name-arn"  
        }  
    ]  
}
```

```
        "ContentTransformation": {
            "AwsLambda": {
                "FunctionArn": "comprehend-pii-redaction-s3-object-lambda-arn",
                "FunctionPayload": "{\"pii_entities_types\": \"CREDIT_DEBIT_NUMBER\"}"
            }
        }
    ]
}
```

The following example demonstrates creating an Amazon S3 Object Lambda Access Point based on the configuration defined in the config.json

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws s3control create-access-point-for-object-lambda \
--region region \
--account-id account-id \
--name s3-object-lambda-access-point \
--configuration file://config.json
```

Invoking an Amazon S3 Object Lambda Access Point to Redact PII from Documents

The following examples invoke an Amazon S3 Object Lambda Access Point to redact PII from documents.

Invoking an Amazon S3 Object Lambda Access Point Using the AWS Command Line Interface

The following example invokes an Amazon S3 Object Lambda Access Point using the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws s3api get-object \
--region region \
--bucket s3-object-lambda-access-point-name-arn \
--key object-prefix-key output-file-name
```

Text Analysis APIs

Amazon Comprehend enables you to examine your documents to gain various insights about their content using a number of pre-trained models.

With Amazon Comprehend, you can perform the following on your documents:

- [Detect Entities \(p. 94\)](#)—Detect textual references to the names of people, places, and items as well as references to dates and quantities.
- [Detect Events \(p. 96\)](#)—Analyze documents to detect specific types of events.
- [Detect Key Phrases \(p. 101\)](#)—Find key phrases such as "good morning" in a document or set of documents.
- [Detect the Dominant Language \(p. 102\)](#)—Examine text to determine the dominant language.
- [Detect Personally Identifiable Information \(PII\) \(p. 106\)](#) — Analyze documents to detect personal data that could be used to identify an individual, such as an address, bank account number, or phone number.
- [Label Documents with Personally Identifiable Information \(PII\) \(p. 110\)](#) — Label text documents with personally identifiable information (PII).
- [Determine Sentiment \(p. 111\)](#)—Analyze documents and determine the dominant sentiment of the text.
- [Analyze Syntax \(p. 112\)](#)—Parse the words in your text and show the speech syntax for each word and enable you to understand the content of the document.
- [Topic Modeling \(p. 114\)](#)—Search the content of documents to determine common themes and topics.

Documents can be processed through these features in several ways: singly or in groups of up to 25 documents run synchronously with the result returned immediately, or in a larger batch run asynchronously, with the result saved to an S3 bucket. For more information, see [Document Processing Modes \(p. 117\)](#).

Detect Entities

Use the [DetectEntities \(p. 294\)](#), [BatchDetectEntities \(p. 223\)](#), and [StartEntitiesDetectionJob \(p. 359\)](#) operations to detect entities in a document. An *entity* is a textual reference to the unique name of a real-world object such as people, places, and commercial items, and to precise references to measures such as dates and quantities.

For example, in the text "John moved to 1313 Mockingbird Lane in 2012," "John" might be recognized as a `PERSON`, "1313 Mockingbird Lane" might be recognized as a `LOCATION`, and "2012" might be recognized as a `DATE`.

Each entity also has a score that indicates the level of confidence that Amazon Comprehend has that it correctly detected the entity type. You can filter out the entities with lower scores to reduce the risk of using incorrect detections.

The following table lists the entity types.

Type	Description
COMMERCIAL_ITEM	A branded product
DATE	A full date (for example, 11/25/2017), day (Tuesday), month (May), or time (8:30 a.m.)
EVENT	An event, such as a festival, concert, election, etc.
LOCATION	A specific location, such as a country, city, lake, building, etc.
ORGANIZATION	Large organizations, such as a government, company, religion, sports team, etc.
OTHER	Entities that don't fit into any of the other entity categories
PERSON	Individuals, groups of people, nicknames, fictional characters
QUANTITY	A quantified amount, such as currency, percentages, numbers, bytes, etc.
TITLE	An official name given to any creation or creative work, such as movies, books, songs, etc.

Detect entities operations can be performed using any of the primary languages supported by Amazon Comprehend. This includes only predefined (non-custom) entity detection. All documents must be in the same language.

You can use any of the following operations to detect entities in a document or set of documents.

- [DetectEntities \(p. 294\)](#)
- [BatchDetectEntities \(p. 223\)](#)
- [StartEntitiesDetectionJob \(p. 359\)](#)

The operations return a list of [Entity \(p. 452\)](#) objects, one for each entity in the document. The `BatchDetectEntities` operation returns a list of `Entity` objects, one list for each document in the batch. The `StartEntitiesDetectionJob` operation starts an asynchronous job that produces a file containing a list of `Entity` objects for each document in the job.

The following example is the response from the `DetectEntities` operation.

```
{
    "Entities": [
        {
            "Text": "today",
            "Score": 0.97,
            "Type": "DATE",
            "BeginOffset": 14,
            "EndOffset": 19
        },
        {
            "Text": "Seattle",
            "Score": 0.95,
            "Type": "LOCATION",
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ],
    "LanguageCode": "en"
}
```

Detect Events

With Amazon Comprehend, you can analyze your text documents to detect specific types of events and related details. You can apply natural language processing with asynchronous jobs for event detection across large collections of documents.

You can use the [StartEventsDetectionJob \(p. 364\)](#) operation to detect events.

Detect Events Results

When your event detection job completes, Amazon Comprehend writes the analysis results to the Amazon S3 output location that you specified.

For each detected event, the output provides details in the following format:

```
{  
    "Entities": [  
        {  
            "Mentions": [  
                {  
                    "BeginOffset": number,  
                    "EndOffset": number,  
                    "Score": number,  
                    "GroupScore": number,  
                    "Text": "string",  
                    "Type": "string"  
                }, ...  
            ]  
        }, ...  
    ],  
    "Events": [  
        {  
            "Type": "string",  
            "Arguments": [  
                {  
                    "EntityIndex": number,  
                    "Role": "string",  
                    "Score": number  
                }, ...  
            ],  
            "Triggers": [  
                {  
                    "BeginOffset": number,  
                    "EndOffset": number,  
                    "Score": number,  
                    "Text": "string",  
                    "GroupScore": number,  
                    "Type": "string"  
                }, ...  
            ]  
        }, ...  
    ]  
}
```

Entities

Amazon Comprehend returns a list of entities from the input text that are related to the detected event. An *entity* can be a real-world object, such as a person, place, or location; an entity can also be a concept, such as a measurement, date, or quantity. Each occurrence of an entity is identified by a *mention*, which is a textual reference to the entity in the input text. For each unique entity, all mentions are grouped into a list. This list provides details for each location in the input text where the entity occurs. Only entities

associated with a supported event type will be detected. You can use the [DetectEntities \(p. 294\)](#) and [StartEntitiesDetectionJob \(p. 359\)](#) operation to detect additional entities.

Each entity associated with a supported event type returns with the following related details:

- **Mentions:** Details for each occurrence of the same entity in the input text.
 - **BeginOffset:** A character offset in the input text that shows where the mention begins (the first character is at position 0).
 - **EndOffset:** A character offset in the input text that shows where the mention ends.
 - **Score:** The level of confidence that Amazon Comprehend has in the accuracy of the entity's type.
 - **GroupScore:** The level of confidence from Amazon Comprehend that the mention is correctly grouped with other mentions of the same entity.
 - **Text:** The text of the entity.
 - **Type:** The entity's type. For all supported entity types, see [Entity Types \(p. 97\)](#).

Events

Amazon Comprehend returns a list of events detected in the input text. Only supported event types will be detected.

Each event returns with the following related details:

- **Type:** The event's type. For all supported event types, see [Event Types \(p. 98\)](#).
- **Arguments:** A list of arguments that are related to the detected event. An *argument* consists of an entity that is related to the detected event. The argument's role describes the relationship, such as *who did what, where and when*.
 - **EntityIndex:** An index value that identifies an entity from the list of entities that Amazon Comprehend returned for this analysis.
 - **Role:** The argument type, which describes how the entity for this argument is related to the event. For all supported argument types, see [Argument Types \(p. 99\)](#).
 - **Score:** The level of confidence that Amazon Comprehend has in the accuracy of the role detection.
- **Triggers:** A list of triggers for the detected event. A *trigger* is a single word or phrase that indicates the occurrence of the event.
 - **BeginOffset:** A character offset in the input text that shows where the trigger begins (the first character is at position 0).
 - **EndOffset:** A character offset in the input text that shows where the trigger ends.
 - **Score:** The level of confidence that Amazon Comprehend has in the accuracy of the detection.
 - **Text:** The text of the trigger.
 - **GroupScore:** The level of confidence from Amazon Comprehend that the trigger is correctly grouped with other triggers for the same event.
 - **Type:** The type of event that this trigger indicates.

Supported Types for Entities, Events, and Arguments

Entity Types

Type	Description
DATE	Any reference to a date or time, whether specific or general.

Type	Description
FACILITY	Buildings, airports, highways, bridges, and other permanent man-made structures and real estate improvements.
LOCATION	Physical locations such as streets, cities, states, countries, bodies of water, or geographic coordinates.
MONETARY_VALUE	The value of something in US or other currency. The value can be specific or approximate.
ORGANIZATION	Companies and other groups of people defined by an established organizational structure.
PERSON	The names or nicknames of individuals or fictional characters.
PERSON_TITLE	Any title which describes a person, which is usually an employment category (such as CEO) or honorific (such as Mr.).
QUANTITY	A number or value and the unit of measurement.
STOCK_CODE	A stock ticker symbol, such as AMZN, an International Securities Identification Number (ISIN), Committee on Uniform Securities Identification Procedures (CUSIP), or Stock Exchange Daily Official List (SEDOL).

Event Types

Type	Description
BANKRUPTCY	A legal proceeding involving a person or company unable to repay outstanding debts.
EMPLOYMENT	Occurs when an employee is hired, fired, retired, or otherwise changes employment state.
CORPORATE_ACQUISTION	Occurs when a company obtains the possession of most or all of another company's shares or physical assets to gain control of that company.
INVESTMENT_GENERAL	Occurs when a person or company purchases an asset with the prospect of generating future income or appreciation.
CORPORATE_MERGER	Occurs when two or more companies unite to create a new legal entity.
IPO	An initial public offering (IPO) of shares of a private corporation to the public in a new stock issuance.
RIGHTS_ISSUE	A group of rights offered to existing shareholders to purchase additional stock shares, known as

Type	Description
	subscription warrants, in proportion to their existing holdings.
SECONDARY_OFFERING	An offer of securities by a shareholder of a company.
SHELF_OFFERING	A Securities and Exchange Commission (SEC) provision that allows an issuer to register a new issue of security and sell portions of the issue over a period of time without re-registering the security or incurring penalties. Also known as a shelf registration.
TENDER_OFFERING	An offer to purchase some or all of shareholders' shares in a company.
STOCK_SPLIT	Occurs when a company's board of directors increases the number of shares that are outstanding by issuing more shares to current shareholders. This event also applies to reverse stock splits.

Argument Types

Argument Types for BANKRUPTCY

Argument Type	Description
FILER	The person or company filing the bankruptcy.
DATE	The date or time of bankruptcy.
PLACE	Location or facility where (or nearest to where) the bankruptcy took place.

Argument Types for EMPLOYMENT

Type	Description
EMPLOYEE	The person employed by a company.
EMPLOYEE_TITLE	The title of the employee.
EMPLOYER	The person or company employing the employee.
END_DATE	The start date or time of the employment.
START_DATE	The end date or time of the employment.

Argument Types for CORPORATE_ACQUISITION, INVESTMENT_GENERAL

Type	Description
AMOUNT	The monetary value associated with the transaction.

Type	Description
INVESTEE	The person or company associated with the investment.
INVESTOR	The person or company investing in the asset.
DATE	The date or time of the acquisition or investment.
PLACE	Location where (or nearest to where) the acquisition or investment took place.

Argument Types for CORPORATE_MERGER

Type	Description
DATE	The date or time of the merger.
NEW_COMPANY	The new legal entity resulting from the merger.
PARTICIPANT	The company involved in the merger.

Argument Types for IPO, RIGHTS_ISSUE, SECONDARY_OFFERING, SHELF_OFFERING, TENDER_OFFERING

Type	Description
EXPIRE_DATE	The expiration date or time of the offering.
INVESTOR	The person or company investing in the asset.
OFFEREES	The person or company receiving the offering.
OFFERING_AMOUNT	The monetary value associated with the offering.
OFFERING_DATE	The date or time of the offering.
OFFEROR	The person or company initiating the offering.
OFFEROR_TOTAL_VALUE	The total monetary value associated with the offering.
RECORD_DATE	The record date or time of the offering.
SELLING_AGENT	The person or company facilitating the sale of the offering.
SHARE_PRICE	The monetary value associated with the share price.
SHARE_QUANTITY	The number of shares associated with the offering.
UNDERWRITERS	The company associated with the underwriting of the offering.

Argument Types for STOCK_SPLIT

Type	Description
COMPANY	The company issuing shares of the stock split.
DATE	The date or time of the stock split.
SPLIT_RATIO	The ratio of the increased new number of shares outstanding to the current number of shares before the stock split.

Detect Key Phrases

You can use Amazon Comprehend operations to find key phrases in your document.

A *key phrase* is a string containing a noun phrase that describes a particular thing. It generally consists of a noun and the modifiers that distinguish it. For example, "day" is a noun; "a beautiful day" is a noun phrase that includes an article ("a") and an adjective ("beautiful"). Each key phrase includes a score that indicates the level of confidence that Amazon Comprehend has that the string is a noun phrase. You can use the score to determine if the detection has high enough confidence for your application.

Detect key phrases operations can be performed using any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

You can use any of the following operations to detect key phrases in a document or set of documents.

- [DetectKeyPhrases \(p. 298\)](#)
- [BatchDetectKeyPhrases \(p. 226\)](#)
- [StartKeyPhrasesDetectionJob \(p. 368\)](#)

The operations return a list of [KeyPhrase \(p. 477\)](#) objects, one for each key phrase in the document. The `BatchDetectKeyPhrases` operation returns a list of `KeyPhrase` objects, one for each document in the batch. The `StartKeyPhrasesDetectionJob` operation starts an asynchronous job that produces a file containing a list of `KeyPhrase` objects for each document in the job.

The following example is the response from the `DetectKeyPhrases` operation.

```
{
    "LanguageCode": "en",
    "KeyPhrases": [
        {
            "Text": "today",
            "Score": 0.89,
            "BeginOffset": 14,
            "EndOffset": 19
        },
        {
            "Text": "Seattle",
            "Score": 0.91,
            "BeginOffset": 23,
            "EndOffset": 30
        }
    ]
}
```

Detect the Dominant Language

You can use Amazon Comprehend to examine text to determine the dominant language. Amazon Comprehend identifies the language using identifiers from RFC 5646 — if there is a 2-letter ISO 639-1 identifier, with a regional subtag if necessary, it uses that. Otherwise, it uses the ISO 639-2 3-letter code. For more information about RFC 5646, see [Tags for Identifying Languages](#) on the *IETF Tools* web site.

The response includes a score that indicates the confidence level that Amazon Comprehend has that a particular language is the dominant language in the document. Each score is independent of the other scores — it does not indicate that a language makes up a particular percentage of a document.

If a long document, like a book, is written in multiple languages, you can break the long document into smaller pieces and run the `DetectDominantLanguage` operation on the individual pieces. You can then aggregate the results to determine the percentage of each language in the longer document.

Amazon Comprehend can detect the following languages.

Code	Language
af	Afrikaans
am	Amharic
ar	Arabic
as	Assamese
az	Azerbaijani
ba	Bashkir
be	Belarusian
bn	Bengali
bs	Bosnian
bg	Bulgarian
ca	Catalan
ceb	Cebuano
cs	Czech
cv	Chuvash
cy	Welsh
da	Danish
de	German
el	Greek
en	English
eo	Esperanto
et	Estonian

Code	Language
eu	Basque
fa	Persian
fi	Finnish
fr	French
gd	Scottish Gaelic
ga	Irish
gl	Galician
gu	Gujarati
ht	Haitian
he	Hebrew
ha	Hausa
hi	Hindi
hr	Croatian
hu	Hungarian
hy	Armenian
ilo	Iloko
id	Indonesian
is	Icelandic
it	Italian
JV	Javanese
ja	Japanese
kn	Kannada
ka	Georgian
kk	Kazakh
km	Central Khmer
ky	Kirghiz
ko	Korean
ku	Kurdish
lo	Lao
la	Latin
lv	Latvian

Code	Language
lt	Lithuanian
lb	Luxembourgish
ml	Malayalam
mt	Maltese
mr	Marathi
mk	Macedonian
mg	Malagasy
mn	Mongolian
ms	Malay
my	Burmese
ne	Nepali
new	Newari
nl	Dutch
no	Norwegian
or	Oriya
om	Oromo
pa	Punjabi
pl	Polish
pt	Portuguese
ps	Pushto
qu	Quechua
ro	Romanian
ru	Russian
sa	Sanskrit
si	Sinhala
sk	Slovak
sl	Slovenian
sd	Sindhi
so	Somali
es	Spanish
sq	Albanian

Code	Language
sr	Serbian
su	Sundanese
sw	Swahili
sv	Swedish
ta	Tamil
tt	Tatar
te	Telugu
tg	Tajik
tl	Tagalog
th	Thai
tk	Turkmen
tr	Turkish
ug	Uighur
uk	Ukrainian
ur	Urdu
uz	Uzbek
vi	Vietnamese
yi	Yiddish
yo	Yoruba
zh	Chinese (Simplified)
zh-TW	Chinese (Traditional)

You can use any of the following operations to detect the dominant language in a document or set of documents.

- [DetectDominantLanguage \(p. 291\)](#)
- [BatchDetectDominantLanguage \(p. 220\)](#)
- [StartDominantLanguageDetectionJob \(p. 354\)](#)

The `DetectDominantLanguage` operation returns a [DominantLanguage \(p. 439\)](#) object. The `BatchDetectDominantLanguage` operation returns a list of `DominantLanguage` objects, one for each document in the batch. The `StartDominantLanguageDetectionJob` operation starts an asynchronous job that produces a file containing a list of `DominantLanguage` objects, one for each document in the job.

The following example is the response from the `DetectDominantLanguage` operation.

```
{
```

```
"Languages": [  
    {  
        "LanguageCode": "en",  
        "Score": 0.9793661236763  
    }  
]
```

Detect Personally Identifiable Information (PII)

You can use Amazon Comprehend to detect entities in your text that contain personally identifiable information (PII), or *PII entities*. A PII entity is a textual reference to personal data that could be used to identify an individual, such as an address, bank account number, or phone number.

For example, you can detect the PII entities in the following text by submitting it to Amazon Comprehend:

Hello Paulo Santos. The latest statement for your credit card account 1111-0000-1111-0000 was mailed to 123 Any Street, Seattle, WA 98109.

When Amazon Comprehend completes its analysis, it returns output that either locates or redacts the PII entities in the text.

For example, if you choose to locate the PII entities, the output includes the character offsets for each one, along with the entity type and other details. In this case, the output states that "Paul Santos" has the type NAME, "1111-0000-1111-0000" has the type CREDIT_DEBIT_NUMBER, and "123 Any Street, Seattle, WA 98109" has the type ADDRESS.

Alternatively, if you choose to redact the PII entities, Amazon Comprehend returns a copy of the input text in which each PII entity is redacted:

*Hello *****. The latest statement for your credit card account ***** was mailed to *** *** *****. *****.*

You can detect PII entities with both real-time synchronous operations and batch asynchronous jobs. However, you must use an asynchronous job if you want to produce output with redacted PII entities.

You can use the following operations to detect PII entities in a document or set of documents:

- [DetectPiiEntities \(p. 301\)](#)
- [StartPiiEntitiesDetectionJob \(p. 373\)](#)

You can use the following operation to label PII in a document or set of documents:

- [ContainsPiiEntities \(p. 238\)](#)

PII Entity Types

Amazon Comprehend can detect the following types of PII entities:

PII entity type	Description
ADDRESS	A physical address, such as "100 Main Street, Anytown, USA" or "Suite #12, Building 123". An address can include a street, building,

PII entity type	Description
	location, city, state, country, county, zip, precinct, neighborhood, and more.
AGE	An individual's age, including the quantity and unit of time. For example, in the phrase "I am 40 years old," Amazon Comprehend recognizes "40 years" as an age.
AWS_ACCESS_KEY	A unique identifier that's associated with a secret access key; the access key ID and secret access key are used together to sign programmatic AWS requests cryptographically.
AWS_SECRET_KEY	A unique identifier that's associated with an access key; the access key ID and secret access key are used together to sign programmatic AWS requests cryptographically.
BANK_ACCOUNT_NUMBER	A US bank account number. These are typically between 10 - 12 digits long, but Amazon Comprehend also recognizes bank account numbers when only the last 4 digits are present.
BANK_ROUTING	A US bank account routing number. These are typically 9 digits long, but Amazon Comprehend also recognizes routing numbers when only the last 4 digits are present.
CREDIT_DEBIT_CVV	A 3-digit card verification code (CVV) that is present on VISA, MasterCard, and Discover credit and debit cards. In American Express credit or debit cards, it is a 4-digit numeric code.
CREDIT_DEBIT_EXPIRY	The expiration date for a credit or debit card. This number is usually 4 digits long and formatted as month/year or MM/YY. For example, Amazon Comprehend can recognize expiration dates such as 01/21, 01/2021, and Jan 2021.
CREDIT_DEBIT_NUMBER	The number for a credit or debit card. These numbers can vary from 13 to 16 digits in length, but Amazon Comprehend also recognizes credit or debit card numbers when only the last 4 digits are present.
DATE_TIME	A date can include a year, month, day, day of week, or time of day. For example, Amazon Comprehend recognizes "January 19, 2020" or "11 am" as dates. Amazon Comprehend will recognize partial dates, date ranges, and date intervals. It will also recognize decades, such as "the 1990s".
DRIVER_ID	The number assigned to a driver's license, which is an official document permitting an individual to operate one or more motorized vehicles on a public road. A driver's license number consists of alphanumeric characters.

PII entity type	Description
EMAIL	An email address, such as <code>marymajor@email.com</code> .
IP_ADDRESS	An IPv4 address, such as <code>198.51.100.0</code> .
MAC_ADDRESS	A media access control (MAC) address is a unique identifier assigned to a network interface controller (NIC).
NAME	An individual's name. This entity type does not include titles, such as Mr., Mrs., Miss, or Dr. Amazon Comprehend does not apply this entity type to names that are part of organizations or addresses. For example, Amazon Comprehend recognizes the "John Doe Organization" as an organization, and it recognizes "Jane Doe Street" as an address.
PASSPORT_NUMBER	A US passport number. Passport numbers range from 6 - 9 alphanumeric characters.
PASSWORD	An alphanumeric string that is used as a password, such as <code>"*very20special#pass*</code> .
PHONE	A phone number. This entity type also includes fax and pager numbers.
PIN	A 4-digit personal identification number (PIN) that allows someone to access their bank account information.
SSN	A Social Security Number (SSN) is a 9-digit number that is issued to US citizens, permanent residents, and temporary working residents. Amazon Comprehend also recognizes Social Security Numbers when only the last 4 digits are present.
URL	A web address, such as <code>www.example.com</code> .
USERNAME	A user name that identifies an account, such as a login name, screen name, nick name, or handle.

Locate PII Entities

To locate the PII entities in your text, you can quickly analyze a single document, or you can start an asynchronous batch job on a large collection of documents.

To locate the PII entities in a single document, submit a [DetectPiiEntities \(p. 301\)](#) request, and include the document text in your request body. Your input text can include up to 5,000 bytes of UTF-8 encoded characters. Amazon Comprehend returns a list that provides the character offsets and other details for each PII entity that it detects, as shown by the following example response:

```
{
    "Entities": [
        {
            "Score": 0.9999669790267944,
```

```
        "Type": "NAME",
        "BeginOffset": 6,
        "EndOffset": 18
    },
    {
        "Score": 0.8905550241470337,
        "Type": "CREDIT_DEBIT_NUMBER",
        "BeginOffset": 69,
        "EndOffset": 88
    },
    {
        "Score": 0.9999889731407166,
        "Type": "ADDRESS",
        "BeginOffset": 103,
        "EndOffset": 138
    }
]
```

In addition to the character offsets, Amazon Comprehend provides the following for each PII entity:

- A score that estimates the probability that the detected text span is the detected entity type.
- The PII entity type.

To run the same analysis on a large collection of documents, run an asynchronous batch job. To run the job, upload your documents to Amazon S3, and submit a [StartPiiEntitiesDetectionJob \(p. 373\)](#) request. In your request, include the following required parameters:

- **InputDataConfig** – Provide an [InputDataConfig \(p. 475\)](#) definition for your request, which includes the input properties for the job. For the `S3Uri` parameter, specify the Amazon S3 location of your input documents.
- **OutputDataConfig** – Provide an [OutputDataConfig \(p. 482\)](#) definition for your request, which includes the output properties for the job. For the `S3Uri` parameter, specify the Amazon S3 location where Amazon Comprehend writes the results of its analysis.
- **DataAccessRoleArn** – Provide the Amazon Resource Name (ARN) of an AWS Identity and Access Management role. This role must grant Amazon Comprehend read access to your input data and write access to your output location in Amazon S3. For more information, see [Role-Based Permissions Required for Asynchronous Operations \(p. 196\)](#).
- **Mode** – Set this parameter to `ONLY_OFFSETS`. With this setting, the output provides the character offsets that locate each PII entity in the input text. The output also includes confidence scores and PII entity types.
- **LanguageCode** – Set this parameter to `en`. Amazon Comprehend supports PII detection in only English text.

Redact PII Entities

To redact the PII entities in your text, you must run an asynchronous batch job. To run the job, upload your documents to Amazon S3, and submit a [StartPiiEntitiesDetectionJob \(p. 373\)](#) request. In your request, include the following required parameters:

- **InputDataConfig** – Provide an [InputDataConfig \(p. 475\)](#) definition for your request, which includes the input properties for the job. For the `S3Uri` parameter, specify the Amazon S3 location of your input documents.
- **OutputDataConfig** – Provide an [OutputDataConfig \(p. 482\)](#) definition for your request, which includes the output properties for the job. For the `S3Uri` parameter, specify the Amazon S3 location where Amazon Comprehend writes the results of its analysis.

- **DataAccessRoleArn** – Provide the Amazon Resource Name (ARN) of an AWS Identity and Access Management role. This role must grant Amazon Comprehend read access to your input data and write access to your output location in Amazon S3. For more information, see [Role-Based Permissions Required for Asynchronous Operations \(p. 196\)](#).
- **Mode** – Set this parameter to `ONLY_REDACTION`. With this setting, Amazon Comprehend writes a copy of your input documents to the output location in Amazon S3. In this copy, each PII entity is redacted.
- **RedactionConfig** – Provide a [RedactionConfig \(p. 491\)](#) definition for your request, which includes the configuration parameters for the redaction. Specify the types of PII to redact, and specify whether each PII entity is replaced with the name of its type or a character of your choice:
 - To replace each PII entity with its type, set the `MaskMode` parameter to `REPLACE_WITH_PII_ENTITY_TYPE`. For example, with this setting, the PII entity "Jane Doe" is replaced with "[NAME]."
 - To replace the characters in each PII entity with a character of your choice, set the `MaskMode` parameter to `MASK`, and set the `MaskCharacter` parameter to the replacement character. Provide only a single character. Valid characters are `!`, `#`, `$`, `%`, `&`, `*`, and `@`. For example, with this setting, the PII entity "Jane Doe" can be replaced with "**** ***".
- **LanguageCode** – Set this parameter to `en`. Amazon Comprehend supports PII detection in only English text.

Label Documents with Personally Identifiable Information (PII)

You can use Amazon Comprehend to check for the presence of personally identifiable information (PII) in your text document, which refers to entity types representing personal data that could be used to identify an individual, such as name, address, bank account number, or phone number. If PII is found in your text document, Amazon Comprehend returns the labels of identified PII entity types.

For example, you can label a document with PII by submitting the following input text to Amazon Comprehend:

Hello Paulo Santos. The latest statement for your credit card account 1111-0000-1111-0000 was mailed to 123 Any Street, Seattle, WA 98109.

When Amazon Comprehend completes its analysis, it returns output that labels the document with PII.

For example, the output includes labels that represent PII entity types along with a confidence score of the accuracy. In this case, the document text "Paul Santos", "1111-0000-1111-0000" and "123 Any Street, Seattle, WA 98109" are represented by the labels `NAME`, `CREDIT_DEBIT_NUMBER`, and `ADDRESS` respectively as PII entity types. For more information about supported entity types, see [PII Entity Types \(p. 106\)](#).

Label Documents with PII Entity Types

To label documents with PII, submit a [ContainsPiiEntities \(p. 238\)](#) request, and include the input text in your request body. Your input text can include up to 50,000 bytes of UTF-8 encoded characters. Amazon Comprehend returns a list of labels of identified PII entity types as shown by the following example response:

```
{  
    "Labels": [  
        {  
            "Name": "NAME",  
            "Score": 0.9149109721183777  
        },  
    ]  
}
```

```
{  
    "Name": "CREDIT_DEBIT_NUMBER",  
    "Score": 0.5698626637458801  
}  
{  
    "Name": "ADDRESS",  
    "Score": 0.9951046109199524  
}  
]  
}
```

Amazon Comprehend provides the following for each label:

- The label name of the PII entity type.
- A score that estimates the probability that the detected text is labeled as a PII entity type.

Determine Sentiment

Use Amazon Comprehend to determine the sentiment of a document. You can determine if the sentiment is positive, negative, neutral, or mixed. For example, you can use sentiment analysis to determine the sentiments of comments on a blog posting to determine if your readers liked the post.

Determine sentiment operations can be performed using any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

You can use any of the following operations to detect the sentiment of a document or a set of documents.

- [DetectSentiment \(p. 303\)](#)
- [BatchDetectSentiment \(p. 229\)](#)
- [StartSentimentDetectionJob \(p. 377\)](#)

The operations return the most likely sentiment for the text as well as the scores for each of the sentiments. The score represents the likelihood that the sentiment was correctly detected. For example, in the example below it is 95 percent likely that the text has a Positive sentiment. There is a less than 1 percent likelihood that the text has a Negative sentiment. You can use the `SentimentScore` to determine if the accuracy of the detection meets the needs of your application.

The `DetectSentiment` operation returns an object that contains the detected sentiment and a [SentimentScore \(p. 496\)](#) object. The `BatchDetectSentiment` operation returns a list of sentiments and `SentimentScore` objects, one for each document in the batch. The `StartSentimentDetectionJob` operation starts an asynchronous job that produces a file containing a list of sentiments and `SentimentScore` objects, one for each document in the job.

The following example is the response from the `DetectSentiment` operation.

```
{  
    "SentimentScore": {  
        "Mixed": 0.030585512690246105,  
        "Positive": 0.94992071056365967,  
        "Neutral": 0.0141543131828308,  
        "Negative": 0.00893945890665054  
    },  
    "Sentiment": "POSITIVE",  
    "LanguageCode": "en"  
}
```

Analyze Syntax

Use the [DetectSyntax \(p. 306\)](#) and [BatchDetectSyntax \(p. 232\)](#) operations to analyze your documents to parse the words from the document and return the part of speech, or syntactic function, for each word in the document. You can identify the nouns, verbs, adjectives and so on in your document. Use this information to gain a richer understanding of the content of your documents, and to understand the relationship of the words in the document.

For example, you can look for the nouns in a document and then look for the verbs related to those nouns. In a sentence like "My grandmother moved her couch" you can see the nouns, "grandmother" and "couch," and the verb, "moved." You can use this information to build applications for analyzing text for word combinations that you are interested in.

To start the analysis, Amazon Comprehend parses the source text to find the individual words in the text. After the text is parsed, each word is assigned the part of speech that it takes in the source text.

Amazon Comprehend can identify 17 parts of speech. The parts of speech recognized are:

Token	Part of speech
ADJ	Adjective Words that typically modify nouns.
ADP	Adposition The head of a prepositional or postpositional phrase.
ADV	Adverb Words that typically modify verbs. They may also modify adjectives and other adverbs.
AUX	Auxiliary Function words that accompanies the verb of a verb phrase.
CCONJ	Coordinating conjunction Words that links words or phrases without subordinating one to the other.
DET	Determiner Articles and other words that specify a particular noun phrase.
INTJ	Interjection Words used as an exclamation or part of an exclamation.
NOUN	Noun

Token	Part of speech
	Words that specify a person, place, thing, animal, or idea.
NUM	<p>Numerical</p> <p>Words, typically determiners, adjectives, or pronouns, that express a number.</p>
O	<p>Other</p> <p>Words that can't be assigned a part of speech category.</p>
PART	<p>Particle</p> <p>Function words associated with another word or phrase to impart meaning.</p>
PRON	<p>Pronoun</p> <p>Words that substitute for nouns or noun phrases.</p>
PROPN	<p>Proper noun</p> <p>A noun that is the name of a specific individual, place or object.</p>
PUNCT	<p>Punctuation</p> <p>Non-alphabetical characters that delimit text.</p>
SCONJ	<p>Subordinating conjunction</p> <p>A conjunction that links parts of sentences by make one of them part of the other.</p>
SYM	<p>Symbol</p> <p>Word-like entities such as the dollar sign (\$) or mathematical symbols.</p>
VERB	<p>Verb</p> <p>Words that signal events and actions.</p>

For more information about the parts of speech, see [Universal POS tags](#) at the *Universal Dependencies* website.

The operations return tokens that identify the word and the part of speech that the word represents in the text. Each token represents a word in the source text. It provides the location of the word in the

source, the part of speech that the word takes in the text, the confidence that Amazon Comprehend has that the part of speech was correctly identified, and the word that was parsed from the source text.

The following is the structure of the list of syntax tokens. One syntax token is generated for each word in the document.

```
{  
    "SyntaxTokens": [  
        {  
            "BeginOffset": number,  
            "EndOffset": number,  
            "PartOfSpeech": {  
                "Score": number,  
                "Tag": "string"  
            },  
            "Text": "string",  
            "TokenId": number  
        }  
    ]  
}
```

Each token provides the following information:

- **BeginOffset** and **EndOffset**—Provides the location of the word in the input text.
- **PartOfSpeech**—Provides two pieces of information, the **Tag** that identifies the part of speech and the **Score** that represents the confidence that Amazon Comprehend Syntax has that the part of speech was correctly identified.
- **Text**—Provides the word that was identified.
- **TokenId**—Provides an identifier for the token. The identifier is the position of the token in the list of tokens.

Topic Modeling

You can use Amazon Comprehend to examine the content of a collection of documents to determine common themes. For example, you can give Amazon Comprehend a collection of news articles, and it will determine the subjects, such as sports, politics, or entertainment. The text in the documents doesn't need to be annotated.

Amazon Comprehend uses a [Latent Dirichlet Allocation](#)-based learning model to determine the topics in a set of documents. It examines each document to determine the context and meaning of a word. The set of words that frequently belong to the same context across the entire document set make up a topic.

A word is associated to a topic in a document based on how prevalent that topic is in a document and how much affinity the topic has to the word. The same word can be associated with different topics in different documents based on the topic distribution in a particular document.

For example, the word "glucose" in an article that talks predominantly about sports can be assigned to the topic "sports," while the same word in an article about "medicine" will be assigned to the topic "medicine."

Each word associated with a topic is given a weight that indicates how much the word helps define the topic. The weight is an indication of how many times the word occurs in the topic compared to other words in the topic, across the entire document set.

For the most accurate results you should provide Amazon Comprehend with the largest possible corpus to work with. For best results:

- You should use at least 1,000 documents in each topic modeling job.
- Each document should be at least 3 sentences long.
- If a document consists of mostly numeric data, you should remove it from the corpus.

Topic modeling is an asynchronous process. You submit your list of documents to Amazon Comprehend from an Amazon S3 bucket using the [StartTopicsDetectionJob \(p. 382\)](#) operation. The response is sent to an Amazon S3 bucket. You can configure both the input and output buckets. Get a list of the topic modeling jobs that you have submitted using the [ListTopicsDetectionJobs \(p. 346\)](#) operation and view information about a job using the [DescribeTopicsDetectionJob \(p. 288\)](#) operation. Content delivered to Amazon S3 buckets might contain customer content. For more information about removing sensitive data, see [How Do I Empty an S3 Bucket?](#) or [How Do I Delete an S3 Bucket?](#).

Documents must be in UTF-8 formatted text files. You can submit your documents two ways. The following table shows the options.

Format	Description
One document per file	Each file contains one input document. This is best for collections of large documents.
One document per line	The input is a single file. Each line in the file is considered a document. This is best for short documents, such as social media postings. Each line must end with a line feed (LF, \n), a carriage return (CR, \r), or both (CRLF, \r\n). The Unicode line separator (u+2028) can't be used to end a line.

For more information, see the [InputDataConfig \(p. 475\)](#) data type.

After Amazon Comprehend processes your document collection, it returns a compressed archive containing two files, `topic-terms.csv` and `doc-topics.csv`. For more information about the output file, see [OutputDataConfig \(p. 482\)](#).

The first output file, `topic-terms.csv`, is a list of topics in the collection. For each topic, the list includes, by default, the top terms by topic according to their weight. For example, if you give Amazon Comprehend a collection of newspaper articles, it might return the following to describe the first two topics in the collection:

Topic	Term	Weight
000	team	0.118533
000	game	0.106072
000	player	0.031625
000	season	0.023633
000	play	0.021118
000	yard	0.024454
000	coach	0.016012
000	games	0.016191

Topic	Term	Weight
000	football	0.015049
000	quarterback	0.014239
001	cup	0.205236
001	food	0.040686
001	minutes	0.036062
001	add	0.029697
001	tablespoon	0.028789
001	oil	0.021254
001	pepper	0.022205
001	teaspoon	0.020040
001	wine	0.016588
001	sugar	0.015101

The weights represent a probability distribution over the words in a given topic. Since Amazon Comprehend returns only the top 10 words for each topic the weights won't sum to 1.0. In the rare cases where there are less than 10 words in a topic, the weights will sum to 1.0.

The words are sorted by their discriminative power by looking at their occurrence across all topics. Typically this is the same as their weight, but in some cases, such as the words "play" and "yard" in the table, this results in an order that is not the same as the weight.

You can specify the number of topics to return. For example, if you ask Amazon Comprehend to return 25 topics, it returns the 25 most prominent topics in the collection. Amazon Comprehend can detect up to 100 topics in a collection. Choose the number of topics based on your knowledge of the domain. It may take some experimentation to arrive at the correct number.

The second file, `doc-topics.csv`, lists the documents associated with a topic and the proportion of the document that is concerned with the topic. If you specified `ONE_DOC_PER_FILE` the document is identified by the file name. If you specified `ONE_DOC_PER_LINE` the document is identified by the file name and the 0-indexed line number within the file. For example, Amazon Comprehend might return the following for a collection of documents submitted with one document per file:

Document	Topic	Proportion
sample-doc1	000	0.999330137
sample-doc2	000	0.998532187
sample-doc3	000	0.998384574
...		
sample-docN	000	3.57E-04

Amazon Comprehend utilizes information from the *Lemmatization Lists Dataset by MBM*, which is made available [here](#) under the [Open Database License \(ODbL\) v1.0](#).

Document Processing Modes

Amazon Comprehend enables you to examine your documents to gain various insights about their content.

When evaluating your documents, you can use one of several methods to process them, depending on how many documents you have and how you want to view the results:

- [Single-Document Processing \(p. 117\)](#)—You call Amazon Comprehend with a single document and receive a synchronous response, delivered to your application right away.
- [Multiple Document Synchronous Processing \(p. 122\)](#)—You call Amazon Comprehend with a collection of up to 25 documents and receive a synchronous response.
- [Asynchronous Batch Processing \(p. 117\)](#)—You put a collection of documents into an Amazon S3 bucket and start an asynchronous operation to analyze the documents. The results of the analysis are returned in an S3 bucket.

Single-Document Processing

The single-document operations are synchronous operations that return the results of analyzing the document directly to your application. You should use the single-document operations when you are creating an interactive application that works on one document at a time.

You can use the following single-document operations:

- [DetectDominantLanguage \(p. 291\)](#)
- [DetectEntities \(p. 294\)](#)
- [DetectKeyPhrases \(p. 298\)](#)
- [DetectPiiEntities \(p. 301\)](#)
- [ContainsPiiEntities \(p. 238\)](#)
- [DetectSentiment \(p. 303\)](#)
- [DetectSyntax \(p. 306\)](#)

Asynchronous Batch Processing

To analyze large documents and large collections of documents, use one of the Amazon Comprehend asynchronous operations. There is an asynchronous version of each of the Amazon Comprehend operations and an additional set of operations for topic modeling.

To analyze a collection of documents, you typically perform the following steps:

1. Store the documents in an Amazon S3 bucket.
2. Start one or more jobs to analyze the documents.
3. Monitor the progress of an analysis job.

4. Retrieve the results of the analysis from an S3 bucket when the job is complete.

The following sections describe using the Amazon Comprehend API to run asynchronous operations.

Prerequisites

Documents must be in UTF-8-formatted text files. You can submit your documents in two formats. The format you use depends on the type of documents you want to analyze, as described in the following table.

Description	Format
Each file contains one input document. This is best for collections of large documents.	One document per file
The input is one or more files. Each line in a file is considered a document. This is best for short documents, such as social media postings.	One document per line
Each line must end with a line feed (LF, \n), a carriage return (CR, \r), or both (CRLF, \r\n). You can't use the UTF-8 line separator (u+2028) to end a line.	

When you start an analysis job, you specify the S3 location for your input data. The URI must be in the same AWS Region as the API endpoint that you are calling. The URI can point to a single file or it can be the prefix for a collection of data files. For more information, see the [InputDataConfig \(p. 475\)](#) data type.

You must grant Amazon Comprehend access to the Amazon S3 bucket that contains your document collection and output files. For more information, see [Role-Based Permissions Required for Asynchronous Operations \(p. 196\)](#).

Starting an Analysis Job

To submit an analysis job, use either the Amazon Comprehend console or the appropriate `Start*` operation:

- [StartDominantLanguageDetectionJob \(p. 354\)](#) — Start a job to detect the dominant language in each document in the collection. For more information about the dominant language in a document, see [Detect the Dominant Language \(p. 102\)](#).
- [StartEntitiesDetectionJob \(p. 359\)](#) — Start a job to detect entities in each document in the collection. For more information about entities, see [Detect Entities \(p. 94\)](#).
- [StartEventsDetectionJob \(p. 364\)](#) — Start a job to detect events in each document in the collection. For more information about entities, see [Detect Events \(p. 96\)](#).
- [StartKeyPhrasesDetectionJob \(p. 368\)](#) — Start a job to detect key phrases in each document in the collection. For more information about key phrases, see [Detect Key Phrases \(p. 101\)](#).
- [StartPiiEntitiesDetectionJob \(p. 373\)](#) — Start a job to detect personally identifiable information (PII) in each document in the collection. For more information about PII, see [Detect Personally Identifiable Information \(PII\) \(p. 101\)](#).
- [StartSentimentDetectionJob \(p. 377\)](#) — Start a job to detect the emotional sentiment in each document in the collection. For more information about sentiments, see [Determine Sentiment \(p. 111\)](#).

- [StartTopicsDetectionJob \(p. 382\)](#) — Start a job to detect the topics in a document collection. For more information about topic modeling, see [Topic Modeling \(p. 114\)](#).

Monitoring Analysis Jobs

The `Start*` operation returns an ID that you can use to monitor the job's progress.

To monitor progress using the API, you use one of two operations, depending on whether you want to monitor the progress of an individual job or multiple jobs.

To monitor the progress of an individual analysis job, use the `Describe*` operations. You provide the job ID returned by the `Start*` operation. The response from the `Describe*` operation contains the `JobStatus` field with the job's status.

To monitor the progress of multiple analysis jobs, use the `List*` operations. `List*` operations return a list of jobs that you submitted to Amazon Comprehend. The response includes a `JobStatus` field for each job that tells you the status of the job.

If the `status` field is set to `COMPLETED` or `FAILED`, job processing has completed.

To get the status of individual jobs, use the `Describe*` operation for the analysis that you are performing.

- [DescribeDominantLanguageDetectionJob \(p. 266\)](#)
- [DescribeEntitiesDetectionJob \(p. 271\)](#)
- [DescribeEventsDetectionJob \(p. 277\)](#)
- [DescribeKeyPhrasesDetectionJob \(p. 279\)](#)
- [DescribePiiEntitiesDetectionJob \(p. 282\)](#)
- [DescribeSentimentDetectionJob \(p. 285\)](#)
- [DescribeTopicsDetectionJob \(p. 288\)](#)

To get the status of a multiple jobs, use the `List*` operation for the analysis that you are performing.

- [ListDominantLanguageDetectionJobs \(p. 317\)](#)
- [ListEntitiesDetectionJobs \(p. 323\)](#)
- [ListEventsDetectionJobs \(p. 332\)](#)
- [ListKeyPhrasesDetectionJobs \(p. 335\)](#)
- [ListPiiEntitiesDetectionJobs \(p. 338\)](#)
- [ListSentimentDetectionJobs \(p. 341\)](#)
- [ListTopicsDetectionJobs \(p. 346\)](#)

To restrict the results to jobs that match certain criteria, use the `List*` operations' `Filter` parameter. You can filter on the job name, the job status, and the date and time that the job was submitted. For more information, see the `Filter` parameter for each of the `List*` operations in the [Actions \(p. 218\)](#) reference.

Getting Analysis Results

After an analysis job has finished, use a `Describe*` operation to get the location of the results. If the job status is `COMPLETED`, the response includes an `OutputDataConfig` field that contains a field with the Amazon S3 location of the output file. The file, `output.tar.gz`, is a compressed archive that contains the results of the analysis.

If the status of a job is **FAILED**, the response contains a **Message** field that describes the reason that the analysis job didn't complete successfully.

To get the status of individual jobs, use the appropriate **Describe*** operation:

- [DescribeDominantLanguageDetectionJob \(p. 266\)](#)
- [DescribeEntitiesDetectionJob \(p. 271\)](#)
- [DescribeEventsDetectionJob \(p. 277\)](#)
- [DescribeKeyPhrasesDetectionJob \(p. 279\)](#)
- [DescribeSentimentDetectionJob \(p. 285\)](#)
- [DescribeTopicsDetectionJob \(p. 288\)](#)

The results are returned in a single file, with one JSON structure for each document. Each response file also includes error messages for any job with the status field set to **FAILED**.

Each of the following sections shows examples of output for the two input formats.

Getting Dominant Language Detection Results

The following is an example of an output file from an analysis that detected the dominant language. The format of the input is one document per line. For more information, see the [DetectDominantLanguage \(p. 291\)](#) operation.

```
{"File": "0_doc", "Languages": [{"LanguageCode": "en", "Score": 0.9514502286911011}, {"LanguageCode": "de", "Score": 0.02374090999364853}, {"LanguageCode": "nl", "Score": 0.003208699868991971}, "Line": 0} {"File": "1_doc", "Languages": [{"LanguageCode": "en", "Score": 0.9822712540626526}, {"LanguageCode": "de", "Score": 0.002621392020955682}, {"LanguageCode": "es", "Score": 0.002386554144322872}], "Line": 1}
```

The following is an example of output from an analysis where the format of the input is one document per file:

```
{"File": "small_doc", "Languages": [{"LanguageCode": "en", "Score": 0.9728053212165833}, {"LanguageCode": "de", "Score": 0.007670710328966379}, {"LanguageCode": "es", "Score": 0.0028472368139773607}]} {"File": "huge_doc", "Languages": [{"LanguageCode": "en", "Score": 0.984955906867981}, {"LanguageCode": "de", "Score": 0.0026436643674969673}, {"LanguageCode": "fr", "Score": 0.0014206881169229746}]}
```

Getting Entity Detection Results

The following is an example of an output file from an analysis that detected entities in documents. The format of the input is one document per line. For more information, see the [DetectEntities \(p. 294\)](#) operation. The output contains two error messages, one for a document that is too long and one for a document that isn't in UTF-8 format.

```
{"File": "50_docs", "Line": 0, "Entities": [{"BeginOffset": 0, "EndOffset": 22, "Score": 0.9763959646224976, "Text": "Cluj-NapocaCluj-Napoca", "Type": "LOCATION"}]} {"File": "50_docs", "Line": 1, "Entities": [{"BeginOffset": 11, "EndOffset": 15, "Score": 0.9615424871444702, "Text": "Maat", "Type": "PERSON"}]} {"File": "50_docs", "Line": 2, "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage": "Document size exceeds maximum size limit 102400 bytes."} {"File": "50_docs", "Line": 3, "ErrorCode": "UNSUPPORTED_ENCODING", "ErrorMessage": "Document is not in UTF-8 format and all subsequent lines are ignored."}
```

The following is an example of output from an analysis where the format of the input is one document per file. The output contains two error messages, one for a document that is too long and one for a document that isn't in UTF-8 format.

```
{"File": "non_utf8.txt", "ErrorCode": "UNSUPPORTED_ENCODING", "ErrorMessage": "Document is not in UTF-8 format and all subsequent line are ignored."}  
{ "File": "small_doc", "Entities": [{"BeginOffset": 0, "EndOffset": 4, "Score": 0.645766019821167, "Text": "Maat", "Type": "PERSON"}]}  
{ "File": "huge_doc", "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage": "Document size exceeds size limit 102400 bytes."}
```

Getting Events Detection Results

The following is an example an output file from an analysis job that detected events in documents. The format of the input is one document per line.

```
{"Entities": [{"Mentions": [{"BeginOffset": 12, "EndOffset": 27, "GroupScore": 1.0, "Score": 0.916355, "Text": "over a year ago", "Type": "DATE"}]}, {"Mentions": [{"BeginOffset": 33, "EndOffset": 39, "GroupScore": 1.0, "Score": 0.996603, "Text": "Amazon", "Type": "ORGANIZATION"}]}, {"Mentions": [{"BeginOffset": 66, "EndOffset": 77, "GroupScore": 1.0, "Score": 0.999283, "Text": "Whole Foods", "Type": "ORGANIZATION"}]}, {"Events": [{"Arguments": [{"EntityIndex": 2, "Role": "INVESTEE", "Score": 0.999283}, {"EntityIndex": 0, "Role": "DATE", "Score": 0.916355}, {"EntityIndex": 1, "Role": "INVESTOR", "Score": 0.996603}], "Triggers": [{"BeginOffset": 373, "EndOffset": 380, "GroupScore": 0.999984, "Score": 0.999955, "Text": "acquire", "Type": "CORPORATE_ACQUISITION"}], "Type": "CORPORATE_ACQUISITION"}, {"Arguments": [{"EntityIndex": 2, "Role": "PARTICIPANT", "Score": 0.999283}], "Triggers": [{"BeginOffset": 115, "EndOffset": 123, "GroupScore": 1.0, "Score": 0.999967, "Text": "combined", "Type": "CORPORATE_MERGER"}], "Type": "CORPORATE_MERGER"}]}, {"File": "doc.txt", "Line": 0}]}
```

For more information about events output file structure and supported event types, see [Detect Events \(p. 96\)](#).

Getting Key Phrase Detection Results

The following is an example of an output file from an analysis that detected key phrases in a document. The format of the input is one document per line. For more information, see the [DetectKeyPhrases \(p. 298\)](#) operation.

```
{"File": "50_docs", "KeyPhrases": [{"BeginOffset": 0, "EndOffset": 22, "Score": 0.8948641419410706, "Text": "Cluj-NapocaCluj-Napoca"}, {"BeginOffset": 45, "EndOffset": 49, "Score": 0.9989854693412781, "Text": "Cluj"}]}, {"Line": 0}]
```

The following is an example of the output from an analysis where the format of the input is one document per file.

```
{"File": "1_doc", "KeyPhrases": [{"BeginOffset": 0, "EndOffset": 22, "Score": 0.8948641419410706, "Text": "Cluj-NapocaCluj-Napoca"}, {"BeginOffset": 45, "EndOffset": 49, "Score": 0.9989854693412781, "Text": "Cluj"}]}
```

Getting Personally Identifiable Information (PII) Detection Results

The following is an example an output file from an analysis job that detected PII entities in documents. The format of the input is one document per line.

```
{"Entities": [{"Type": "NAME", "BeginOffset": 40, "EndOffset": 69, "Score": 0.999995}, {"Type": "ADDRESS", "BeginOffset": 247, "EndOffset": 253, "Score": 0.998828}, {"Type": "BANK_ACCOUNT_NUMBER", "BeginOffset": 406, "EndOffset": 411, "Score": 0.693283}], "File": "doc.txt", "Line": 1}, {"Entities": [{"Type": "SSN", "BeginOffset": 1114, "EndOffset": 1124, "Score": 0.999999}, {"Type": "EMAIL", "BeginOffset": 3742, "EndOffset": 3775, "Score": 0.999993}, {"Type": "PIN", "BeginOffset": 4098, "EndOffset": 4102, "Score": 0.999995}], "File": "doc.txt", "Line": 1}
```

The following is an example of output from an analysis where the format of the input is one document per file.

```
{"Entities": [{"Type": "NAME", "BeginOffset": 40, "EndOffset": 69, "Score": 0.999995}, {"Type": "ADDRESS", "BeginOffset": 247, "EndOffset": 253, "Score": 0.998828}, {"Type": "BANK_ROUTING", "BeginOffset": 279, "EndOffset": 289, "Score": 0.999999}], "File": "doc.txt"}
```

Getting Sentiment Detection Results

The following is an example of an output file from an analysis that detected the sentiment expressed in a document. It includes an error message because one document is too long. The format of the input is one document per line. For more information, see the [DetectSentiment \(p. 303\)](#) operation.

```
{"File": "50_docs", "Line": 0, "Sentiment": "NEUTRAL", "SentimentScore": {"Mixed": 0.002734508365392685, "Negative": 0.008935936726629734, "Neutral": 0.9841893315315247, "Positive": 0.004140198230743408}}, {"File": "50_docs", "Line": 1, "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage": "Document size is exceeded maximum size limit 5120 bytes."}, {"File": "50_docs", "Line": 2, "Sentiment": "NEUTRAL", "SentimentScore": {"Mixed": 0.0023119584657251835, "Negative": 0.0029857370536774397, "Neutral": 0.9866572022438049, "Positive": 0.008045154623687267}}
```

The following is an example of the output from an analysis where the format of the input is one document per file.

```
{"File": "small_doc", "Sentiment": "NEUTRAL", "SentimentScore": {"Mixed": 0.0023450672160834074, "Negative": 0.0009663937962614, "Neutral": 0.9795311689376831, "Positive": 0.017157377675175667}}, {"File": "huge_doc", "ErrorCode": "DOCUMENT_SIZE_EXCEEDED", "ErrorMessage": "Document size is exceeds the limit of 5120 bytes."}
```

Getting Topic Modeling Results

Topic modeling analysis returns two files in the `output.tar.gz` file. For more information, see [Topic Modeling \(p. 114\)](#).

Multiple Document Synchronous Processing

When you have multiple documents that you want to process, you can use the `Batch*` operations to send more than one document to Amazon Comprehend at a time. You can send up to 25 documents in each request. Amazon Comprehend sends back a list of responses, one for each document in the request.

You can use the following operations to process multiple documents in a single request. Requests made with these operations are synchronous. Your application calls the operation and then waits for the response from the service.

- [BatchDetectDominantLanguage \(p. 220\)](#)

- [BatchDetectEntities \(p. 223\)](#)
- [BatchDetectKeyPhrases \(p. 226\)](#)
- [BatchDetectSentiment \(p. 229\)](#)
- [BatchDetectSyntax \(p. 232\)](#)

Using the `Batch*` operations is identical to calling the single document APIs for each of the documents in the request. Using these APIs can result in better performance for your applications.

The input to each of the APIs is a JSON structure containing the documents to process. For all operations except `BatchDetectDominantLanguage`, you must set the input language. You can set only one input language for each request. For example, the following is the input to the `BatchDetectEntities` operation. It contains two documents and is in English.

```
{
    "LanguageCode": "en",
    "TextList": [
        "I have been living in Seattle for almost 4 years",
        "It is raining today in Seattle"
    ]
}
```

The response from a `Batch*` operation contains two lists, the `ResultList` and the `ErrorList`. The `ResultList` contains one record for each document that was successfully processed. The result for each document in the request is identical to the result you would get if you ran a single document operation on the document. The results for each document are assigned an index based on the order of the documents in the input file. The response from the `BatchDetectEntities` operation is:

```
{
    "ResultList" : [
        {
            "Index": 0,
            "Entities": [
                {
                    "Text": "Seattle",
                    "Score": 0.95,
                    "Type": "LOCATION",
                    "BeginOffset": 22,
                    "EndOffset": 29
                },
                {
                    "Text": "almost 4 years",
                    "Score": 0.89,
                    "Type": "QUANTITY",
                    "BeginOffset": 34,
                    "EndOffset": 48
                }
            ]
        },
        {
            "Index": 1,
            "Entities": [
                {
                    "Text": "today",
                    "Score": 0.87,
                    "Type": "DATE",
                    "BeginOffset": 14,
                    "EndOffset": 19
                },
                {
                    "Text": "Seattle",
                    "Score": 0.95,
                    "Type": "LOCATION",
                    "BeginOffset": 22,
                    "EndOffset": 29
                }
            ]
        }
    ]
}
```

```
        "Score": 0.96,
        "Type": "LOCATION",
        "BeginOffset": 23,
        "EndOffset": 30
    }
]
],
"ErrorList": []
}
```

When an error occurs in the request the response contains an `ErrorList` that identifies the documents that contained an error. The document is identified by its index in the input list. For example, the following input to the `BatchDetectLanguage` operation contains a document that cannot be processed:

```
{
    "TextList": [
        "hello friend",
        "$$$$$$",
        "hola amigo"
    ]
}
```

The response from Amazon Comprehend includes an error list that identifies the document that contained an error:

```
{
    "ResultList": [
        {
            "Index": 0,
            "Languages": [
                {
                    "LanguageCode": "en",
                    "Score": 0.99
                }
            ]
        },
        {
            "Index": 2,
            "Languages": [
                {
                    "LanguageCode": "es",
                    "Score": 0.82
                }
            ]
        }
    ],
    "ErrorList": [
        {
            "Index": 1,
            "ErrorCode": "InternalServerError",
            "ErrorMessage": "Unexpected Server Error. Please try again."
        }
    ]
}
```

Comprehend Custom

Comprehend custom helps you meet your specific needs even if you don't have the skillset to build machine learning-based NLP solutions. Using automatic machine learning, or AutoML, Comprehend Custom builds customized NLP models on your behalf, using data you already have. Training and calling custom comprehend models are both async (batch) operations.

Amazon Comprehend uses a proprietary, state-of-the-art sequence tagging deep neural network model that powers the same Amazon Comprehend detect entities service to train your custom entity recognizer models. In addition, we understand that acquiring training data could be costly. To help customers build a highly accurate model with limited amount of data, Amazon Comprehend uses a technique called *transfer learning* to train your custom models based on an sophisticated general-purpose entities recognition model that was pre-trained with a large amount of data we collected from multiple domains. Offline experiments have shown that transfer learning significantly improves custom entity recognizer model accuracy, especially when the amount of training data is small.

Topics

- [Custom Classification \(p. 125\)](#)
- [Custom Entity Recognition \(p. 147\)](#)
- [Managing Endpoints with Amazon Comprehend \(p. 170\)](#)

Custom Classification

You can use Amazon Comprehend to build your own models for *custom classification*. You can also assign a document to a specific class or category, or to multiple ones.

Custom classification is a two-step process. First, you train a custom classifier to recognize the classes that are of interest to you. Then you send unlabeled documents to be classified.

To train the classifier, specify the options you want, and send Amazon Comprehend documents to be used as training material. Based on the options you indicated, Amazon Comprehend creates a custom ML model that it trains based on the documents you provided. This custom model (the classifier) examines each document you submit. It then returns either the specific class that best represents the content (if you're using multi-class mode) or the set of classes that apply to it (if you're using multi-label mode).

For example, you can categorize the content of support requests so that you can route the request to the proper support team. Or you can categorize emails received from customers to provide guidance on the requests that customers are making. You can combine Amazon Comprehend with Amazon Transcribe to convert speech to text and then to classify the requests coming from support phone calls.

You can have multiple custom classifiers in your account, each trained using different data. When you submit a classification job, you choose which classifier to use. Amazon Comprehend returns results based on that classifier, how it was trained, and whether it was trained using multi-class or multi-label mode. The multi-class mode can be used asynchronously for a large document or set of documents or synchronously (in real-time) for a single document. The multi-label mode can only be used asynchronously.

Multi-Class and Multi-Label Modes

You can classify your documents using two modes: multi-class or multi-label. You can only use one mode at a time and it must be set when training your classifier. Some of the basic concepts and necessary

formats are different for each. In the Amazon Comprehend console, you choose which mode to use when creating your training job.

Multi-Class mode

Out of a group of at least two possible classes, multi-class mode specifies a single class for each document. The individual classes are mutually exclusive. For example, a movie can be classed as a documentary or as science fiction, but not both at the same time.

After training the custom classifier, you can then analyze documents in either asynchronous or synchronous operations. You can analyze a large number of documents at once using the asynchronous operation, with the resulting analysis returned in a separate file. Using the synchronous operation, you can only analyze a single document, but you can get results in real time. These options are not available when using multi-label mode. For more information, see [Asynchronous Classification \(p. 126\)](#).

Multi-Label mode

Out of a group of at least two possible classes, multi-label mode identifies one or more classes for each document. Unlike multi-class mode, these classes are not mutually exclusive and each document can have more than one class assigned to it. For example, a movie can simply be an action movie, or it can be an action movie, a science fiction movie, and a comedy, all at the same time.

Asynchronous Classification

When using multi-class mode, custom classification can be used for both asynchronous operations.

For asynchronous analysis, you first train a custom classifier (or custom model) to recognize the categories that are of interest to you. To train the classifier, you send Amazon Comprehend a group of classified documents, along with the class to which each belongs. After Amazon Comprehend builds the classifier, you send documents to be classified. The custom classifier examines each document and returns the category that best represents the content of the document. The results are then saved to a file in your S3 bucket. The cost of asynchronous custom classification is based on the number of characters used.

You can have multiple custom classifiers in your account, each trained using different data. You can then choose the classifier to meet your needs.

Topics

- [Training a Custom Classifier \(p. 126\)](#)
- [Running an Asynchronous Classification Job \(p. 132\)](#)
- [Real-time Analysis with Custom Classification \(p. 135\)](#)
- [Tagging Custom Classifiers \(p. 137\)](#)
- [Custom Classifier Metrics \(p. 142\)](#)

Training a Custom Classifier

To train a custom classifier follow these steps.

1. Provide your training data with the custom classes that you want the classifier to recognize.
2. Put your training data in an Amazon Simple Storage Service (Amazon S3) bucket. The bucket must have AWS Identity and Access Management (IAM) Amazon Comprehend read permissions that allow Amazon Comprehend access. For more information, see [Role-Based Permissions Required for Asynchronous Operations \(p. 196\)](#). Create another S3 bucket with the same requirements for your output.
3. Submit a training job using the [CreateDocumentClassifier \(p. 240\)](#) operation.

Train your custom classifier model in either multi-class or multi-label mode. The concept of *class* is used for both modes. It's a custom category that applies to the document being analyzed. However, each mode uses class differently. Multi-class mode associates only a single class with each document. Multi-label mode associates more than one class with a document. The training data formats are different for each mode as well.

You can train a custom classifier by using any of the following languages that work with Amazon Comprehend: English, Spanish, German, Italian, French, or Portuguese. However, you can only train the classifier in one language. Classifiers do not support multiple languages.

After you ask Amazon Comprehend to create a custom classifier, you can monitor the progress of the request using the [DescribeDocumentClassifier \(p. 263\)](#) operation. Once the Status field is TRAINED you can then use the classifier to classify documents.

Topics

- [Creating Training Data \(p. 127\)](#)
- [Multi-Class Mode \(p. 127\)](#)
- [Multi-Label Mode \(p. 129\)](#)
- [Testing the Training Data \(p. 131\)](#)

Creating Training Data

To train a custom classifier (custom model), identify the classes you want to use for classification. For example, **pricing**, **defect**, or **profanity**. Next, identify examples of documents for each of these classes. For each class, provide a minimum of 10 documents for training. For example, if you have 10 possible classes, you need a total of at least 100 classified documents to train the model. For more accurate training, we recommend at least 50 documents or more for each class.

Note

Even though you can use multiple classes in a classifier, no hierarchy is determined by them when you use the classifier on a document.

We recommend that you train the model with 50 or more training documents for each class. While a minimum of 10 training documents for each class is required, you get better accuracy with more documents. The total size of the training documents must be less than 5 GB.

Multi-Class Mode

In multi-class classification, each document can have one and only one class assigned to it. The individual classes are mutually exclusive. For example, a movie can be classed as a documentary or as science fiction, but not both at the same time.

After you train the custom classifier, you can analyze documents in either asynchronous or synchronous operations. You can analyze a large number of documents at once using the asynchronous operation. The resulting analysis is returned in a separate file. Using the synchronous operation, you can only analyze a single document, but you can get results in real time. These options are not available when you use multi-label mode. For more information, see [Asynchronous Classification \(p. 126\)](#).

To train a custom classifier, you must provide labeled training data. The labels in your training data should resemble the type of output that the trained model produces later when you provide unlabeled input. You can provide training data as a CSV file or as an augmented manifest file from SageMaker Ground Truth.

[CSV File](#)

To train a custom classifier, you can provide training data as a two-column CSV file. In it, labels are provided in the first column, and documents are provided in the second.

Classes can be any valid UTF-8 string. We suggest classes that are clear and don't overlap in meaning. They can have white space, and they can consist of multiple words connected by underscores or hyphens.

Training documents must end with \n or \r\n and be valid UTF-8 in a CSV file.

The data must be in two columns. Do not include headers for the individual columns. Including headers in your file may cause runtime errors. Each line of the file contains a single class and the text of a document that demonstrates that class.

```
CLASS,Text of document 1
CLASS,Text of document 2
CLASS,Text of document 3
```

For example, the following line belongs to a CSV file that trains a custom classifier to detect whether an email message is spam:

```
SPAM, "Paulo, your $1000 award is waiting for you! Claim it while you still can at http://example.com."
```

Augmented Manifest File

An augmented manifest file is a labeled dataset that is produced by SageMaker Ground Truth. Ground Truth is a data labeling service that helps you—or a workforce that you employ—build training datasets for machine learning models. Amazon Comprehend accepts augmented manifest files as training data for custom models. You can provide these files when you create a custom classifier by using the Amazon Comprehend console or the [CreateDocumentClassifier \(p. 240\)](#) API action.

For more information about Ground Truth and the output that it produces, see [Use Amazon SageMaker Ground Truth to Label Data](#) in the *Amazon SageMaker Developer Guide*.

Augmented manifest files are in JSON lines format. In these files, each line is a complete JSON object that contains a training document and its associated labels. The following example is an augmented manifest file that trains a custom classifier to determine whether an email message is spam:

```
{"source": "Document 1 text", "MultiClassJob": 0, "MultiClassJob-metadata": {"confidence": 0.62, "job-name": "labeling-job/multiclassjob", "class-name": "not_spam", "human-annotated": "yes", "creation-date": "2020-05-21T17:36:45.814354", "type": "groundtruth/text-classification"}}, {"source": "Document 2 text", "MultiClassJob": 1, "MultiClassJob-metadata": {"confidence": 0.81, "job-name": "labeling-job/multiclassjob", "class-name": "spam", "human-annotated": "yes", "creation-date": "2020-05-21T17:37:51.970530", "type": "groundtruth/text-classification"}}, {"source": "Document 3 text", "MultiClassJob": 1, "MultiClassJob-metadata": {"confidence": 0.81, "job-name": "labeling-job/multiclassjob", "class-name": "spam", "human-annotated": "yes", "creation-date": "2020-05-21T17:37:51.970566", "type": "groundtruth/text-classification"}}
```

Each line in this JSON lines file is a complete JSON object, where the attributes include the document text, a single class name, and other metadata from Ground Truth. The following example is a single JSON object in the augmented manifest file, but it's formatted for readability:

```
{ "source": "Paulo, your $1000 award is waiting for you! Claim it while you still can at http://example.com.", "MultiClassJob": 0, "MultiClassJob-metadata": { "confidence": 0.98, "job-name": "labeling-job/multiclassjob", "class-name": "spam",
```

```
        "human-annotated": "yes",
        "creation-date": "2020-05-21T17:36:45.814354",
        "type": "groundtruth/text-classification"
    }
}
```

In this example, the `source` attribute provides the text of the training document, and the `MultiClassJob` attribute assigns the index of a class from a classification list. The name of the `MultiClassJob` attribute is arbitrary, and you provide a name of your choice when you define the labeling job in Ground Truth.

In this example, the `MultiClassJob` attribute is the *label attribute name*, which is the attribute that provides the labels that a Ground Truth worker assigns to the training data. When you provide your training data to Amazon Comprehend, you must specify one or more label attribute names. The number of attribute names that you specify depends on whether your augmented manifest file is the output of a single labeling job or a chained labeling job.

If your file is the output of a single labeling job, specify the single label attribute name that was used when the job was created in Ground Truth.

If your file is the output of a chained labeling job, specify the label attribute name for one or more jobs in the chain. Each label attribute name provides the annotations from an individual job. You can specify up to 5 of these attributes for augmented manifest files that are produced by chained labeling jobs.

In an augmented manifest file, the label attribute name typically follows the `source` key. If the file is the output of a chained job, there will be multiple label attribute names. When you provide your training data to Amazon Comprehend, provide only those attributes that contain annotations that are relevant for your model. Do not specify the attributes that end with `-metadata`.

For more information about chained labeling jobs, and for examples of the output that they produce, see [Chaining Labeling Jobs](#) in the Amazon SageMaker Developer Guide.

Multi-Label Mode

In multi-label classification, individual classes represent different categories, but these categories are somehow related and are not mutually exclusive. As a result, each document has at least one class assigned to it, but can have more. For example, a movie can simply be an action movie, or it can be an action movie, a science fiction movie, and a comedy, all at the same time.

For training, multi-label mode supports up to 1 million examples containing up to 100 unique classes.

You can provide training data as a CSV file or as an augmented manifest file from Amazon SageMaker Ground Truth.

CSV File

To train a custom classifier, you can provide training data as a two-column CSV file. In it, labels are provided in the first column, and documents are provided in the second.

Do not include headers for the individual columns. Including headers in your CSV file may cause runtime errors. Each line of the file contains one or more classes and the text of the training document. More than one class can be indicated by using a delimiter (such as a |) between each class.

```
CLASS,Text of document 1
CLASS,Text of document 2
CLASS|CLASS|CLASS,Text of document 3
```

For example, the following line belongs to a CSV file that trains a custom classifier to detect genres in movie abstracts:

```
COMEDY|MYSTERY|SCIENCE_FICTION|TEEN, "A band of misfit teens become unlikely detectives when they discover troubling clues about their high school English teacher. Could the strange Mrs. Doe be an alien from outer space?"
```

The default delimiter between class names is a pipe (|). However, you can use a different character as a delimiter. The delimiter cannot be part of your class name. For example, if your classes are CLASS_1, CLASS_2, and CLASS_3, the underscore (_) is part of the class name. You cannot use then use an underscore as the delimiter for separating class names.

Augmented Manifest File

An augmented manifest file is a labeled dataset that is produced by SageMaker Ground Truth. Ground Truth is a data labeling service that helps you—or a workforce that you employ—build training datasets for machine learning models. Amazon Comprehend accepts augmented manifest files as training data for custom models. You can provide these files when you create a custom classifier by using the Amazon Comprehend console or the [CreateDocumentClassifier \(p. 240\)](#) API action.

For more information about Ground Truth and the output that it produces, see [Use Amazon SageMaker Ground Truth to Label Data](#) in the *Amazon SageMaker Developer Guide*.

Augmented manifest files are in JSON lines format. In these files, each line is a complete JSON object that contains a training document and its associated labels. The following example is an augmented manifest file that trains a custom classifier to detect genres in movie abstracts:

```
{"source": "Document 1 text", "MultiLabelJob": [0, 4], "MultiLabelJob-metadata": {"job-name": "labeling-job/multilabeljob", "class-map": {"0": "action", "4": "drama"}, "human-annotated": "yes", "creation-date": "2020-05-21T19:02:21.521882", "confidence-map": {"0": 0.66}, "type": "groundtruth/text-classification-multilabel"}, {"source": "Document 2 text", "MultiLabelJob": [3, 6], "MultiLabelJob-metadata": {"job-name": "labeling-job/multilabeljob", "class-map": {"3": "comedy", "6": "horror"}, "human-annotated": "yes", "creation-date": "2020-05-21T19:00:01.291202", "confidence-map": {"1": 0.61, "0": 0.61}, "type": "groundtruth/text-classification-multilabel"}, {"source": "Document 3 text", "MultiLabelJob": [1], "MultiLabelJob-metadata": {"job-name": "labeling-job/multilabeljob", "class-map": {"1": "action"}, "human-annotated": "yes", "creation-date": "2020-05-21T18:58:51.662050", "confidence-map": {"1": 0.68}, "type": "groundtruth/text-classification-multilabel"}}
```

Each line in this JSON lines file is a complete JSON object, where the attributes include the document text, one or more class names, and other metadata from Ground Truth. The following example is a single JSON object in the augmented manifest file, but it's formatted for readability:

```
{
    "source": "A band of misfit teens become unlikely detectives when they discover troubling clues about their high school English teacher. Could the strange Mrs. Doe be an alien from outer space?",
    "MultiLabelJob": [
        3,
        8,
        10,
        11
    ],
    "MultiLabelJob-metadata": {
        "job-name": "labeling-job/multilabeljob",
        "class-map": {
            "3": "comedy",
            "8": "mystery",
            "10": "science_fiction",
            "11": "teen"
        },
        "human-annotated": "yes",
        "creation-date": "2020-05-21T19:00:01.291202",
```

```

    "confidence-map": {
        "3": 0.95,
        "8": 0.77,
        "10": 0.83,
        "11": 0.92
    },
    "type": "groundtruth/text-classification-multilabel"
}
}

```

In this example, the `source` attribute provides the text of the training document, and the `MultiLabelJob` attribute assigns the indexes of several classes from a classification list. The name of the `MultiLabelJob` attribute is arbitrary, and you provide a name of your choice when you define the labeling job in Ground Truth.

In this example, the `MultiLabelJob` attribute is the *label attribute name*, which is the attribute that provides the labels that a Ground Truth worker assigns to the training data. When you provide your training data to Amazon Comprehend, you must specify one or more label attribute names. The number of attribute names that you specify depends on whether your augmented manifest file is the output of a single labeling job or a chained labeling job.

If your file is the output of a single labeling job, specify the single label attribute name that was used when the job was created in Ground Truth.

If your file is the output of a chained labeling job, specify the label attribute name for one or more jobs in the chain. Each label attribute name provides the annotations from an individual job. You can specify up to 5 of these attributes for augmented manifest files that are produced by chained labeling jobs.

In an augmented manifest file, the label attribute name typically follows the `source` key. If the file is the output of a chained job, there will be multiple label attribute names. When you provide your training data to Amazon Comprehend, provide only those attributes that contain annotations that are relevant for your model. Do not specify the attributes that end with "-metadata".

For more information about chained labeling jobs, and for examples of the output that they produce, see [Chaining Labeling Jobs](#) in the Amazon SageMaker Developer Guide.

Testing the Training Data

Once the model has been trained, Amazon Comprehend uses between 10 and 20 percent of the training documents to test the custom classifier model. Testing the model provides you with metrics that you can use to determine if the model is trained well enough for your purposes. These metrics are displayed in the **Classifier performance** section of the **Classifier details** page in the console. They are also returned in the `Metrics` fields returned by the [the section called " DescribeDocumentClassifier " \(p. 263\)](#) operation.

For example, in the sample of training data below, there are 5 labels, DOCUMENTARY, DOCUMENTARY, SCIENCE_FICTION, DOCUMENTARY, ROMANTIC_COMEDY. There are **3 unique classes**: DOCUMENTARY, SCIENCE_FICTION, ROMANTIC_COMEDY.

column 1	column 2
DOCUMENTARY	document text 1
DOCUMENTARY	document text 2
SCIENCE_FICTION	document text 3
DOCUMENTARY	document text 4
ROMANTIC_COMEDY	document text 5

For instance, if the data contained 1000 instances of the DOCUMENTARY class, 900 instances of the SCIENCE_FICTION, and a single instance of the ROMANTIC_COMEDY class, then the test set would approximately be 100 DOCUMENTARY and 90 SCIENCE_FICTION instances. The ROMANTIC_COMEDY class would not be included in the test set, as there is only a single example available. This is because it's highly unlikely you will see a document classified as ROMANTIC_COMEDY during prediction/inference in a setting like this.

Once you've finished training your model, the training metrics can provide you with information that you can use to decide if the model is trained sufficiently for your needs.

Running an Asynchronous Classification Job

After you've trained your model, your custom classifier is available for asynchronous use to categorize unlabeled documents.

Note

If trained in multi-class mode, your customer classifier can also be used for real-time insights into documents. However, real-time analysis can only be applied to a single document at a time. For more information, see [Real-time Analysis with Custom Classification \(p. 135\)](#).

For asynchronous analysis, all documents must be in UTF-8-formatted text files. Although you can only train your custom classification model using the one document per line format, you can submit your documents in that format or as one document per file. The format you use depends on the type of documents you want to analyze, as described below.

Description	Format
Each file contains one input document. This is best for collections of large documents, such as newspaper articles or scientific papers.	One document per file
The input is one or more files. Each line in a file is considered a document. This is best for short documents, such as text messages or social media posts.	One document per line
Each line must end with a line feed (LF, \n), a carriage return (CR, \r), or both (CRLF, \r\n). You can't use the UTF-8 line separator (u+2028) to end a line.	

One document per line

With the One document per line method, each document is placed on a separate line and no header is used. The label is not included on each line (since you don't yet know the label for the document). Each line of the file (the end of the individual document) must end with a line feed (LF, \n), a carriage return (CR, \r), or both (CRLF, \r\n).

The format of the input file can be seen as thus:

```
Text of document 1 \n
Text of document 2 \n
Text of document 3 \n
Text of document 4 \n
```

After preparing the documents file, you place that file in the S3 bucket that you're using for input data.

One Document per File

As with the previous method, the files used for this must be UTF-8 formatted text files. Each of these is placed into the S3 bucket being used for input data.

The input data bucket contains the files used to run the classification job. Each file represents one document.

When you start a classification job, you will specify this Amazon S3 location for your input data. The URI must be in the same AWS Region as the API endpoint that you are calling. The URI can point to a single file (as when using the "one document per line" method, or it can be the prefix for a collection of data files.

For example, if you use the URI `S3://bucketName/prefix`, if the prefix is a single file, Amazon Comprehend uses that file as input. If more than one file begins with the prefix, Amazon Comprehend uses all of them as input.

Grant Amazon Comprehend access to the S3 bucket that contains your document collection and output files. For more information, see [Role-Based Permissions Required for Asynchronous Operations \(p. 196\)](#).

The Classification Job

Use the [the section called " StartDocumentClassificationJob " \(p. 349\)](#) operation to start classifying unlabeled documents. You provide the S3 bucket that contains the documents to be classified, the S3 bucket where the output should be placed, and classifier to use.

Custom classification is asynchronous. Once you have started the job, use the [DescribeDocumentClassificationJob \(p. 260\)](#) operation to monitor its progress. When the `Status` field in the response shows `COMPLETED`, you can access the output in the location that you specified.

Classification Job Output

For asynchronous analysis, all documents must be in UTF-8-formatted text files. Although you can only train your custom classification model using the one document per line format, you can submit your documents in that format or as one document per file. The format you use depends on the type of documents you want to analyze, as described below.

The output from your classification job depends not only on the dataset you use. The output also depends on whether the classifier you use was trained for multi-class or multi-label mode. Inference (the classification job) is directly dependent on the classifier model training mode used.

Regardless of the mode used, the job output consists of a single file named `output.tar.gz`. It is a compressed archive file that contains a text file with the output.

Multi-Class Output

When you use a classifier trained in multi-class mode, your results are shown in terms of `classes`. Each of these `classes` is the class used to create the set of categories when training your classifier.

The following examples use the following mutually exclusive classes.

```
DOCUMENTARY
SCIENCE_FICTION
ROMANTIC_COMEDY
SERIOUS_DRAMA
OTHER
```

If your input data is formatted as one document per line, the output file contains one line for each line in the input. Each line has the file name, the zero-based line number of the input line, the class or classes

found in the document. It ends with the confidence that Amazon Comprehend has that the individual instance was correctly classified.

For example:

```
{"File": "file1.txt", "Line": "0", "Classes": [{"Name": "Documentary", "Score": 0.8642}, {"Name": "Other", "Score": 0.0381}, {"Name": "Serious_Drama", "Score": 0.0372}]}  
{"File": "file1.txt", "Line": "1", "Classes": [{"Name": "Science_Fiction", "Score": 0.5}, {"Name": "Science_Fiction", "Score": 0.0381}, {"Name": "Science_Fiction", "Score": 0.0372}]}  
{"File": "file2.txt", "Line": "2", "Classes": [{"Name": "Documentary", "Score": 0.1}, {"Name": "Documentary", "Score": 0.0381}, {"Name": "Documentary", "Score": 0.0372}]}  
{"File": "file2.txt", "Line": "3", "Classes": [{"Name": "Serious_Drama", "Score": 0.3141}, {"Name": "Other", "Score": 0.0381}, {"Name": "Other", "Score": 0.0372}]}
```

If your input data is formatted as one document per file, the output file contains one line for each document. Each line has the name of the file and the class or classes found in the document. It ends with the confidence that Amazon Comprehend has that the individual instance was correctly classified.

For example:

```
{"File": "file0.txt", "Classes": [{"Name": "Documentary", "Score": 0.8642}, {"Name": "Other", "Score": 0.0381}, {"Name": "Serious_Drama", "Score": 0.0372}]}  
{"File": "file1.txt", "Classes": [{"Name": "Science_Fiction", "Score": 0.5}, {"Name": "Science_Fiction", "Score": 0.0381}, {"Name": "Science_Fiction", "Score": 0.0372}]}  
{"File": "file2.txt", "Classes": [{"Name": "Documentary", "Score": 0.1}, {"Name": "Documentary", "Score": 0.0381}, {"Name": "Documentary", "Score": 0.0372}]}  
{"File": "file3.txt", "Classes": [{"Name": "Serious_Drama", "Score": 0.3141}, {"Name": "Other", "Score": 0.0381}, {"Name": "Other", "Score": 0.0372}]}
```

Multi-Label Output

When you use a classifier trained in multi-label mode, your results are shown in terms of labels. Each of these labels is the labels used to create the set of categories when training your classifier.

The following examples use these unique labels.

```
SCIENCE_FICTION
ACTION
DRAMA
COMEDY
ROMANCE
```

If your input data is formatted as one document per line, the output file contains one line for each line in the input. Each line has the file name, the zero-based line number of the input line, the class or classes found in the document. It ends with the confidence that Amazon Comprehend has that the individual instance was correctly classified.

For example:

```
{"File": "file1.txt", "Line": "0", "Labels": [{"Name": "Action", "Score": 0.8642}, {"Name": "Drama", "Score": 0.650}, {"Name": "Science Fiction", "Score": 0.0372}]}  
{"File": "file1.txt", "Line": "1", "Labels": [{"Name": "Comedy", "Score": 0.5}, {"Name": "Action", "Score": 0.0381}, {"Name": "Drama", "Score": 0.0372}]}  
{"File": "file1.txt", "Line": "2", "Labels": [{"Name": "Action", "Score": 0.9934}, {"Name": "Drama", "Score": 0.0381}, {"Name": "Action", "Score": 0.0372}]}  
{"File": "file1.txt", "Line": "3", "Labels": [{"Name": "Romance", "Score": 0.9845}, {"Name": "Comedy", "Score": 0.8756}, {"Name": "Drama", "Score": 0.7723}, {"Name": "Science_Fiction", "Score": 0.6157}]}]
```

If your input data is formatted as one document per file, the output file contains one line for each document. Each line has the name of the file and the class or classes found in the document. It ends with the confidence that Amazon Comprehend has that the individual instance was correctly classified.

For example:

```
{"File": "file0.txt", "Labels": [{"Name": "Action", "Score": 0.8642}, {"Name": "Drama", "Score": 0.650}, {"Name": "Science Fiction", "Score": 0.0372}]}  
{"File": "file1.txt", "Labels": [{"Name": "Comedy", "Score": 0.5}, {"Name": "Action", "Score": 0.0381}, {"Name": "Drama", "Score": 0.0372}]}  
{"File": "file2.txt", "Labels": [{"Name": "Action", "Score": 0.9934}, {"Name": "Drama", "Score": 0.0381}, {"Name": "Action", "Score": 0.0372}]}  
{"File": "file3.txt", "Labels": [{"Name": "Romance", "Score": 0.9845}, {"Name": "Comedy", "Score": 0.8756}, {"Name": "Drama", "Score": 0.7723}, {"Name": "Science_Fiction", "Score": 0.6157}]}
```

Note

For more information about the asynchronous analysis job format, see [Asynchronous Batch Processing \(p. 117\)](#)

Real-time Analysis with Custom Classification

Once a custom model has been trained, you can create an endpoint to gain real-time insights into your content. While the process of training the custom model and creating the endpoint are asynchronous, these insights are provided synchronously. This enables you to build real-time applications using a custom model.

An endpoint allows you to create an Amazon Comprehend-managed resource that makes your custom model available for real-time inference. You have full control over the endpoint and specify the throughput in increments of 100 characters per second when you create it. You can then adjust the throughput up or down to suit your needs. The cost of real-time analysis is based on the throughput of an endpoint and the duration of time it is active. You then delete the endpoint when it's no longer needed. All endpoint-related costs cease when the endpoint is deleted. For more information on endpoint cost, see [Amazon Comprehend Pricing](#).

Your endpoint uses inference units as a measure of the throughput of an endpoint. Each inference unit provides you with a throughput of 100 characters per second for up to 2 documents per second. You can scale the endpoint throughput either up or down by updating the endpoint. If the endpoint is no longer required, you can delete the endpoint.

Topics

- [Creating an Endpoint for Custom Classification \(p. 135\)](#)
- [Running Real-Time Custom Classification \(p. 136\)](#)

Creating an Endpoint for Custom Classification

Once you have a custom model created and trained, all you need is an endpoint to enable real-time analysis using that model.

To create an endpoint (console)

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Endpoints** and choose the **Create endpoint** button. A **Create endpoint** screen opens.
3. Give the endpoint a name. The name must be unique within the AWS Region and account.

4. Choose a custom model you want to attach the new endpoint to. From the dropdown, you can search by model name.

Note

You need to create a model before you can attach an endpoint to it. If you don't have a model yet, go to **Custom classification** or **Custom entity recognition** to create one.

5. (Optional) To add a tag to the endpoint, enter a key-value pair under **Tags** and choose **Add tag**. To remove this pair before creating the endpoint, choose **Remove tag**
6. Enter the number of inference units (IUs) to assign to the endpoint. Each unit represents a throughput of 100 characters per second for up to 2 documents per second.
7. (Optional) If you are creating a new endpoint, you have the option to use the IU estimator. It can be difficult to understand how many inference units you require depending on the throughput or the number of characters you want to analyze per second—especially at scale. This optional step can help you determine how many IUs to request.

Note

The range for available IUs is 1 to 10. The maximum characters you can analyze per second is 1000.

8. From the **Purchase summary**, review your estimated hourly, daily, and monthly endpoint cost.
9. Select the checkbox if you understand that you will be charged for the endpoint from the time it starts until it is deleted.

10. Choose **Create endpoint**

To create an endpoint (AWS CLI)

The following example demonstrates using the *CreateEndpoint* operation with the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend create-endpoint \
--desired-inference-units number of inference units \
--endpoint-name endpoint name \
--model-arn arn:aws:comprehend:region:account-id:model/example \
--tags Key=My1stTag,Value=Value1
```

Amazon Comprehend responds with the following:

```
{  
    "EndpointArn": "Arn"  
}
```

Running Real-Time Custom Classification

Once you've created an endpoint, you can run real-time analysis using your custom model. There are two different ways to run real-time analysis from the console, shown below, as well as the CLI method.

(Procedure A) To run real-time analysis using a custom model (console)

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Real-time analysis**.
3. Under **Input type**, choose **Custom** for **Analysis type**.
4. For **Select endpoint**, choose the endpoint that you want to use. This endpoint is linked to a specific custom model.
5. Enter the text you want to analyze.

6. Choose **Analyze**. The text analysis based on your custom model is displayed, along with a confidence assessment of the analysis.

(Procedure B) To run real-time analysis using a custom model (console)

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Customization** and then choose **Custom classification**.
3. From the **Classifiers** list, choose the name of the custom model for which you want to update the endpoint and follow the link. The custom model details page is displayed.
4. Navigate to the **Endpoints** list, choose the name of the endpoint you want to use for real-time analysis and follow the link. The endpoint details page is displayed.
5. Choose **Use in real-time analysis**.
6. For **Input text**, enter the text you want to analyze.
7. Choose **Analyze**. The text analysis based on your custom model is displayed, along with a confidence assessment of the analysis.

To run real-time analysis using a custom model (AWS CLI)

The following example demonstrates using the *ClassifyDocument* operation with the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend classify-document \
    --endpoint-arn arn:aws:comprehend:region:account-id:endpoint/endpoint name \
    --text 'From the Tuesday, April 16th, 1912 edition of The Guardian newspaper: The
maiden voyage of the White Star liner Titanic,
the largest ship ever launched ended in disaster. The Titanic started her trip from
Southampton for New York on Wednesday. Late
on Sunday night she struck an iceberg off the Grand Banks of Newfoundland. By wireless
telegraphy she sent out signals of distress,
and several liners were near enough to catch and respond to the call.'
```

Amazon Comprehend responds with the following:

```
{
  "Classes": [
    {
      "Name": "string",
      "Score": 0.9793661236763
    }
  ]
}
```

Tagging Custom Classifiers

A tag is a key-value pair which can be added to a Amazon Comprehend resource as metadata.

Tags can be used for organizing your resources as well as helping you to search and filter by tag. Using tag-based access control can allow for finer control than API-level control as well as more dynamic control than resource-based access control. IAM policies can be created that allow or disallow an operation based on tags provided in the request (request-tags) or tags on the resource that is being operated on (resource-tags). For more information on using IAM roles for this, [Controlling Access Using Tags](#) in the *IAM User Guide*

For instance, you could add tags such as 'Sales' or 'Legal' to different custom classifiers to break down documents into dynamic categories for your company.

Multiple tags can be added to an Amazon Comprehend resource, either when you create it, or later. Up to 50 tags can be associated with each resource.

Each tag key must be unique for a specific resource. However, the same key can be used for different resources. The tag value can be used to make the tag more specific, but is optional.

The following options are available when using tags with Amazon Comprehend custom classifiers:

- [Tagging a resource when you originally create it \(p. 138\)](#)
- [Tagging a resource after it's been created \(p. 138\)](#)
- [Modifying an existing tag \(p. 139\)](#)
- [Removing tags from a resource \(p. 140\)](#)
- [Viewing all tags associated with a resource \(p. 141\)](#)
- [Tagging restrictions \(p. 141\)](#)

Tagging a resource when you originally create it

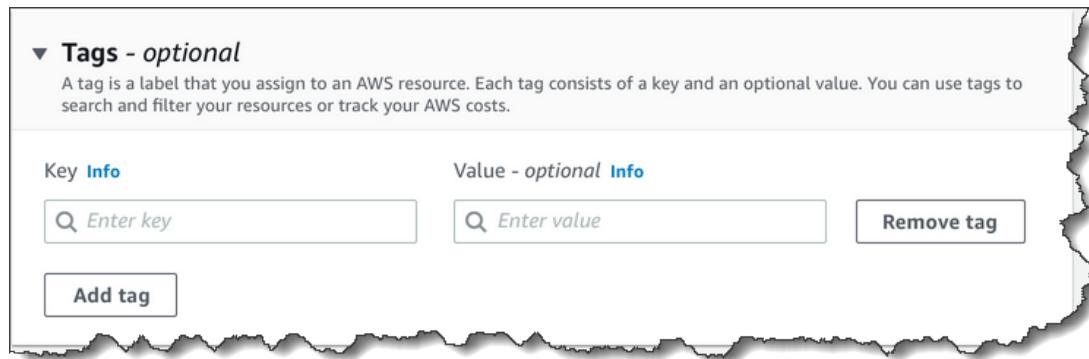
When you want to add one or more tags to a custom classifier when you originally create it, you can do this when training the classifier.

To tag a resource when creating it

1. In the optional **Tags** section, enter the key-value pair for your tag.

Note

Remember the key must be unique for the given resource. The value is optional.



2. Choose **Add tag** to associate the tag with your resource. You can do this multiple times to add more than one tag.

Once you've added a tag, you can remove it before creating the resource by choosing **Remove tag**.

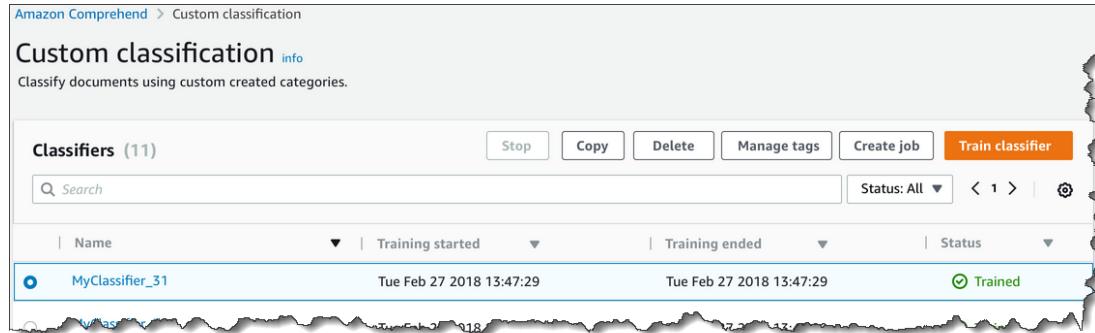
When using the APIs, you can do this using the `CreateDocumentClassifier` operation.

Tagging a resource after it's been created

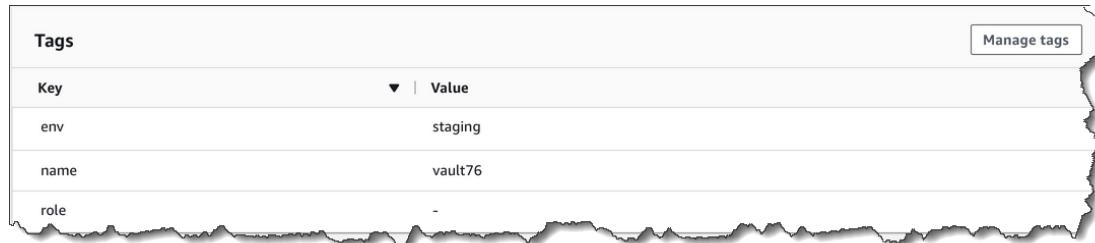
You can also easily add a tag to an existing custom classifier using the resource list or the detail page for a specific custom classifier.

To add a tag to an existing custom classifier

- On the Custom Classifier resource list, select the classifier to which you want to add the tag, and then choose **Manage tags**.



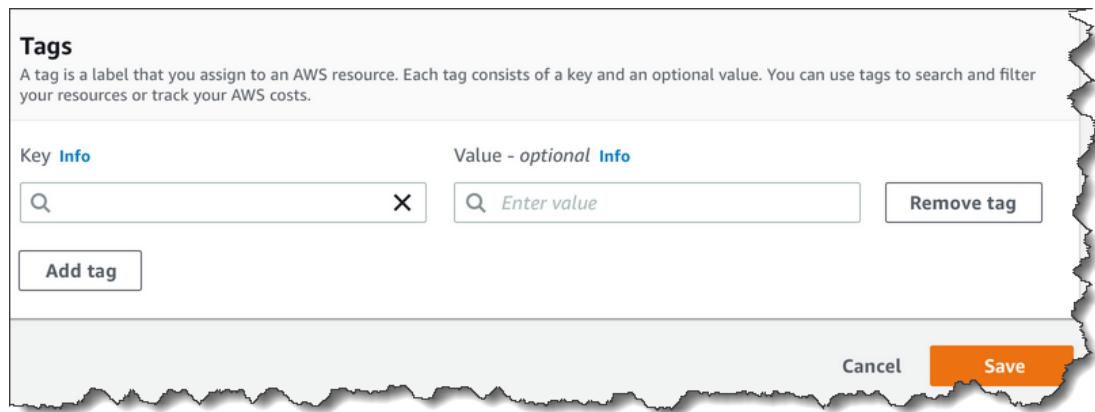
Alternatively, choose **Manage tags** in the **Tags** section of a specific classifier's details page.



- Under **Tags**, enter the key-value pair for your tag.

Note

Remember the key must be unique for the given resource. The value is optional.



- Choose **Add tag** to associate your tag with the resource. You can do this multiple times to add more than one tag.

Once you've added a tag, you can remove it before creating the resource by choosing **Remove tag**.

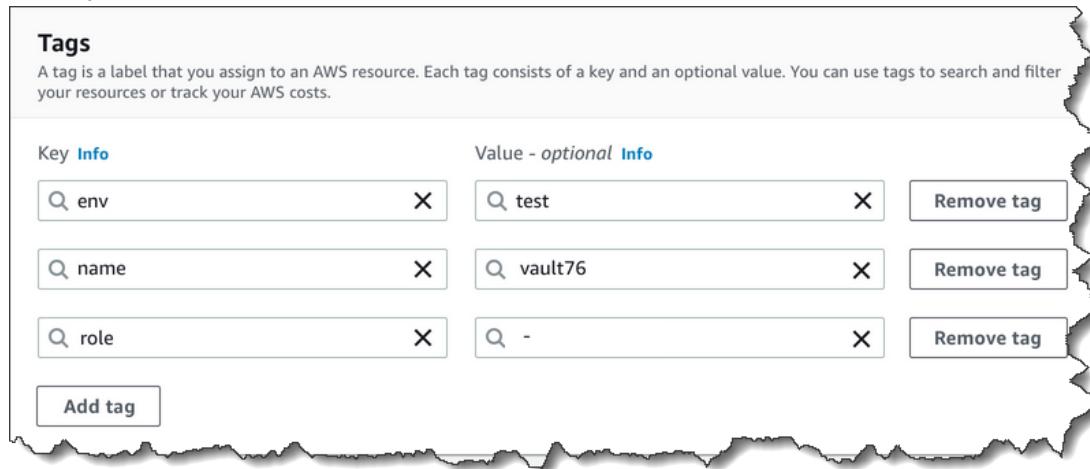
- Choose **Save**.

Modifying an existing tag

You can remove an existing using the resource list or the detail page for a specific resource.

To modify an existing tag

- Under **Tags**, choose the tag you want to change and choose the X for the key or value box. Enter the new key or value.



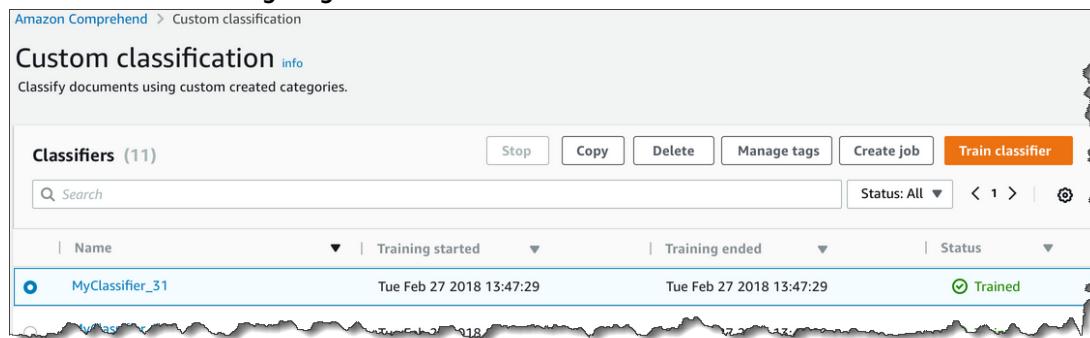
- Make any additional changes desired to the tags.
- Choose **Save**.

Removing tags from a resource

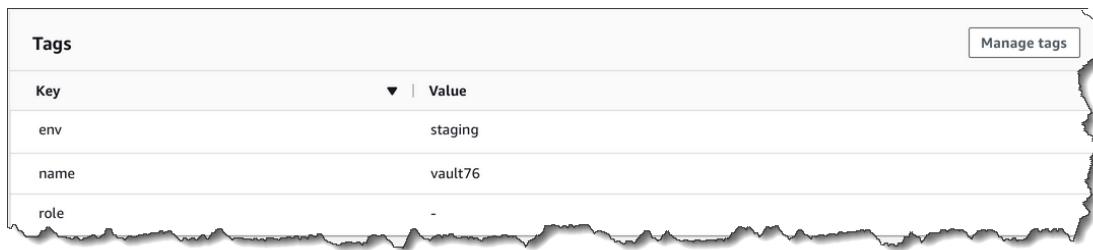
You can remove a tag from an existing custom classifier using the resource list or the detail page for a specific classifier.

To remove a tag from an existing custom classifier

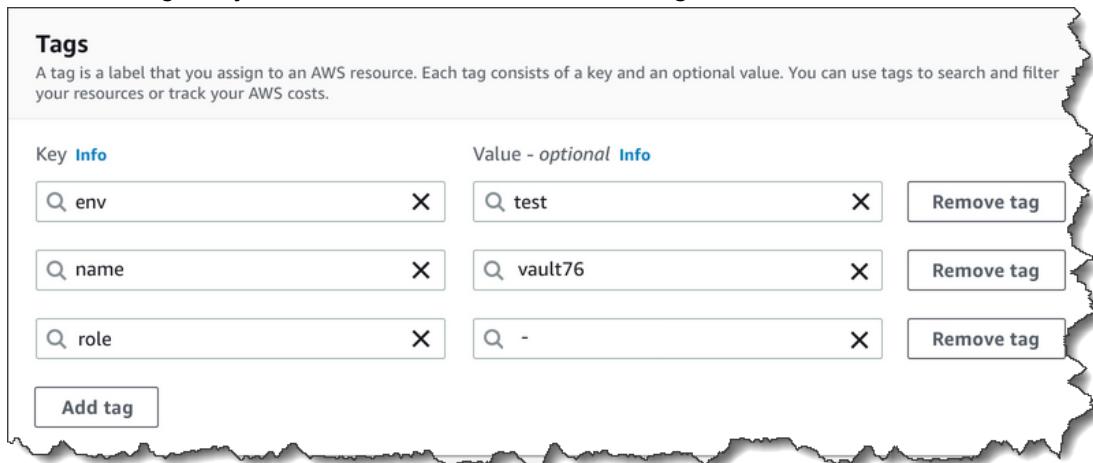
- On the Custom Classifier resource list, select the classifier from which you want to remove the tag, and then choose **Manage tags**.



Alternatively, choose **Manage tags** in the **Tags** section of a specific classifier's details page.



2. Next to the tag that you want to remove, choose **Remove tag**.



3. Choose **Save**.

Viewing all tags associated with a resource

To view all tags associated with a specific resource

- On the Custom Classifier resource list, select the classifier for which you want to view the tags, and then choose **Manage tags**.

Tagging restrictions

The following basic restrictions apply to Custom Classification tags:

- The maximum number of tags per classifier is 50.
- The maximum length of a tag key is 127 characters.
- The maximum length of a tag value is 255 characters.
- Tag keys and values are case sensitive.
- The `aws :` prefix is reserved for AWS use; you cannot add, edit, or delete tags whose key begins with `aws : .` Tags that begin with `aws : do` not count against your tags-per-resource limit.
- If you plan to use your tagging schema across multiple services and resources, remember that other services may have different restrictions for allowed characters. Refer to the documentation for that service.

Custom Classifier Metrics

Amazon Comprehend provides you with metrics to help you estimate how well a custom classifier should work for your job. They are based on training the classifier model, and so while they accurately represent the performance of the model during training, they are only an approximation of the API performance during classification.

Metrics are included any time metadata from a trained custom classifier is returned.

Note

Please refer to [Metrics: Precision, Recall, and FScore](#) for an understanding of the underlying Precision, Recall, and F1 score metrics. These metrics are defined at a class level. We have used **macro** averaging for combining these metrics together to come up with the test set P,R,F1, as discussed below.

Amazon Comprehend creates a [Confusion Matrix \(p. 145\)](#) as part of the custom classifier model training. This is placed in the output file specified in the [CreateDocumentClassifier \(p. 240\)](#) operation and can be used to assess how well the model works.

We also support the following metrics:

- **Accuracy**
- **Precision (Macro Precision)**
- **Recall (Macro Recall)**
- **F1 Score (Macro F1 Score)**
- **Hamming Loss**
- **Micro Precision**
- **Micro Recall**
- **Micro F1 Score**

These can be seen on the **Classifier Details** page in the console.

Classifier performance Info			
Accuracy	Precision	Recall	F1 score
0.34	0.3298	0.3304	0.32
Hamming loss	Micro precision	Micro recall	Micro F1 score
-	-	-	-

Accuracy

Accuracy indicates the percentage of labels from the test data that are predicted exactly right by the model. In other words, this is the fraction of the labels that were correctly recognized. It is computed by dividing the number of labels in the test documents that were correctly recognized by the total number of labels in the test documents.

For example

Actual Label	Predicted Label	Right/Wrong
1	1	Right
0	1	Wrong
2	3	Wrong
3	3	Right
2	2	Right
1	1	Right
3	3	Right

The accuracy consists of the number of "rights" divided by the number of overall test samples = 5/7 = 0.714, or 71.4%

Precision (Macro Precision)

Precision is a measure of the usefulness of the classifier results in the test data. It's defined as the number of documents correctly classified, divided by the total number of classifications for the class. High precision means that the classifier returned substantially more relevant results than irrelevant ones.

The Precision metric is also known as *Macro Precision*.

This is demonstrated in the following test set:

Label	Sample Size	Label Precision
Label_1	400	0.75
Label_2	300	0.80
Label_3	30000	0.90
Label_4	20	0.50
Label_5	10	0.40

The Precision (Macro Precision) metric for the model is therefore:

$$\text{Macro Precision} = (0.75 + 0.80 + 0.90 + 0.50 + 0.40)/5 = 0.67$$

Recall (Macro Recall)

This indicates the percentage of correct categories in your text that the model can predict. This metric comes from averaging the recall scores of all available labels. Recall is a measure of how complete the classifier results are for the test data.

High recall means that the classifier returned most of the relevant results.

The Recall metric is also known as *Macro Recall*.

This is demonstrated in the following test set:

Label	Sample Size	Label Recall
Label_1	400	0.70
Label_2	300	0.70
Label_3	30000	0.98
Label_4	20	0.80
Label_5	10	0.10

The Recall (Macro Recall) metric for the model is therefore:

$$\text{Macro Recall} = (0.70 + 0.70 + 0.98 + 0.80 + 0.10)/5 = 0.656$$

F1 Score (Macro F1 Score)

A combination of the Precision and Recall metrics. The F1 score is the harmonic mean of the Precision and Recall metrics. This score is based on the Precision and Recall created by the averaging method and is also known as the Macro F1 score. A measure of how accurate the classifier results are for the test data. It is derived from the Precision and Recall values. The F1Score is the harmonic average of the two scores. The highest score is 1, and the worst score is 0.

The F1 Score metric is also known as the *Macro F1 Score*.

This is demonstrated in the following test set:

Label	Sample Size	Label F1 Score
Label_1	400	0.724
Label_2	300	0.824
Label_3	30000	0.94
Label_4	20	0.62
Label_5	10	0.16

The F1 Score (Macro F1 Score) for the model is therefore as follows:

$$\text{Macro F1 Score} = (0.724 + 0.824 + 0.94 + 0.62 + 0.16)/5 = 0.6536$$

Hamming Loss

The fraction of labels that are incorrectly predicted. Also seen as the fraction of wrong labels compared to the total number of labels. Scores closer to zero are better.

Micro Precision

As Precision above, except that instead of averaging the precision scores of all available labels, this is based on the overall score of all precision scores added together.

Micro Recall

As Recall above, except that instead of averaging the recall scores of all labels, this is based on the overall score of all recall scores added together.

Micro F1 Score

As F1 Score above, but instead a combination of the Micro Precision and Micro Recall metrics.

Improving Your Custom Classifier's Performance

The metrics provide an insight into how your custom classifier will perform during a classification job. If the metrics are low, it's very likely that the classification model might not be effective for your use case. If this happens, you have several options to improve your classifier performance.

1. In your training data, provide more concrete data that can easily separate the categories. For example, provide documents that can best represent the label in terms of unique words/sentences.
2. Add more data for under-represented labels in your training data.
3. Try to reduce skew in the categories. If the largest label in your data is more than 10X the documents that are in the smallest label, try to increase the number of documents in the smallest and make sure to get the skew ratio down to at least 10:1 between highly represented and least represented classes. You can also try removing few documents from highly represented classes as well.

Confusion Matrix

A confusion matrix can give a very good indication on the classes for which adding more data would help model performance. A higher fraction of samples for a label shown along the diagonal of the matrix shows that the classifier is able to classify that label more accurately. If this number is lower (if the label class has a higher fraction of its samples in the non-diagonal portion of the matrix), you can try to add more samples. For example, if 40 percent of label A samples are classified as label D, adding more samples for both label A and label D will enhance the performance of the classifier. For more information, see [Confusion Matrix \(p. 145\)](#).

Confusion Matrix

When a custom classifier model is trained, Amazon Comprehend creates a confusion matrix that provides metrics on how well the model performed in training. This enables you to assess how well the classifier will perform when run. This matrix shows a matrix of labels as predicted by the model compared to actual labels and is created using 10 to 20 percent of the documents submitted to test the trained model.

The format of the confusion matrix produced varies, depending on if you train your classifier using multi-class or multi-label mode.

Confusion Matrix for Multi-Class Mode

Recall that in multi-class classification, the individual classes are mutually exclusive and each document is expected to have one and only one label assigned to it. For example, an animal can be a dog or a cat, but not both at the same time.

Consider the following example of a confusion matrix for a multi-class trained classifier:

A	B	X	Y	<-(predicted label)
A	1	2	0	4

```

B 0 3 0 1
X 0 0 1 0
Y 1 1 1 1
^
|
(actual label)

```

In this case, the model predicted the following:

- One "A" label was correctly identified, two "A" labels were incorrectly identified as actually "B" labels, and four "A" labels were incorrectly identified as "Y" labels.
- Three "B" labels were correctly identified, and one "B" label was incorrectly identified as a "Y" label.
- One "X" was correctly identified.
- One "Y" label was incorrectly identified as an "A" label, one was incorrectly identified as a "B" label, one was incorrectly identified as an "X" label, and one was correctly identified as a "Y" label.

In this matrix, the correctly identified labels are shown on the diagonal line (A:A, B:B, X:X, and Y:Y) so you can easily check the table for prediction errors because they will be represented as values outside this diagonal. In this case, you can see that the model correctly identifies "X" labels (although the sample is very small) and can correctly identify "B" labels 75% of the time. However, it incorrectly identifies "A" labels 86% of the time and correctly identifies "Y" labels at a rate no better than random chance.

The confusion matrix is presented in JSON format and for the above example is shown as the following:

```
{
  "type": "multi_class",
  "confusion_matrix": [
    [1, 2, 0, 4],
    [0, 3, 0, 1],
    [0, 0, 1, 0],
    [1, 1, 1, 1]],
  "labels": ["A", "B", "X", "Y"],
  "all_labels": ["A", "B", "X", "Y"]
}
```

Confusion Matrix for Multi-Label Mode

Recall that in multi-label classification, the individual labels represent different categories, but these categories are somehow related. For example, a movie can be just an action movie, or it can be an action movie, a science fiction movie, and a comedy, all at the same time.

Consider the following example of a confusion matrix for a multi-class trained classifier.

In this example, there are three possible labels: `Comedy`, `Action`, and `Drama`. Unlike the multi-class confusion matrix, the multi-label confusion matrix creates one 2x2 matrix for each label as shown below.

Comedy			Action			Drama			
No	Yes		No	Yes		No	Yes		<-(predicted label)
No	2	1	No	1	1	No	3	0	
Yes	0	2	Yes	2	1	Yes	1	1	
^			^			^			
----- (was this label actually used) -----									

In this case, the model returned the following for the Comedy label:

- Two instances where a Comedy label was not present and was correctly predicted to not be present. True negative (TN).
- Zero instances where a Comedy label was predicted to be present but was not. False positive (FP).
- One instance where a Comedy label was predicted to be absent but was present. False negative (FN).
- Two instances where a Comedy label was present were correctly predicted to be present. True positive (TP).

As with the multi-class confusion matrix above, the correctly identified labels here are shown on the diagonal line (Yes:Yes and No:No). You can easily check the table for prediction errors because they will be represented as values outside this diagonal. In this case, you can see that the model correctly identifies Comedy labels as present 40% of the time and can correctly identify their absence to the same degree (40%). However, it incorrectly identifies Comedy labels as being present 20% of the time. It did not incorrectly identify a Comedy label as absent when it was not.

The confusion matrix is presented in JSON format and for the above example is shown as the following:

```
{  
  "type": "multi_label",  
  "confusion_matrix": [  
    [[2, 1],  
     [0, 2]],  
    [[1, 1],  
     [2, 1]],  
    [[3, 0],  
     [1, 1]]  
  ],  
  "labels": ["Comedy", "Action", "Drama"]  
  "all_labels": ["Comedy", "Action", "Drama"]  
}
```

The CreateDocumentClassifier API

The confusion matrix is available when running the [CreateDocumentClassifier \(p. 240\)](#) API. When the operation is run, the confusion matrix is shown in the `confusion_matrix.json` file, located at `s3://user-defined-path/unique-value/output/output.tar.gz` where the user-defined-path is the S3Uri value of the `OutputDataConfig` parameter in the [CreateDocumentClassifier \(p. 240\)](#) operation.

Custom Entity Recognition

Custom entity recognition extends the capability of Amazon Comprehend by helping you identify your specific new entity types that are not of from the preset [generic entity types](#). This means that in addition to identifying predefined entity types such as LOCATION, DATE, PERSON, you can analyze documents and extract entities like product codes or business-specific entities that fit your particular needs.

Building an accurate in-house custom entity recognizer on your own can be a complex process, requiring preparation of large sets of manually annotated training documents and the selection of the right algorithms and parameters for model training. Amazon Comprehend helps you to sidestep some of these issues by providing automatic annotation and model development to create a custom entity recognition model.

Creating a custom entity recognition model is a more effective approach, compared to using string matching or regular expressions to extract entities from documents. For example, to extract ENGINEER names in a document, it would be difficult to enumerate all possible names. Additionally, without context, it would be challenging to distinguish between ENGINEER names and ANALYST names. A custom entity recognition model can learn the context where those names are likely to appear. Additionally, string matching will not detect entities that have typos or follow new naming conventions, while this is possible using a custom model.

To use Amazon Comprehend's custom entity recognition service, you have two options: The service automatically tests for the best algorithm and parameters while training the model to use, looking for the most accurate combination of these components.

You can train a model on up to 25 custom entities at once. Once your model is trained, you can search for those custom entities in each entities detection job. For more details, see the [Guidelines and Quotas page](#).

Topics

- [Training Custom Entity Recognizers \(p. 148\)](#)
- [Detecting Custom Entities with an Asynchronous Batch Job with Amazon Comprehend \(p. 160\)](#)
- [Detecting Custom Entities in Real Time with Amazon Comprehend \(p. 161\)](#)
- [Tagging Custom Entity Recognizers \(p. 163\)](#)
- [Custom Entity Recognizer Metrics \(p. 168\)](#)

Training Custom Entity Recognizers

Amazon Comprehend's custom entity recognition helps you to analyze your documents to find entities specific to your needs, rather than the entity types already available in the Detect Entities API. You can analyze plain text, PDF, and Word documents with no pre-processing or doc flattening required. You can identify almost any kind of entity, simply by providing a sufficient number of details to train your model effectively.

Note

For an analysis job on Word docs or PDFs, you need to have trained a custom entity recognizer from annotated PDFs. We don't support Word documents with macro.

Building a successful custom entity recognition model requires training your model. The training process usually requires extensive knowledge of machine learning (ML) and a complex process for model optimization. Amazon Comprehend automates this for you using a technique called *transfer learning* which builds on state of the art models in natural language processing (NLP) and generates a sophisticated general-purpose entity recognition model framework. When you prepare to build a successful custom entity recognition model it's important that you supply the model trainer with high quality data as input. Without good data the model won't learn how to correctly identify entities.

You can choose one of two ways to provide data to Amazon Comprehend in order to train a custom entity recognition model:

- [Annotations \(p. 150\)](#)—Provides the location of your entities in a large number of documents so Amazon Comprehend can train on both the entity and its context. To create a model which can be used to analyze PDF, Word and plain text documents, you must train your recognizer using PDF annotations.
- [Entity Lists \(Plain Text Only\) \(p. 159\)](#)—Lists the specific entities so Amazon Comprehend can train to identify your custom entities. Note: Entity lists can only be used for plain text documents.

In both cases, Amazon Comprehend will learn about the kind of documents and the context where the entities occur and build a recognizer that can generalize to new entities in documents at inference.

Annotations

By submitting annotation along with your documents, you can increase the accuracy of the model. With Annotations, you're not simply providing the location of the entity you're looking for, but you're also providing more accurate context to the custom entity you're seeking.

For instance, if you're searching for the name John Johnson, with the entity type JUDGE, providing your annotation might help the model to learn that the person you want to find is a judge. If it is able to use the context, then Amazon Comprehend won't find people named John Johnson who are attorneys or witnesses. Without providing annotations, Amazon Comprehend will create its own version of an annotation, but won't be as effective at including only judges. Providing your own annotations might help to achieve better results and to generate models that are capable of better leverage context when extracting custom entities.

Providing your own annotation takes more work, but can be significantly more refined. Not using your own annotation is quicker and less expensive (in terms of work), but the results are rougher and less accurate.

Entity Lists

With custom entity recognition, Amazon Comprehend helps you to train your model using an entity list. If you want to use an entity list, you provide two pieces of information: a list of the entity names with their corresponding custom entity types and a collection of unannotated documents which you expect your entities will appear. This is a more straightforward choice than the annotations option, but is probably going to result in a rougher, less specific result. This is because the annotations provide more context for Amazon Comprehend to use when training the model. Without that context, Amazon Comprehend will have a higher number of false positives when trying to identify the entities.

When you provide an Entity List, Amazon Comprehend uses an intelligent algorithm to detect occurrences of the entity in the documents to serve as the basis for training the custom entity recognizer model.

For instance, if you are searching for the name John Johnson, with the entity type JUDGE, using an entity list will enable you to identify instances of his name. However, without exact locations where John Johnson or JUDGE is annotated, Amazon Comprehend might view all instances of the name John Johnson, including other employees with the same name match for your custom entity.

Annotations or Entity List?

This comparison would make it seem as if annotations are always the best approach, but this isn't the case. Only the name John Johnson might be significant to your search and whether it's the exact individual isn't relevant. Or the result is useful, but not worth the time and cost of producing an annotation list. Or the metrics when using the entity list are good enough to provide you with the recognizer you need. There are many scenarios when it makes more business sense to avoid the higher expense and workload of creating the annotations necessary for the other option. In such instances, using an entity list instead can be the more effective choice.

We recommend using the annotations mode in the following cases:

- Annotations are required if you want to get inferences for PDF or Word documents. In this scenario, you can only annotate PDFs but your model can run inferences for PDF and Word documents.
- When the meaning of the entities could be ambiguous and context-dependent. For example, the term *Amazon* could either refer to the river in Brazil, or the online retailer Amazon.com. When you build a custom entity recognizer to identify business entities such as *Amazon*, you should use annotations instead of an entity list because this method is better able to use context to find entities.
- When you are comfortable setting up a process to acquire annotations, which can require some effort.

We recommend using entity list in the following cases:

- When you already have a list of entities or when it is relatively easy to compose a comprehensive list of entities. If you use an entity list, the list should be complete or at least covers the majority of valid entities that might appear in the documents you provide for training.
- For first-time users, it is generally recommended to use an entity list because this requires a smaller effort than constructing annotations. However, it is important to note that the trained model might not be as accurate as if you used annotations.

Data Options

- [Annotations \(p. 150\)](#)
- [PDF Annotations \(p. 153\)](#)
- [Annotation Best Practices \(p. 158\)](#)
- [Entity Lists \(Plain Text Only\) \(p. 159\)](#)

Annotations

To train a custom entity recognizer, you can provide annotated training data. Annotations label entities in context by associating your custom entity types with the locations where they occur in your training documents. To learn about best practices, please see [Annotation Best Practices \(p. 158\)](#).

You can provide training data as a CSV file or as an augmented manifest file from SageMaker Ground Truth.

CSV File (Plain Text Only)

Annotations for custom entity recognition needs to be in a comma-separated value (CSV) file, with the following columns:

- **File**—The name of the file containing the document. For example, if one of the document files is located at `s3://custom-ner-test-datasets-us-west-2/train-small-001/documents/documents.txt`, the value in the **File** column will be `documents.txt`. Please note that if the file name has an extension such as `.txt`, this must be included as part of the file name.
- **Line**—The line number containing the entity, starting with line 0.
- **Begin Offset**—The character offset in the input text (relative to the beginning of the line) that shows where the entity begins. The first character is at position 0.
- **End Offset**—The character offset in the input text that shows where the entity ends.
- **Type**—The customer-defined entity type. Entity types must be an uppercase, underscore-separated string. We recommend using descriptive entity types such as `MANAGER`, `SENIOR_MANAGER`, or `PRODUCT_CODE`. Up to 25 entity types can be trained per model.

For example:

The file `documents.txt` contains four lines:

```
Josef Brown is an engineer in the high tech industry.  
J Doe has been a engineer for 14 years.  
Emilio Johnson is a judge on the Washington Supreme Court.  
Our latest new employee, Smith, has been a manager in the industry for 4 years.
```

The CSV file with the list of annotations has the following lines:

```
File, Line, Begin Offset, End Offset, Type  
documents.txt, 0, 0, 11, ENGINEER
```

```
documents.txt, 1, 0, 5, ENGINEER
documents.txt, 3, 25, 30, MANAGER
```

Note

In the annotations file, the line number containing the entity starts starting with line 0. In this example, line 2 is not present in the CSV file because no entity was found in line 2 of documents.txt.

Creating your data files

It is important that your annotations be in a properly configured CSV file so your chance of having problems with your annotations file is minimal. To manually configure your CSV file, the following must be true:

- UTF-8 encoding must be explicitly specified, even if its used as a default in most cases.
- It must include in the first line the column names: File, Line, Begin Offset, End Offset, Type.

We highly recommended that CSV input files are generated programmatically to avoid potential issues.

The following example uses Python to generate a CSV for the annotations shown above:

```
import csv
with open("./annotations/annotations.csv", "w", encoding="utf-8") as csv_file:
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow(["File", "Line", "Begin Offset", "End Offset", "Type"])
    csv_writer.writerow(["documents.txt", 0, 0, 11, "ENGINEER"])
    csv_writer.writerow(["documents.txt", 1, 0, 5, "ENGINEER"])
    csv_writer.writerow(["documents.txt", 3, 25, 30, "MANAGER"])
```

Augmented Manifest File

An augmented manifest file is a labeled dataset that is produced by SageMaker Ground Truth. Ground Truth is a data labeling service that helps you—or a workforce that you employ—build training datasets for machine learning models. Amazon Comprehend accepts augmented manifest files as training data for custom models. You can provide these files when you create a custom entity recognizer by using the [Amazon Comprehend console](#) or the [CreateEntityRecognizer \(p. 249\)](#) API action.

You can use the Ground Truth built-in task type, Named Entity Recognition, to create a labeling job to have workers identify entities in text. To learn more, see [Named Entity Recognition](#) in the *Amazon SageMaker Developer Guide*. To learn more about Amazon SageMaker Ground Truth, see [Use Amazon SageMaker Ground Truth to Label Data](#).

Augmented manifest files are in JSON lines format. In these files, each line is a complete JSON object that contains a training document and its associated labels. The following example is an augmented manifest file that trains an entity recognizer to detect the professions of individuals who are mentioned in the text:

```
{"source":"Diego Ramirez is an engineer in the high tech
industry.", "NamedEntityRecognitionDemo": {"annotations": {"entities": [{"endOffset": 13, "startOffset": 0, "label": "ENGINEER"}], "labels": [{"label": "ENGINEER"}]}}, "NamedEntityRecognitionDemo-metadata": {"entities": [{"confidence": 0.92}], "job-name": "labeling-job/
namedentityrecognitiondemo", "type": "groundtruth/text-span", "creation-
date": "2020-05-14T21:45:27.175903", "human-annotated": "yes"}}
{"source": "J Doe is a judge on the Washington Supreme Court.", "NamedEntityRecognitionDemo": {"annotations": {"entities": [{"endOffset": 5, "startOffset": 0, "label": "JUDGE"}]}, "labels": [{"label": "JUDGE"}]}}, "NamedEntityRecognitionDemo-metadata": {"entities": [{"confidence": 0.72}], "job-name": "labeling-job/
```

```
namedentityrecognitiondemo", "type": "groundtruth/text-span", "creation-date": "2020-05-14T21:45:27.174910", "human-annotated": "yes"}}
{"source": "Our latest new employee, Mateo Jackson, has been a manager in
the industry for 4 years.", "NamedEntityRecognitionDemo": {"annotations":
{"entities": [{"endOffset": 38, "startOffset": 26, "label": "MANAGER"}]}, "labels":
[{"label": "MANAGER"}]}}, "NamedEntityRecognitionDemo-metadata": {
"entities": [{"confidence": 0.91}], "job-name": "labeling-job/
namedentityrecognitiondemo", "type": "groundtruth/text-span", "creation-
date": "2020-05-14T21:45:27.174035", "human-annotated": "yes"}}
```

Each line in this JSON lines file is a complete JSON object, where the attributes include the document text, the annotations, and other metadata from Ground Truth. The following example is a single JSON object in the augmented manifest file, but it's formatted for readability:

```
{
  "source": "Diego Ramirez is an engineer in the high tech industry.",
  "NamedEntityRecognitionDemo": {
    "annotations": {
      "entities": [
        {
          "endOffset": 13,
          "startOffset": 0,
          "label": "ENGINEER"
        }
      ],
      "labels": [
        {
          "label": "ENGINEER"
        }
      ]
    }
  },
  "NamedEntityRecognitionDemo-metadata": {
    "entities": [
      {
        "confidence": 0.92
      }
    ],
    "job-name": "labeling-job/namedentityrecognitiondemo",
    "type": "groundtruth/text-span",
    "creation-date": "2020-05-14T21:45:27.175903",
    "human-annotated": "yes"
  }
}
```

In this example, the `source` attribute provides the text of the training document, and the `NamedEntityRecognitionDemo` attribute provides the annotations for the entities in the text. The name of the `NamedEntityRecognitionDemo` attribute is arbitrary, and you provide a name of your choice when you define the labeling job in Ground Truth.

In this example, the `NamedEntityRecognitionDemo` attribute is the *label attribute name*, which is the attribute that provides the labels that a Ground Truth worker assigns to the training data. When you provide your training data to Amazon Comprehend, you must specify one or more label attribute names. The number of attribute names that you specify depends on whether your augmented manifest file is the output of a single labeling job or a chained labeling job.

If your file is the output of a single labeling job, specify the single label attribute name that was used when the job was created in Ground Truth.

If your file is the output of a chained labeling job, specify the label attribute name for one or more jobs in the chain. Each label attribute name provides the annotations from an individual job. You can specify up to 5 of these attributes for augmented manifest files that are produced by chained labeling jobs.

In an augmented manifest file, the label attribute name typically follows the source key. If the file is the output of a chained job, there will be multiple label attribute names. When you provide your training data to Amazon Comprehend, provide only those attributes that contain annotations that are relevant for your model. Do not specify the attributes that end with "-metadata".

For more information about chained labeling jobs, and for examples of the output that they produce, see [Chaining Labeling Jobs](#) in the Amazon SageMaker Developer Guide.

PDF Annotations

With Amazon Comprehend you can train your models and use your models to gain insight on text files, PDFs, and Word documents. Before you can annotate your training PDFs in SageMaker Ground Truth, there are a few prerequisites and tools required. The lifecycle of this process consists of three main parts - installing the prerequisites and *preparing* your environments, *building* out the tool, and then once everything is set up, *annotating*. During this process you'll be required to go to a few places —you'll be pulling our resources from [GitHub](#), you'll be working locally on your machine, you'll run commands in the AWS CLI, and you'll end up in the AWS Console and in SageMaker Ground Truth. This process might take 20 - 25 minutes. If you have any questions or need assistance, please reach out to [AWS support](#).

Prerequisites

- Install python3.8.x (e.g. You can use [pyenv](#) for python version management)
- Install [jq](#)
- Have AWS credentials configured - you can do this from the [Configuration and credential file setting](#) page
- Create a private SageMaker Ground Truth workforce to support annotation. See instructions here - [Create and Manage Private SageMaker Ground Truth Workforce](#).

Important

Make sure to record the corresponding workteam name you created in your new private workforce. You'll be required to use it during installation.

If you are installing this tool in the us-east-1 Region you can skip installing the AWS CLI and AWS SAM CLI, because it's already installed with your Python environment. You just need to create a virtual environment to use Python 3.8 in AWS Cloud9.

Get Resources from GitHub

In this step, you navigate to Git Hub to get the resources required for you to build out the annotation tooling locally. Download the annotation artifacts from [github.com/aws-samples/amazon-comprehend-semi-structured-documents-annotation-tools](#)

Run and Configure From the AWS CLI

Once you've downloaded the .zip file, unzip the annotation code. Then, head to the AWS CLI and navigate into the **ComprehendSSIEAnnotationTool** folder.

Note

The following instructions are for Linux/Ubuntu/Mac. For Windows, please install [Cygwin](#) and follow the same instructions.

1. From the folder, run:

```
make bootstrap
```

2. Run the command:

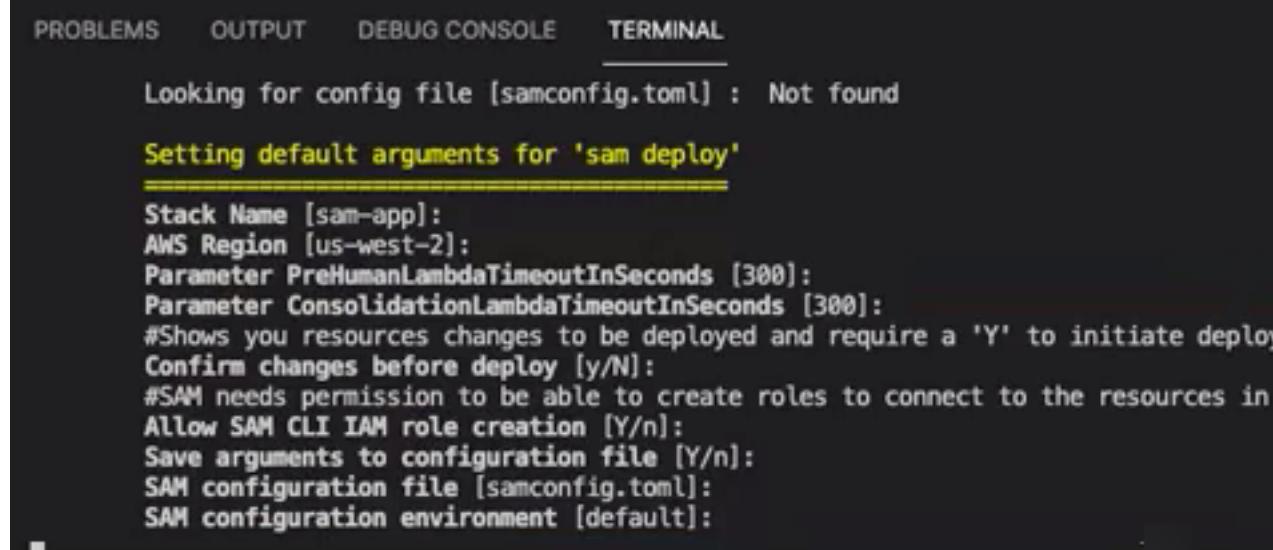
```
make build
```

3. Next, run the following command to build out the CloudFormation template. The template will be ready for your CloudFormation deployment. This CloudFormation stack will manage the [AWS Lambdas](#) you created, [AWS IAM](#) roles, and your [AWS S3](#) bucket:

```
make deploy-guided
```

When you run the command, you are prompted to name the stack. If you leave it blank, the default name is listed as **sam-app**.

4. You are presented with a set of configuration options. You can accept all of them as default, except make sure you select your correct AWS Region.



```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

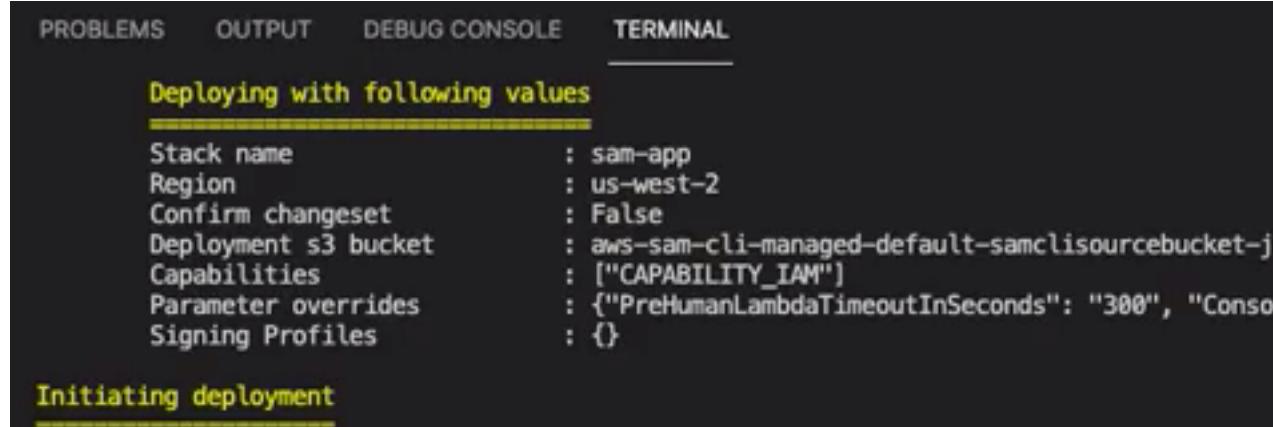
Looking for config file [samconfig.toml] : Not found

Setting default arguments for 'sam deploy'

Stack Name [sam-app]:
AWS Region [us-west-2]:
Parameter PreHumanLambdaTimeoutInSeconds [300]:
Parameter ConsolidationLambdaTimeoutInSeconds [300]:
#Shows you resources changes to be deployed and require a 'Y' to initiate deploy
Confirm changes before deploy [y/N]:
#SAM needs permission to be able to create roles to connect to the resources in
Allow SAM CLI IAM role creation [Y/n]:
Save arguments to configuration file [Y/n]:
SAM configuration file [samconfig.toml]:
SAM configuration environment [default]:

```

5. Enter the values. Your deployment successfully initiates. This step linked your AWS account and resources to your AWS S3 bucket and your IAM roles.



```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Deploying with following values

Stack name          : sam-app
Region              : us-west-2
Confirm changeset   : False
Deployment s3 bucket: aws-sam-cli-managed-default-samclisourcebucket-j
Capabilities        : ["CAPABILITY_IAM"]
Parameter overrides: {"PreHumanLambdaTimeoutInSeconds": "300", "Conso
Signing Profiles    : {}

Initiating deployment

```

Upload PDF to Amazon S3 Bucket

In step 3, you deployed a CloudFormation stack which contains an S3 bucket which is named **make-deploy-guided**. This bucket is used to store all the data that is needed for your labeling job, and it is also referred to by the Lambda IAM Execution Role policy to ensure the Lambda functions have necessary permission to access the data. The S3 bucket name can be found in **CloudFormation Stack Outputs** with

a Key of **SemiStructuredDocumentsS3Bucket**. You need to upload your source PDF documents into this Bucket.

1. In your S3 bucket, **make-deploy-guided**, create a new folder.
2. Name your folder **/src**.
3. Add your source files to the folder. These are PDF files which you will annotate to train your recognizer.

Here is a sample AWS CLI command you can use to directly upload source documents from your local directory to the S3 bucket:

- local-path-to-source-docs: the file path to local source documents
 - source-folder-name: name of a folder within the CloudFormation stack's managed S3 bucket
- ```
AWS_REGION= ` aws configure get region` ;

AWS_ACCOUNT_ID= ` aws sts get-caller-identity | jq -r '.Account' `;

aws s3 cp --recursive < local-path-to-source-docs> s3://comprehend-semi-
structured-documents- ${ AWS_REGION} -${ AWS_ACCOUNT_ID} / < source-folder-
name> /
```

## CLI - Create Your Annotation Job

At this point, you should have a private SageMaker Ground Truth workforce and you've built out and uploaded your source files to the S3 bucket. Now it's time to finish up in the CLI. After you've run **make deploy-guided**, you are presented with some settings you need to update in your job creation script

1. **input-s3-path**: S3 Uri to the source documents you uploaded to your S3 bucket
2. **cfn-name**: The CloudFormation stack name

### Important

Write down the default CloudFormation name, you will need it in the next steps.

3. **work-team-name**: The workforce name you created when you build out the private workforce in SageMaker Ground Truth
4. **job-name-prefix**: The prefix for the SageMaker Ground Truth labeling job (LIMIT: 29 characters). Extra text will be appended to job name prefix, ex. -labeling-job-task-20210902T232116
5. **entity-types**: The entities you would like to use during the labeling job (separated by commas). This should be an exhaustive list of all the entities, across all document types in the training dataset.
6. Copy and run the script with these updated details.

```
AWS_REGION=`aws configure get region`;
AWS_ACCOUNT_ID=`aws sts get-caller-identity | jq -r '.Account'`;
python bin/comprehend-ssie-annotation-tool-cli.py \
 --input-s3-path s3://comprehend-semi-structured-documents-${AWS_REGION}-${
AWS_
 --cfn-name sam-app| \
 --work-team-name <private-work-team-name> \
 --region ${AWS_REGION} \
 --job-name-prefix "${USER}-job" \
 --entity-types "EntityTypeA, EntityTypeB, EntityTypeC" \
 --annotator-metadata "key=Info,value=Sample information,key=Due Date,value=Sa...
```

The comprehend-ssie-annotation-tool-cli.py under bin/ directory is a simple wrapper command that can be used streamline the creation of a SageMaker GroundTruth Job. Under the hood, this CLI will read the source documents from the S3 path that you specify, and create a corresponding input manifest file with a single page of one source document per line, and the input manifest file is then used as input to your labeling job.

## Annotating with SageMaker Ground Truth

You are ready to start annotating your documents.

1. Login to the AWS Console and navigate to Amazon SageMaker.
2. Select **Labeling workforces** and choose **Private**.
3. Under **Private workforce summary** you should see the labeling portal sign-in URL which you created in a prerequisite step.
4. Select the sign-in URL.
5. Sign in with the username and password you selected when you created your private workforce.
6. Select the sign-in URL. Once you're in, you might not see any jobs - it can take some time to update depending on how many files you uploaded for annotation.

### Note

Firefox and Chrome are the supported browsers.

7. Once your task populates, select the task and in the top right corner choose **Start working** to open to annotation screen.

On the annotation screen you will see one of your documents open. Above the document, you'll see the entity types you provided during set up. To the right of your entity types you will see an arrow to navigate through your documents. Once you have finished annotating your document and before you move on to the next one, make sure you select **Submit** on the bottom right.

You also have options to **Remove**, **Undo**, or **Auto tag** your annotations on each document. To use auto tag, simply annotate a word which is an instance of one of your entity types. If you select Auto tag, all of the other instances of that entity type (for example, instances of the same company name) will be automatically annotated with that entity type (CompanyName).

Here is an example of an annotation screen. You can see the entity types at the top above the PDF - those will change depending on what entitiy types you listed during setup. On the top right, you can move through the PDFs you have. Remember - after you annotate each PDF, select **Submit** on the bottom of the screen to save your annotations.

Instructions   Shortcuts

Labeling Task: NER ▾ **OFFERING\_PRICE** **OFFERED\_SHARES**

*Table of Contents*

DIL

If you purchase units in this offering, you will experience dilution to the ex value to the warrants) and the net tangible book value per share of our common stock.

Our net tangible book value as of June 30, 2017 was approximately \$15.9 equal to our total tangible assets minus total liabilities, all divided by the number of **OFFERED\_SHARES** **OFFERING**.

After giving effect to the sale of 3,265,309 units at a price of \$2.45 per unit payable by us, and attributing no value to the warrants, our as adjusted net tangible per share of common stock, as of June 30, 2017. This represents an immediate increase stockholders and an immediate dilution of approximately \$1.714 per share to new investors.

Public offering price per unit

Net tangible book value per share as of June 30, 2017

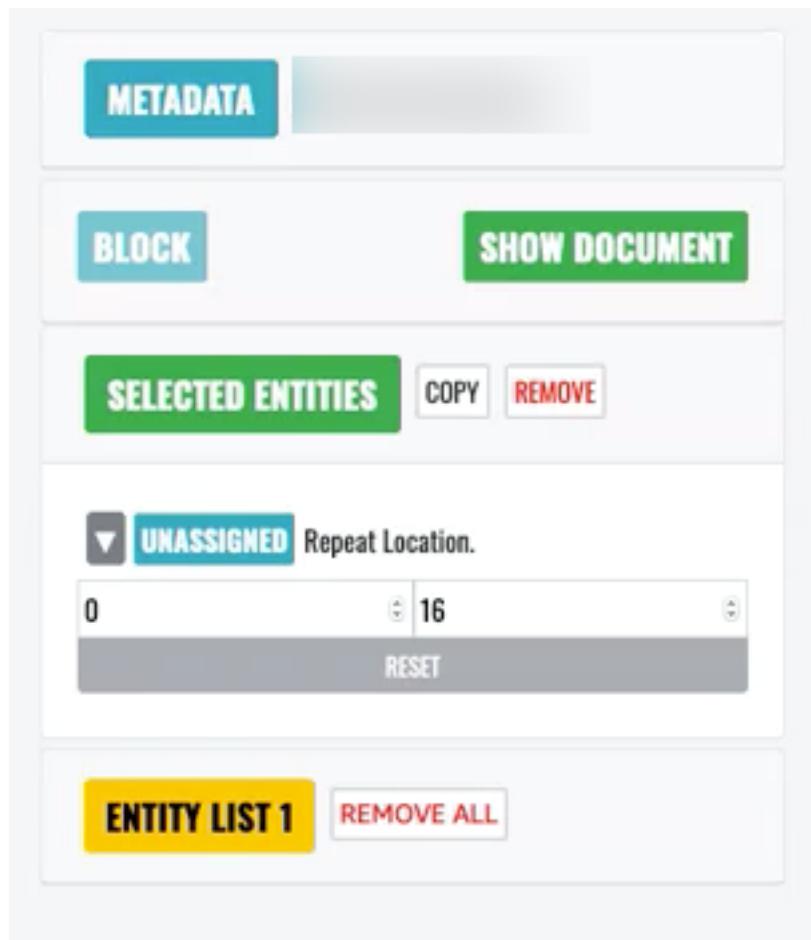
Increase per share attributable to this offering

As adjusted net tangible book value per share as of June 30, 2017, after giving effect to the sale of 3,265,309 units at a price of \$2.45 per unit payable by us, and attributing no value to the warrants, our as adjusted net tangible per share of common stock, as of June 30, 2017. This represents an immediate increase stockholders and an immediate dilution of approximately \$1.714 per share to new investors.

The foregoing table and discussion is based on 28,452,305 shares outstanding.

- 1,937,871 shares of our common stock subject to outstanding options and warrants.
- 54,300 shares of our common stock subject to outstanding restricted stock units.

In this image, you see the details from the right panel of the annotation tool. You can edit, copy, remove and reset your annotations from this section. To learn about PDF annotation best practices, please see [Annotation Best Practices \(p. 158\)](#)



## Annotation Best Practices

There are a number of things to consider to get the best result when using annotations, including:

- Annotate your data with care and verify that you annotate every mention of the entity. Imprecise annotations can lead to poor results.
- In general, more annotations will lead to better results.
- Input data should not contain duplicates, like a duplicate of a PDF you are going to annotate. Presence of a duplicate sample might result in test set contamination and could negatively affect the training process, model metrics, and model behavior.
- Make sure that all of your documents are annotated, and that the documents without annotations are due to lack of legitimate entities, not due to negligence. For example, if you have a document that says "J Doe has been an engineer for 14 years", you should also provide an annotation for "J Doe" as well as "John Doe". Failing to do so will confuse the model and might lead to not recognizing "J Doe" as ENGINEER. This should be consistent within the same document and across documents.
- To get started you need 250 documents and 100 annotations per entity. This minimum of 250 documents helps to ensure prediction quality. With more training data, you are more likely to produce a higher-quality model. If you want higher accuracy, we recommend increasing the volume of annotated data by 10% to further improve the accuracy. This improvement might be best visible to you by running inference on a held-out test set which remains unchanged and can be tested by different models. In this way you can compare successive models.

- Provide documents that resemble real use cases as closely as possible. Synthesized data with repetitive patterns should be avoided. The input data should be as diverse as possible to avoid overfitting and help the underlying model better generalize on real examples.
- It is important that documents should be diverse in terms of word count. For example, if all documents in the training data are short, the resulting model will have difficulty predicting entities in longer documents.
- Try and give the same data distribution for training as you expect to be using when you're actually detecting your custom entities (inference time). For example, at inference time, if you expect to be sending us documents that have no entities in them, this should also be part of your training document set.

Additional suggestions can be found at [Improving Custom Entity Recognizer Performance](#).

## Entity Lists (Plain Text Only)

An entity list for custom entity recognition needs a comma-separated value (CSV) file, with the following columns:

- **Text**—The text of an entry example exactly as seen in the accompanying document corpus.
- **Type**—The customer-defined entity type. Entity types must be an uppercase, underscore separated string such as MANAGER or SENIOR\_MANAGER. Up to 25 entity types can be trained per model.

The file `documents.txt` contains four lines:

```
Jo Brown is an engineer in the high tech industry.
John Doe has been a engineer for 14 years.
Emilio Johnson is a judge on the Washington Supreme Court.
Our latest new employee, Jane Smith, has been a manager in the industry for 4 years.
```

The CSV file with the list of entities has the following lines:

```
Text, Type
Jo Brown, ENGINEER
John Doe, ENGINEER
Jane Smith, MANAGER
```

### Note

In the entities list, the entry for Emilio Johnson is not present because it does not contain either the ENGINEER or MANAGER entity.

A minimum of 200 entity matches are needed per entity in the entity list to train a model for custom entity recognition.

### Creating your data files

It is important that your entity list be in a properly configured CSV file so your chance of having problems with your entity list file is minimal. To manually configure your CSV file, the following must be true:

- UTF-8 encoding must be explicitly specified, even if its used as a default in most cases.
- It must include the column names: `Type` and `Text`.

We highly recommend that CSV input files are generated programmatically to avoid potential issues.

The following example uses Python to generate a CSV for the annotations shown above:

```
import csv
with open("./entitylist/entitylist.csv", "w", encoding="utf-8") as csv_file:
 csv_writer = csv.writer(csv_file)
 csv_writer.writerow(["Text", "Type"])
 csv_writer.writerow(["Jo Brown", "ENGINEER"])
 csv_writer.writerow(["John Doe", "ENGINEER"])
 csv_writer.writerow(["Jane Smith", "MANAGER"])
```

## Getting the Best Results

### Getting the Best Results

There are a number of things to consider to get the best result when using an entity list, including:

- The order of the entities in your list has no effects on model training.
- Use entity list items that cover 80%-100% of positive entity examples mentioned in the unannotated corpus of documents.
- Avoid entity examples that match non-entities in the document corpus by removing common words and phrases. Even a handful of incorrect matches can significantly affect the accuracy of your resulting model. For example, a word like *the* in the entity list will result in a high number of matches which are unlikely to be the entities you are looking for and thus will significantly affect your accuracy.
- Input data should not contain duplicates. Presence of duplicate samples might result into test set contamination and therefore negatively affect training process, model metrics, and behavior.
- Provide documents that resemble real use cases as closely as possible. Don't use toy data or synthesized data for production systems. The input data should be as diverse as possible to avoid overfitting and help underlying model better generalize on real examples.
- The entity list is case sensitive, and regular expressions are not currently supported. However, the trained model can often still recognize entities even if they do not match exactly to the casing provided in the entity list.
- If you have an entity that is a substring of another entity (such as "Smith" and "Jane Smith"), provide both in the entity list.

Additional suggestions can be found at [Improving Custom Entity Recognizer Performance \(p. 170\)](#)

## Detecting Custom Entities with an Asynchronous Batch Job with Amazon Comprehend

Once you have a trained model, use the [StartEntitiesDetectionJob \(p. 359\)](#) operation to detect custom entities in your documents.

### Before you begin

Before you can detect custom entities, you must have a custom entity recognition model.

For more information about these models, see [the section called "Training Custom Entity Recognizers" \(p. 148\)](#). For the steps to train a model, see [the section called "Creating a Custom Entity Recognizer Using the Console - CSV Format" \(p. 40\)](#).

Using this operation, you provide the same information as you would when detecting preset entities. However, in addition to the input and output locations (S3 buckets), you also provide the EntityRecognizerArn, which is the Amazon Resource Name (ARN) of the trained model. This ARN is supplied by the response to the [CreateEntityRecognizer \(p. 249\)](#) operation.

You can examine one document or many, and each model can be trained on up to 25 custom entities at a time. You can search for up to 25 entities per [StartEntitiesDetectionJob](#) operation..

To detect custom entities in a document set, you use the following request syntax:

The examples are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend start-entities-detection-job \
 --entity-recognizer-arn "entity recognizer arn" \
 --job-name job name \
 --data-access-role-arn "data access role arn" \
 --language-code en \
 --input-data-config "S3Uri=s3://Bucket Name/Bucket Path" \
 --output-data-config "S3Uri=s3://Bucket Name/Bucket Path/" \
 --region region
```

Amazon Comprehend will respond with the JobID and JobStatus and will return the output from the job in the S3 bucket that you specified in your request. This output will be similar to the following:

```
{"File": "50_docs", "Line": 0, "Entities": [{"BeginOffset": 0, "EndOffset": 22, "Score": 0.9763959646224976, "Text": "John Johnson", "Type": "JUDGE"}]}
{"File": "50_docs", "Line": 1, "Entities": [{"BeginOffset": 11, "EndOffset": 15, "Score": 0.9615424871444702, "Text": "Thomas Kincaid", "Type": "JUDGE"}]}
```

## Detecting Custom Entities in Real Time with Amazon Comprehend

With Amazon Comprehend, you can quickly detect custom entities in individual text documents by running real-time analysis. Unlike asynchronous batch jobs that analyze large documents or large sets of documents, real-time analysis is useful for applications that process small bodies of text as they arrive. For example, you can immediately detect custom entities in social media posts, support tickets, or customer reviews.

After you train a custom entity recognition model, you enable real-time analysis by creating an endpoint. After you create the endpoint, your custom model is available for real-time analysis, and you can detect entities by using the Amazon Comprehend console, the Amazon Comprehend API, the AWS CLI, or the AWS SDKs.

### Before you begin

Before you can detect custom entities, you must train a custom entity recognition model. For more information about these models, see [Training Custom Entity Recognizers \(p. 148\)](#). For the steps to train a model, see [Creating a Custom Entity Recognizer Using the Console - CSV Format \(p. 40\)](#).

## Creating an Endpoint for Custom Entity Detection

After you train a custom entity recognition model, you can use that model to quickly detect custom entities in individual documents. First, you must create an *endpoint*, which makes your model available for real-time analysis.

To meet your text processing needs, you assign *inference units* to the endpoint, and each unit allows a throughput of 100 characters per second for up to 2 documents per second. You can then adjust the throughput up or down. The cost of real-time analysis is based on the throughput of an endpoint and the duration of time it is active. For more information on endpoint cost, see [Amazon Comprehend Pricing](#).

After you create an endpoint, you can monitor it with Amazon CloudWatch, update it to change its inference units, or delete it when you no longer need it. For more information, see [Managing Endpoints with Amazon Comprehend \(p. 170\)](#).

## Creating an Endpoint with the Console

### To create an endpoint (console)

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Endpoints** and choose the **Create endpoint** button. A **Create endpoint** screen opens.
3. Give the endpoint a name. The name must be unique within the AWS Region and account.
4. Choose a custom model you want to attach the new endpoint to. From the dropdown, you can search by model name.

#### Note

You need to create a model before you can attach an endpoint to it. If you don't have a model yet, go to **Custom classification** or **Custom entity recognition** to create one.

5. (Optional) To add a tag to the endpoint, enter a key-value pair under **Tags** and choose **Add tag**. To remove this pair before creating the endpoint, choose **Remove tag**
6. Enter the number of inference units (IUs) to assign to the endpoint. Each unit represents a throughput of 100 characters per second for up to 2 documents per second.
7. (Optional) If you are creating a new endpoint, you have the option to use the IU estimator. It can be difficult to understand how many inference units you require depending on the throughput or the number of characters you want to analyze per second—especially at scale. This optional step can help you determine how many IUs to request.

#### Note

The range for IUs is 1 to 10. The maximum characters you can analyze per second is 1000.

8. From the **Purchase summary**, review your estimated hourly, daily, and monthly endpoint cost.
9. Select the checkbox if you understand that you will be charged for the endpoint from the time it starts until it is deleted.

10. Choose **Create endpoint**

## Creating an Endpoint with the AWS CLI

To create an endpoint by using the AWS CLI, use the `create-endpoint` command:

```
$ aws comprehend create-endpoint \
> --desired-inference-units number of inference units \
> --endpoint-name endpoint name \
> --model-arn arn:aws:comprehend:region:account-id:model/example \
> --tags Key=Key,Value=Value
```

If your command succeeds, Amazon Comprehend responds with the endpoint ARN:

```
{
 "EndpointArn": "Arn"
}
```

For more information about this command, its parameter arguments, and its output, see [create-endpoint](#) in the AWS CLI Command Reference

## Running Real-Time Custom Entity Detection

After you create an endpoint for your custom entity recognizer model, you can run real-time analysis to quickly detect entities in individual bodies of text.

## Detecting Entities with the Console

Complete the following steps to detect custom entities in your text by using the Amazon Comprehend console.

1. Sign in to the AWS Management Console and open the Amazon Comprehend console at <https://console.aws.amazon.com/comprehend/>.
2. From the left menu, choose **Real-time analysis**.
3. In the **Input text** section, for **Analysis type**, choose **Custom**.
4. For **Select endpoint**, choose the endpoint that is associated with the entity-detection model that you want to use.
5. Under **Input text**, provide the text you want to analyze.
6. Choose **Analyze**. The text analysis based on your custom model is displayed, along with a confidence assessment of the analysis.

## Detecting Entities with the AWS CLI

To detect custom entities by using the AWS CLI, use the `detect-entities` command:

```
$ aws comprehend detect-entities \
> --endpoint-arn arn \
> --language-code en \
> --text "Andy Jassy is the CEO of Amazon."
```

If your command succeeds, Amazon Comprehend responds with the analysis. For each entity that Amazon Comprehend detects, it provides the entity type, text, location, and confidence score.

For more information about this command, its parameter arguments, and its output, see [detect-entities](#) in the AWS CLI Command Reference

## Tagging Custom Entity Recognizers

A tag is a key-value pair which can be added to a Amazon Comprehend resource as metadata.

Tags can be used for organizing your resources as well as helping you to search and filter by tag. Using tag-based access control can allow for finer control than API-level control as well as more dynamic control than resource-based access control. IAM policies can be created that allow or disallow an operation based on tags provided in the request (request-tags) or tags on the resource that is being operated on (resource-tags). For more information on using IAM roles for this, [Controlling Access Using Tags](#) in the *IAM User Guide*

For instance, you could add tags such as 'Sales' or 'Legal' to different custom entity recognizers to break down documents into dynamic categories for your company.

Multiple tags can be added to an Amazon Comprehend resource, either when you create it, or later. Up to 50 tags can be associated with each resource.

Each tag key must be unique for a specific resource. However, the same key can be used for different resources. The tag value can be used to make the tag more specific, but is optional.

The following options are available when using tags with Amazon Comprehend:

- [Tagging a resource when you originally create it \(p. 164\)](#)

- [Tagging a resource after it's been created \(p. 164\)](#)
- [Modifying an existing tag \(p. 165\)](#)
- [Removing tags from a resource \(p. 166\)](#)
- [Viewing all tags associated with a resource \(p. 167\)](#)
- [Tagging restrictions \(p. 167\)](#)

## Tagging a resource when you originally create it

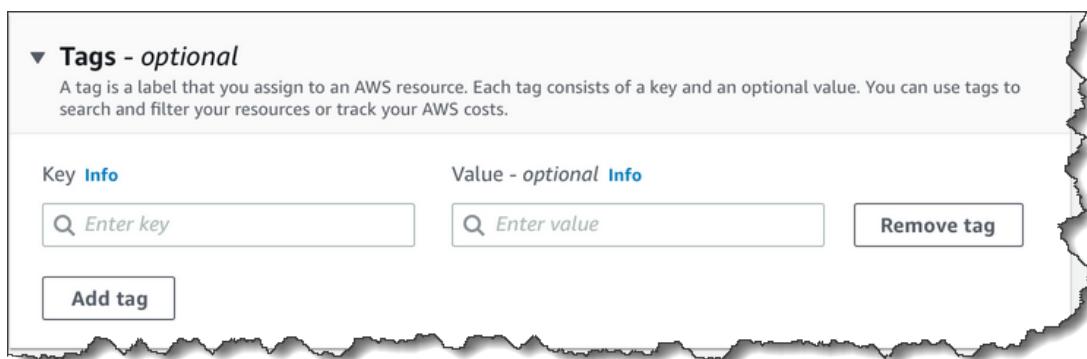
When you want to add one or more tags to a custom entity recognizer when you originally create it, you can do this when training the recognizer.

### To tag a resource when creating it

1. In the optional **Tags** section, enter the key-value pair for your tag.

**Note**

Remember the key must be unique for the given resource. The value is optional.



2. Choose **Add tag** to associate the tag with your recognizer. You can do this multiple times to add more than one tag.

Once you've added a tag, you can remove it before training the recognizer by choosing **Remove tag**.

When using the APIs, you can do this using the `CreateEntityRecognizer` operation.

## Tagging a resource after it's been created

You can also easily add a tag to an existing custom entity recognizer using the resource list or the detail page for a specific recognizer.

### To add a tag to an existing custom entity recognizer

1. On the Custom Entity Recognizer resource list, select the recognizer to which you want to add the tag, and then choose **Manage tags**.

| Name            | Custom entity type | Training started         | Training ended           | Status               |
|-----------------|--------------------|--------------------------|--------------------------|----------------------|
| MyRecognizer_31 | PRODUCT_CODE31     | Tue Feb 27 2018 13:47:29 | Tue Feb 27 2018 13:47:29 | <span>Trained</span> |

Alternatively, choose **Manage tags** in the **Tags** section of a specific recognizer's details page.

| Key  | Value   |
|------|---------|
| env  | staging |
| name | vault76 |
| role | -       |

- Under **Tags**, enter the key-value pair for your tag.

**Note**

Remember the key must be unique for the given resource. The value is optional.

**Tags**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

**Key** [Info](#)      **Value - optional** [Info](#)

- Choose **Add tag** to associate your tag with the recognizer. You can do this multiple times to add more than one tag.

Once you've added a tag, you can remove it before training the recognizer by choosing **Remove tag**.

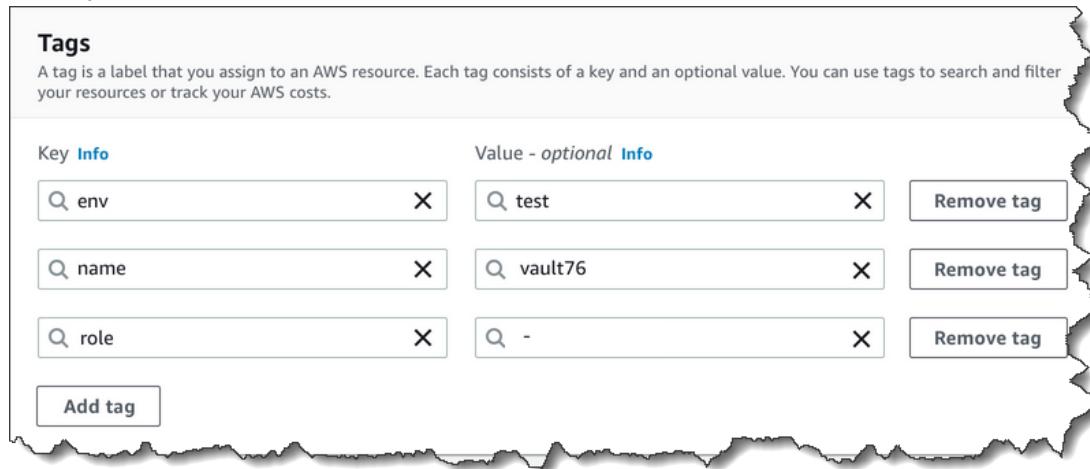
- Choose **Save**.

## Modifying an existing tag

You can remove an existing using the resource list or the detail page for a specific resource.

### To modify an existing tag

- Under **Tags**, choose the tag you want to change and choose the X for the key or value box. Enter the new key or value.



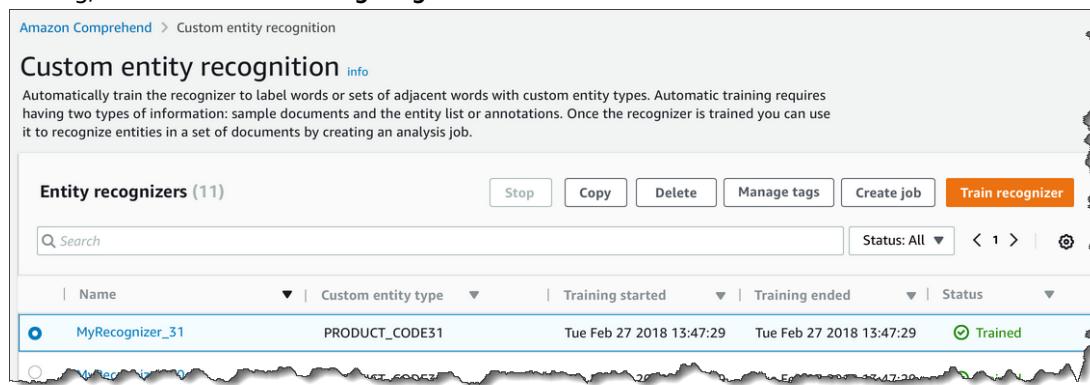
- Make any additional changes desired to the tags.
- Choose **Save**.

## Removing tags from a resource

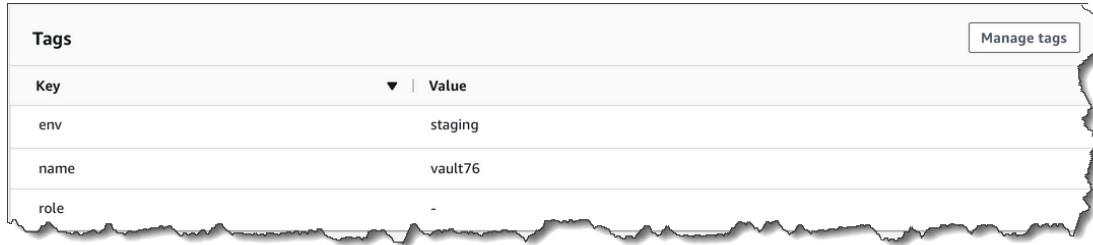
You can remove a tag from an existing custom entity recognizer using the resource list or the detail page for a specific recognizer.

### To remove a tag from an existing custom entity recognizer

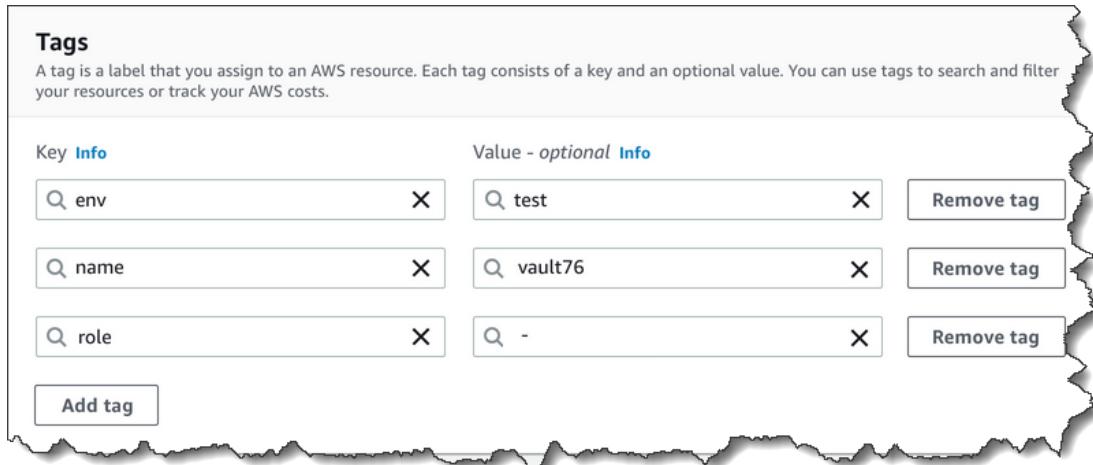
- On the Custom Entity Recognizer resource list, select the recognizer from which you want to remove the tag, and then choose **Manage tags**.



Alternatively, choose **Manage tags** in the **Tags** section of a specific recognizer's details page.



2. Next to the tag that you want to remove, choose **Remove tag**.



3. Choose **Save**.

## Viewing all tags associated with a resource

### To view all tags associated with a specific resource

- On the Custom Entity Recognizer resource list, select the recognizer for which you want to view the tags, and then choose **Manage tags**.

## Tagging restrictions

The following basic restrictions apply to Custom Entity Recognizer tags:

- The maximum number of tags per recognizer is 50.
- The maximum length of a tag key is 127 characters.
- The maximum length of a tag value is 255 characters.
- Tag keys and values are case sensitive.
- The `aws :` prefix is reserved for AWS use; you cannot add, edit, or delete tags whose key begins with `aws :.` Tags that begin with `aws : do not count against your tags-per-resource limit.`
- If you plan to use your tagging schema across multiple services and resources, remember that other services may have different restrictions for allowed characters. Refer to the documentation for that service.

## Custom Entity Recognizer Metrics

Amazon Comprehend provides you with metrics to help you estimate how well an entity recognizer should work for your job. They are based on training the recognizer model, and so while they accurately represent the performance of the model during training, they are only an approximation of the API performance during entity discovery.

Metrics are returned any time metadata from a trained entity recognizer is returned.

Amazon Comprehend supports training a model on up to 25 entities at a time. When metrics are returned from a trained entity recognizer, scores are computed against both the recognizer as a whole (global metrics) and for each individual entity (entity metrics).

Three metrics are available, both as global and entity metrics:

- **Precision**

This indicates the fraction of entities produced by the system that are correctly identified and correctly labeled. This shows how many times the model's entity identification is truly a good identification. It is a percentage of the total number of identifications.

In other words, precision is based on *true positives (tp)* and *false positives (fp)* and it is calculated as  $precision = tp / (tp + fp)$ .

For example, if a model predicts that two examples of an entity are present in a document, where there's actually only one, the result is one true positive and one false positive. In this case,  $precision = 1 / (1 + 1)$ . The precision is 50%, as one entity is correct out of the two identified by the model.

- **Recall**

This indicates the fraction of entities present in the documents that are correctly identified and labeled by the system. Mathematically, this is defined in terms of the total number of correct identifications *true positives (tp)* and missed identificcations *false negatives (fn)*.

It is calculated as  $recall = tp / (tp + fn)$ . For example if a model correctly identifies one entity, but misses two other instances where that entity is present, the result is one true positive and two false negatives. In this case,  $recall = 1 / (1 + 2)$ . The recall is 33.33%, as one entity is correct out of a possible three examples.

- **F1 Score**

This is a combination of the Precision and Recall metrics, which measures the overall accuracy of the model for custom entity recognition. The F1 score is the harmonic mean of the Precision and Recall metrics:  $F1 = 2 * Precision * Recall / (Precision + Recall)$ .

**Note**

Intuitively, the harmonic mean penalizes the extremes more than the simple average or other means (example:  $precision = 0$ ,  $recall = 1$  could be achieved trivially by predicting all possible spans. Here, the simple average would be 0.5, but F1 would penalize it as 0).

In the examples above,  $precision = 50\%$  and  $recall = 33.33\%$ , therefore  $F1 = 2 * 0.5 * 0.3333 / (0.5 + 0.3333)$ . The F1 Score is .3975, or 39.75%.

### Global and Individual Entity Metrics

The relationship between global and individual entity metrics can be seen when analyzing the following sentence for entities that are either a *place* or a *person*

John Washington and his friend Smith live in San Francisco, work in San Diego, and own a house in Seattle.

In our example, the model makes the following predictions.

```
John Washington = Person
Smith = Place
San Francisco = Place
San Diego = Place
Seattle = Person
```

However, the predictions should have been the following.

```
John Washington = Person
Smith = Person
San Francisco = Place
San Diego = Place
Seattle = Place
```

The individual entity metrics for this would be:

```
entity: Person
True positive (TP) = 1 (because John Washington is correctly predicted to be a Person).
False positive (FP) = 1 (because Seattle is incorrectly predicted to be a Person, but is actually a Place).
False negative (FN) = 1 (because Smith is incorrectly predicted to be a Place, but is actually a Person).
Precision = 1 / (1 + 1) = 0.5 or 50%
Recall = 1 / (1+1) = 0.5 or 50%
F1 Score = 2 * 0.5 * 0.5 / (0.5 + 0.5) = 0.5 or 50%

entity: Place
TP = 2 (because San Francisco and San Diego are each correctly predicted to be a Place).
FP = 1 (because Smith is incorrectly predicted to be a Place, but is actually a Person).
FN = 1 (because Seattle is incorrectly predicted to be a Person, but is actually a Place).
Precision = 2 / (2+1) = 0.6667 or 66.67%
Recall = 2 / (2+1) = 0.6667 or 66.67%
F1 Score = 2 * 0.6667 * 0.6667 / (0.6667 + 0.6667) = 0.6667 or 66.67%
```

The global metrics for this would be:

Global:

```
Global:
TP = 3 (because John Washington, San Francisco and San Diego are predicted correctly.
This is also the sum of all individual entity TP).
FP = 2 (because Seattle is predicted as Person and Smith is predicted as Place. This is the sum of all individual FP).
FN = 2 (because Seattle is predicted as Person and Smith is predicted as Place. This is the sum of all individual FN).
Global Precision = 3 / (3+2) = 0.6 or 60%
(Global Precision = Global TP / (Global TP + Global FP))
Global Recall = 3 / (3+2) = 0.6 or 60%
(Global Recall = Global TP / (Global TP + Global FN))
Global F1Score = 2 * 0.6 * 0.6 / (0.6 + 0.6) = 0.6 or 60%
(Global F1Score = 2 * Global Precision * Global Recall / (Global Precision +
```

Global Recall))

## Improving Custom Entity Recognizer Performance

These metrics provide an insight into how accurately the trained model will perform when you use it to identify entities. Here are a few options you can use to improve your metrics if they are lower than your expectations:

1. Depending on whether you use [Annotations \(p. 150\)](#) or [Entity Lists \(Plain Text Only\) \(p. 159\)](#), make sure to follow the guidelines in the respective documentation to improve data quality. If you observe better metrics after improving your data and re-training the model, you can keep iterating and improving data quality to achieve better model performance.
2. If you are using an Entity List, consider using Annotations instead. Manual annotations can often improve your results.
3. If you are sure there is not a data quality issue, and yet the metrics remain unreasonably low, please submit a support request.

## Managing Endpoints with Amazon Comprehend

In Amazon Comprehend, endpoints make your custom models available for real-time classification or entity detection. After you create an endpoint, you can make changes to it as your business needs evolve. For example, you can monitor your endpoint utilization and apply auto scaling to automatically set endpoint provisioning to fit your capacity needs. You can manage all your endpoints from a single view, and when you no longer need an endpoint you can delete it to save costs.

Before you can manage an endpoint, you must create one. For more information, see the following procedures:

- [Creating an Endpoint for Custom Classification \(p. 135\)](#).
- [Creating an Endpoint for Custom Entity Detection \(p. 161\)](#).

### Topics

- [Endpoints Overview with Amazon Comprehend \(p. 170\)](#)
- [Monitoring an Endpoint with Amazon Comprehend \(p. 171\)](#)
- [Updating an Endpoint with Amazon Comprehend \(p. 172\)](#)
- [Using Trusted Advisor with Amazon Comprehend \(p. 174\)](#)
- [Deleting an Endpoint with Amazon Comprehend \(p. 176\)](#)
- [Auto Scaling with Endpoints \(p. 177\)](#)

## Endpoints Overview with Amazon Comprehend

The endpoints page from Amazon Comprehend console provides you a global view of your endpoints. From the endpoints overview page, you can view all of your endpoints in one place to understand your endpoint usage versus your actual resource usage. On the top right of the endpoints page you can specify what endpoints you want to view—all of them, custom classifier endpoints, or your custom entity endpoints.

You can create, update, monitor, and delete endpoints from this page. From the endpoints overview section, you can view a list of your endpoints, what custom models the endpoints are hosting, their

creation time, the provisioned throughput, and the status of the endpoint. When you select a specific endpoint from the endpoint overview table, the endpoint details are displayed.

Also, if you are a [AWS Business Support](#) or an [AWS Enterprise Support](#) customer, you have access to Trusted Advisor checks specific to your endpoints. To learn more, see [Using Trusted Advisor with Amazon Comprehend \(p. 174\)](#). For a complete list of checks and descriptions, see the [Trusted Advisor Best Practices](#).

For more information on managing your endpoints, see the following topics.

- [Monitoring an Endpoint with Amazon Comprehend \(p. 171\)](#)
- [Updating an Endpoint with Amazon Comprehend \(p. 172\)](#)
- [Using Trusted Advisor with Amazon Comprehend \(p. 174\)](#)
- [Deleting an Endpoint with Amazon Comprehend \(p. 176\)](#)

**Important**

The cost for real-time custom classification is based on both the throughput you set and the length of time the endpoint is active. If you are no longer using the endpoint, or are not using it for an extended period, you should set up an auto scaling policy to reduce your costs. Or, if you are no longer using an endpoint you can delete the endpoint to avoid incurring additional cost. For more information, see [Auto Scaling with Endpoints \(p. 177\)](#).

## Monitoring an Endpoint with Amazon Comprehend

Depending on your needs, you might need to adjust the throughput of your endpoint after creating it. This can be achieved by updating the endpoint's inference units (IUs). When you edit an endpoint, you can add more IUs to an endpoint, or you can decrease the IUs. A single endpoint can have 1 to 10 IUs.

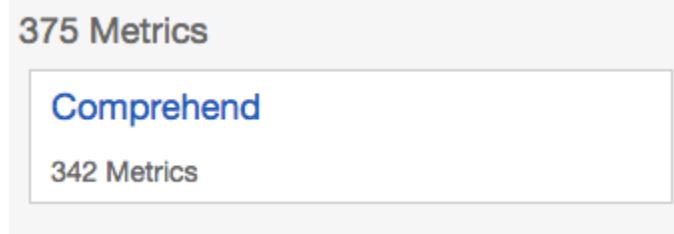
For more information on updating your endpoint, see [the section called "Updating an Endpoint" \(p. 172\)](#).

To view all of your endpoints, see [the section called "Endpoints Overview" \(p. 170\)](#).

You can determine how to best adjust your endpoint's throughput by monitoring its usage with the Amazon CloudWatch console.

### Monitor your endpoint usage with CloudWatch

1. Sign in to the AWS Management Console and open the [CloudWatch console](#).
2. On the left, choose **Metrics** and select **All metrics**.
3. Under **All metrics**, choose **Comprehend**.



The CloudWatch console will show the dimensions for the metrics.

4. Choose the **EndpointArn** dimension.

## 342 Metrics

EndpointArn

342 Metrics

The console displays **ConsumedInferenceUnits**, **ProvisionedInferenceUnits**, and **InferenceUtilization** for each of your endpoints.

| Metric Name               |
|---------------------------|
| ConsumedInferenceUnits    |
| ProvisionedInferenceUnits |
| InferenceUtilization      |

5. Set the Statistic column for **ConsumedInferenceUnits** and **InferenceUtilization** to **Sum**.
6. Set the Statistic column for **ProvisionedInferenceUnits** to **Average**.
7. Change the Period column for all metrics to **1 Minute**.
8. Select **InferenceUtilization** and select the arrow to move it to a separate **Y Axis**.

Your graph is ready for analysis.

Based on the CloudWatch metrics, you can also set up auto scaling to automatically adjust the throughput of your endpoint. For more information about using auto scaling with your endpoints, see [Auto Scaling with Endpoints \(p. 177\)](#).

- **ProvisionedInferenceUnits** - This metric represents the number of average provisioned IUs at the time the request was made.
- **ConsumedInferenceUnits** - This is based on the usage of each request submitted to the service that was successfully processed. This can be helpful when you compare what you're consuming against your provisioned IUs. The value for this metric is calculated by taking the number of characters processed and dividing it by the number of characters that can be processed in a minute for 1 IU.
- **InferenceUtilization** - This is emitted per request. This value is calculated by taking the consumed IUs defined in **ConsumedInferenceUnits** and dividing it by **ProvisionedInferenceUnits** and converting to a percentage out of 100.

### Note

All of the metrics are emitted only for successful requests. The metric won't appear if it's from a request that is throttled or fails with an internal server error or a customer error.

## Updating an Endpoint with Amazon Comprehend

Frequently, the level of throughput you need changes after creating an endpoint, or your first estimation of your needs changes. When this happens, it may be necessary to update your endpoint to adjust the throughput up or down. Throughput is governed by the number of inference units with which you've

provisioned your endpoint. Each inference unit represents a throughput of 100 characters per second for up to 2 documents per second. You might also want to update the version of the model associated with the endpoint. When you edit an endpoint, you can choose a different version of the model for the endpoint.

It can also be helpful to add tags to your endpoint to help keep them organized. This can also be done while updating your endpoint. For more information on endpoints, see [Tagging Custom Classifiers \(p. 137\)](#)

### To update an endpoint (console)

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Endpoints**.
3. From the **Classifiers** list, choose the name of the custom model from which you want to update the endpoint and follow the link. The model details page displays.
4. From the model details page, select the version details. The endpoints list displays.
5. Select the endpoint checkbox for your endpoint. At the top right of the endpoints table, select the **Actions** icon.
6. Choose **Edit**. You can update provisioned IUs and edit tags.
7. Save your changes.
8. To edit the number of inference units with which the endpoint is provisioned, choose **Edit**.
9. Enter the updated number of inference units to assign to the endpoint. Each unit represents a throughput of 100 characters per second. You can assign up to a maximum of 10 inference units per endpoint.

#### Note

The cost of using an endpoint is based on the amount of time operating and the throughput (based on the number of inference units). Increasing the number of inference units will thus increase the cost of operation. For more information, see [Amazon Comprehend Pricing](#).

10. Choose **Edit endpoint**. The endpoint details page is displayed.
11. Confirm that the endpoint is updating by choosing the model name from the breadcrumbs at the top of the page. On the custom model details page, navigate to the **Endpoints** list and verify that it shows **Updating** next to the endpoint. When the update is complete, it will show **Ready**.

The following example demonstrates using the *UpdateEndpoint* operation with the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend update-endpoint \
--desired-inference-units updated number of inference units \
--desired-model-arn arn:aws:comprehend:region:account-id:model type/model name \
--desired-data-access-role-arn arn:aws:iam:account id:role/role name \
--endpoint-arn arn:aws:comprehend:region:account id:endpoint/endpoint name
```

If the action is successful, Amazon Comprehend responds with an HTTP 200 response with an empty HTTP body.

12. To edit the custom model attached to your endpoint, from the custom model details page, navigate to the **Endpoints** list.
13. Select the endpoint you want to change and select **Edit**.
14. From the endpoint settings page, under **Select classifier model** or **Select recognizer model** depending on your endpoint, you can search for a model in the dropdown. Select the model you want.

15. Under **Select version** you can search for the model version you want. Select the version.
16. Select **Edit endpoint** to save.

## Using Trusted Advisor with Amazon Comprehend

AWS Trusted Advisor is an online tool that provides recommendations to help you provision your resources following AWS best practices.

If you have a Basic or Developer Support plan, you can use the Trusted Advisor console to access all checks in the Service Limits category and six checks in the Security category. If you have a Business or Enterprise Support Plan, you can use the Trusted Advisor console and the [AWS Support API](#) to access all of the Trusted Advisor checks.

Amazon Comprehend supports the following Trusted Advisor checks to help customers optimize the cost and the security of their Amazon Comprehend endpoints by providing actionable recommendations.

### Amazon Comprehend Underutilized Endpoints

The **Amazon Comprehend Underutilized Endpoints** check evaluates the throughput configuration of your endpoints. This check alerts you when endpoints are not actively used for real-time inference requests. An endpoint that isn't used for more than 15 days is considered underutilized. All endpoints accrue charges based on both the throughput set and the length of time that the endpoint is active. For the endpoint not used in last 15 days, we recommend that you define a scaling policy for the resource using [Application Autoscaling](#). For an endpoint that hasn't been used in the last 30 days and does have an auto scaling policy defined we recommend that you use asynchronous inference or delete it. These check results are automatically refreshed once every day and can be viewed under the **CostOptimization** category on the Trusted Advisor console.

#### To view the utilization status of all your endpoints and the corresponding recommendations

1. Sign in to the AWS Management Console and open the Trusted Advisor console.
2. In the navigation pane, choose the **CostOptimization** check category.
3. On the category page, you can view the summary for each check category:
  - **Action recommended (red)** – Trusted Advisor recommends an action for the check.
  - **Investigation recommended (yellow)** – Trusted Advisor detects a possible issue for the check.
  - **No problems detected (green)** – Trusted Advisor doesn't detect an issue for the check.
  - **Excluded items (gray)** – The number of checks that have excluded items, such as resources that you want a check to ignore.
4. Choose **Amazon Comprehend Underutilized Endpoints check** to view the check description and the following details:
  - **Alert Criteria** – Describes the threshold when a check will change status.
  - **Recommended Action** – Describes the recommended actions for this check.
  - **Resource Table:** A table that lists your endpoint details and the status for each based on your recommendations.
5. In the Resource table, if an endpoint is flagged with a **Investigation Recommended** because of a **Not used in last 30 days** warning, you can navigate to the Endpoint Details page on the Amazon Comprehend console.
  - If you do not want to use this endpoint anymore, choose **Delete**.
  - Choose **Delete** again to confirm the deletion. The custom model details page is displayed. Confirm that the endpoint you deleted shows **deleting** next to it. When it has deleted, the endpoint is removed from the **Endpoints** list.

6. In the Resource table on the Trusted Advisor console, if an endpoint is flagged with an **Investigation Recommended** status because it hasn't been used in the last 15 days, and if it has AutoScaling disabled, you can navigate to the Endpoint Details page on the Amazon Comprehend console to adjust the endpoint.
  - If you want to reduce the throughput configured for this endpoint, click **Edit**. Enter the updated number of inference units to assign to the endpoint, then select the checkbox to acknowledge and then choose **Edit Endpoint**. When the update is complete, the status will show as **Ready**.
  - If you want to automatically set endpoint provisioning on your endpoint instead of manually adjusting the throughput configuration, we recommend you use **Application Autoscaling**.
7. In the Resource table on the Trusted Advisor console, if an endpoint is flagged with the **No problems detected** status because of the **Used Actively** reason, then it implies the endpoint is being utilized actively for running real-time inference requests and no actions are recommended.

Here's an example which shows the CostOptimization category view on the Trusted Advisor console:



## Amazon Comprehend Endpoint Access Risk

The **Amazon Comprehend Endpoint Access Risk** check evaluates the AWS Key Management Service (AWS KMS) key permissions for an endpoint where the underlying model was encrypted using customer managed keys. If the customer managed key is disabled or the key policy was changed to alter the allowed permissions for Amazon Comprehend, the endpoint availability might be affected. If the key has been disabled, we recommend that you enable it. If the key policy has been altered and you wish to continue using this endpoint, we recommend that you update the key policy. The check results are automatically refreshed multiple times during the day. This check can be viewed under the **Fault Tolerance** category of the Trusted Advisor console.

### To view the AWS KMS key status of your Amazon Comprehend endpoints

1. Sign in to the AWS Management Console and open the Trusted Advisor console.
2. In the navigation pane, choose the **FaultTolerance** check category.
3. On the category page, you can view the summary for each check category:
  - **Action recommended (red)** – Trusted Advisor recommends an action for the check.
  - **Investigation recommended (yellow)** – Trusted Advisor detects a possible issue for the check.
  - **No problems detected (green)** – Trusted Advisor doesn't detect an issue for the check.
  - **Excluded items (gray)** – The number of checks that have excluded items, such as resources that you want a check to ignore.
4. Choose Amazon Comprehend Endpoint Access Risk Check and you can view the check description and the following details:
  - **Alert Criteria** – Describes the threshold when a check will change status.
  - **Recommended Action** – Describes the recommended actions for this check.
  - **Resource Table**: A table that lists your KMS encrypted endpoint details and the status for each one based on if there are recommended actions.
5. In the Resource table, if an endpoint is flagged with an **Action Recommended** status, select the link in the KMS KeyId column and you will be redirected to the corresponding AWS KMS key page.
  - To enable a disabled AWS KMS key, choose **Key Actions**, and select **Enable**.

- If the Key Status is listed as **Enabled**, update the key policy by choosing **Switch to policy view** in the Key Policy section. Edit the key policy document to provide the necessary permissions to Amazon Comprehend and then choose **Save changes**.

Here's an example of the FaultTolerance category view on the Trusted Advisor console:



These checks and their results can also be viewed by referring the Trusted Advisor section of the AWS Support API.

To learn more about setting up alarms using CloudWatch, see: [Creating Trusted Advisor alarms using CloudWatch](#). For a full set of Trusted Advisor Best Practice Checks, see: [AWS Trusted Advisor best practice checklist](#).

## Deleting an Endpoint with Amazon Comprehend

Once you no longer need your endpoint, you should delete it so that you stop incurring costs from it. You can easily create another endpoint whenever you need it from the **Endpoints** section.

### To delete an endpoint (console)

1. Sign in to the AWS Management Console and open the [Amazon Comprehend console](#).
2. From the left menu, choose **Endpoints**.
3. From the **Endpoints** table locate the endpoint you want to delete. You can search or filter all of the endpoints to find the one you need.
4. Select the endpoint checkbox for the endpoint you want to delete. At the top right of the endpoints table, select the **Actions** icon.
5. Choose **Delete**.
6. Choose **Delete** again to confirm the deletion. The endpoints page is displayed. Confirm that the endpoint you deleted shows **Deleting** next to it. When it's deleted, the endpoint is removed from the **Endpoints** list.

### To delete an endpoint (AWS CLI)

The following example demonstrates using the *DeleteEndpoint* operation with the AWS CLI.

The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```
aws comprehend delete-endpoint \
--endpoint-arn arn:aws:comprehend:<region>:<account-id> endpoint/<endpoint name>
```

If the action is successful, Amazon Comprehend responds with an HTTP 200 response with an empty HTTP body.

## Auto Scaling with Endpoints

Instead of manually adjusting the number of inference units provisioned for your document classification endpoints and entity recognizer endpoints, you can use auto scaling to automatically set endpoint provisioning to fit your capacity needs.

There are two ways to use auto scaling to adjust the number of inference units provisioned for your endpoint:

- [Target Tracking \(p. 177\)](#): Set auto scaling to adjust endpoint provisioning to fit capacity needs based on usage.
- [Scheduled Scaling \(p. 179\)](#): Set auto scaling to adjust endpoint provisioning to fit capacity needs on a specified schedule.

You can set auto scaling only with the AWS Command Line Interface (AWS CLI). For more information about auto scaling, see [What is Application Auto Scaling?](#)

### Target Tracking

With target tracking, you can adjust endpoint provisioning to fit your capacity needs based on usage. The number of inference units automatically adjust so that the utilized capacity is within a target percentage of the provisioned capacity. You can use target tracking to accommodate temporary surges of use for your document classification endpoints and entity recognizer endpoints. For more information, see [Target Tracking Scaling Policies for Application Auto Scaling](#).

#### Note

The following examples are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

### Setting up Target Tracking

To set up target tracking for an endpoint, you use AWS CLI commands to register a scalable target and then create a scaling policy. The scalable target defines inference units as the resource used to adjust endpoint provisioning, and the scaling policy defines the metrics that control the auto scaling of the provisioned capacity.

#### To set up target tracking

1. Register a scalable target. The following examples register a scalable target to adjust endpoint provisioning with a minimum capacity of 1 inference unit and a maximum capacity of 2 inference units.

For a document classification endpoint, use the following AWS CLI command:

```
aws application-autoscaling register-scalable-target \
--service-namespace comprehend \
--resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
--scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
\
--min-capacity 1 \
--max-capacity 2
```

For an entity recognizer endpoint, use the following AWS CLI command:

```
aws application-autoscaling register-scalable-target \
```

```
--service-namespace comprehend \
--resource-id arn:aws:comprehend:region:account-id:entity-recognizer-endpoint/name
 \
--scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits \
--min-capacity 1 \
--max-capacity 2
```

2. To verify the registration of the scalable target, use the following AWS CLI command:

```
aws application-autoscaling describe-scalable-targets \
--service-namespace comprehend \
--resource-id endpoint ARN
```

3. Create a target tracking configuration for the scaling policy and save the configuration in a file called config.json. The following is an example of a target tracking configuration that automatically adjusts the number of inference units so that utilized capacity is always 70% of the provisioned capacity.

```
{
 "TargetValue": 70,
 "PredefinedMetricSpecification":
 {
 "PredefinedMetricType": "ComprehendInferenceUtilization"
 }
}
```

4. Create a scaling policy. The following examples create a scaling policy based on the target tracking configuration defined in the config.json file.

For a document classification endpoint, use the following AWS CLI command:

```
aws application-autoscaling put-scaling-policy \
--service-namespace comprehend \
--resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
--scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
 \
--policy-name TestPolicy \
--policy-type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

For an entity recognizer endpoint, use the following AWS CLI command:

```
aws application-autoscaling put-scaling-policy \
--service-namespace comprehend \
--resource-id arn:aws:comprehend:region:account-id:entity-recognizer-endpoint/name
 \
--scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits \
--policy-name TestPolicy \
--policy-type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

## Removing Target Tracking

To remove target tracking for an endpoint, you use AWS CLI commands to delete the scaling policy and then deregister the scalable target.

## To remove target tracking

1. Delete the scaling policy. The following examples delete a specified scaling policy.

For a document classification endpoint, use the following AWS CLI command:

```
aws application-autoscaling delete-scaling-policy \
 --service-namespace comprehend \
 --resource-id arn:aws:comprehend:region:account-id:document-classifier-
 endpoint/name \
 --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
 \
 --policy-name TestPolicy \
```

For an entity recognizer endpoint, use the following AWS CLI command:

```
aws application-autoscaling delete-scaling-policy \
 --service-namespace comprehend \
 --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-endpoint/name
 \
 --scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits \
 --policy-name TestPolicy
```

2. Deregister the scalable target. The following examples deregister a specified scalable target.

For a document classification endpoint, use the following AWS CLI command:

```
aws application-autoscaling deregister-scalable-target \
 --service-namespace comprehend \
 --resource-id arn:aws:comprehend:region:account-id:document-classifier-
 endpoint/name \
 --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
```

For an entity recognizer endpoint, use the following AWS CLI command:

```
aws application-autoscaling deregister-scalable-target \
 --service-namespace comprehend \
 --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-endpoint/name
 \
 --scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits
```

## Scheduled Scaling

With scheduled scaling, you can adjust endpoint provisioning to fit your capacity needs on a specified schedule. Scheduled scaling automatically adjusts the number of inference units to accommodate surges of use at specific times. You can use scheduled scaling for document classification endpoints and entity recognizer endpoints. For additional information about scheduled scaling, see [Scheduled Scaling for Application Auto Scaling](#).

### Note

The following examples are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

## Setting up Scheduled Scaling

To set up scheduled scaling for an endpoint, you use AWS CLI commands to register a scalable target and then create a scheduled action. The scalable target defines inference units as the resource used to adjust endpoint provisioning, and the scheduled action controls the auto scaling of the provisioned capacity at specific times.

### To set up scheduled scaling

1. Register a scalable target. The following examples register a scalable target to adjust endpoint provisioning with a minimum capacity of 1 inference unit and a maximum capacity of 2 inference units.

For a document classification endpoint, use the following AWS CLI command:

```
aws application-autoscaling register-scalable-target \
 --service-namespace comprehend \
 --resource-id arn:aws:comprehend:region:account-id:document-classifier-
 endpoint/name \
 --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
 \
 --min-capacity 1 \
 --max-capacity 2
```

For an entity recognizer endpoint, use the following AWS CLI command:

```
aws application-autoscaling register-scalable-target \
 --service-namespace comprehend \
 --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-endpoint/name
 \
 --scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits \
 --min-capacity 1 \
 --max-capacity 2
```

2. Create a scheduled action. The following examples create a scheduled action to automatically adjust the provisioned capacity every day at 12:00 UTC with a minimum of 2 inference units and a maximum of 5 inference units. For more information about chronological expressions and scheduled scaling, see [Schedule Expressions](#).

For a document classification endpoint, use the following AWS CLI command:

```
aws application-autoscaling put-scheduled-action \
 --service-namespace comprehend \
 --resource-id arn:aws:comprehend:region:account-id:document-classifier-
 endpoint/name \
 --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
 \
 --scheduled-action-name TestScheduledAction \
 --schedule "cron(0 12 * * ? *)" \
 --scalable-target-action MinCapacity=2,MaxCapacity=5
```

For an entity recognizer endpoint, use the following AWS CLI command:

```
aws application-autoscaling put-scheduled-action \
 --service-namespace comprehend \
 --resource-id arn:aws:comprehend:region:account-id:entity-recognizer-endpoint/name
 \
 --scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits \
 --scheduled-action-name TestScheduledAction \
 --schedule "cron(0 12 * * ? *)" \
```

```
--scalable-target-action MinCapacity=2,MaxCapacity=5
```

## Removing Scheduled Scaling

To remove scheduled scaling for an endpoint, you use AWS CLI commands to delete the scheduled action and then deregister the scalable target.

### To remove scheduled scaling

1. Delete the scheduled action. The following examples delete a specified scheduled action.

For a document classification endpoint, use the following AWS CLI command:

```
aws application-autoscaling delete-scheduled-action \
--service-namespace comprehend \
--resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
--scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
\
--scheduled-action-name TestScheduledAction
```

For an entity recognizer endpoint, use the following AWS CLI command:

```
aws application-autoscaling delete-scheduled-action \
--service-namespace comprehend \
--resource-id arn:aws:comprehend:region:account-id:entity-recognizer-endpoint/name
\
--scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits \
--scheduled-action-name TestScheduledAction
```

2. Deregister the scalable target. The following examples deregister a specified scalable target.

For a document classification endpoint, use the following AWS CLI command:

```
aws application-autoscaling deregister-scalabe-target \
--service-namespace comprehend \
--resource-id arn:aws:comprehend:region:account-id:document-classifier-
endpoint/name \
--scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
```

For an entity recognizer endpoint, use the following AWS CLI command:

```
aws application-autoscaling deregister-scalabe-target \
--service-namespace comprehend \
--resource-id arn:aws:comprehend:region:account-id:entity-recognizer-endpoint/name
\
--scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits
```

# Security in Amazon Comprehend

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security of the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to Amazon Comprehend, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Comprehend. The following topics show you how to configure Amazon Comprehend to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Comprehend resources.

## Topics

- [Data Protection in Amazon Comprehend \(p. 182\)](#)
- [Authentication and Access Control for Amazon Comprehend \(p. 190\)](#)
- [Logging Amazon Comprehend API Calls with AWS CloudTrail \(p. 201\)](#)
- [Compliance Validation for Amazon Comprehend \(p. 210\)](#)
- [Resilience in Amazon Comprehend \(p. 210\)](#)
- [Infrastructure Security in Amazon Comprehend \(p. 210\)](#)
- [Permissions Required for a Custom Asynchronous Analysis Job \(p. 211\)](#)

## Data Protection in Amazon Comprehend

The AWS [shared responsibility model](#) applies to data protection in Amazon Comprehend. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the [AWS Security Blog](#).

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.

- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with Amazon Comprehend or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

#### Topics

- [KMS Encryption in Amazon Comprehend \(p. 183\)](#)
- [Protect Jobs by Using an Amazon Virtual Private Cloud \(p. 185\)](#)
- [Amazon Comprehend and interface VPC endpoints \(AWS PrivateLink\) \(p. 188\)](#)

## KMS Encryption in Amazon Comprehend

Amazon Comprehend works with AWS Key Management Service (AWS KMS) to provide enhanced encryption for your data. Amazon S3 already enables you to encrypt your input documents when creating a text analysis, topic modeling, or custom Amazon Comprehend job. Integration with AWS KMS enables you to encrypt the data in the storage volume for Start\* and Create\* jobs, and it encrypts the output results of Start\* jobs using your own KMS key.

For the AWS Management Console, Amazon Comprehend encrypts custom models with its own KMS key. For the AWS CLI, Amazon Comprehend can encrypt custom models using either its own KMS key or a provided customer managed key (CMK).

#### KMS encryption using the AWS Management Console

Two encryption options are available when using the console:

- Volume encryption
- Output result encryption

#### To enable volume encryption

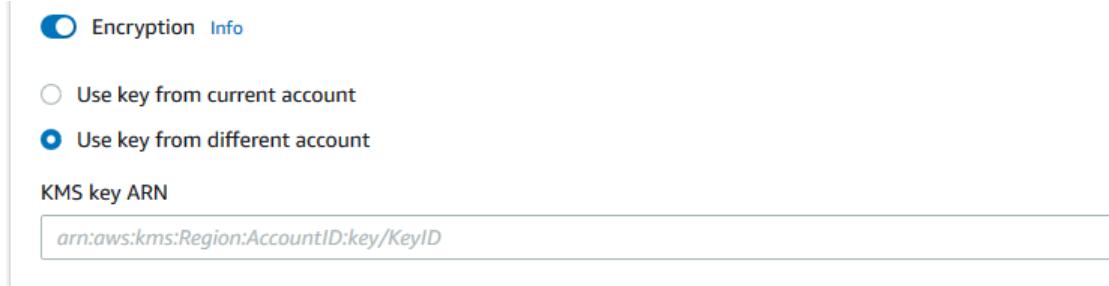
1. Under **Job Settings**, choose the **Job encryption** option.



2. Choose whether the KMS customer-managed key (CMK) is from the account you're currently using or from a different account. If you want to use a key from the current account, choose the key alias from **KMS key ID**. If you're using a key from a different account, you must enter the key's ARN.

## To enable output result encryption

- Under **Output Settings**, choose the **Encryption** option.



- Choose whether the customer-managed key (CMK) is from the account you're currently using or from a different account. If you want to use a key from the current account, choose the key ID from **KMS key ID**. If you're using a key from a different account, you must enter the key's ARN.

If you have previously setup encryption using SSE-KMS on the your S3 input documents, this can provide you with additional security. However, if you do this, the IAM role used must have `kms:Decrypt` permission for the KMS key with which the input documents are encrypted. For more information, see [Permissions Required to Use KMS Encryption \(p. 194\)](#).

### KMS encryption with API operations

All Amazon Comprehend Start\* and Create\* API operations support KMS encrypted input documents. Describe\* and List\* API operations return the `KmsKeyId` in `OutputDataConfig` if the original job had `KmsKeyId` provided as an input. If it was not provided as input, it isn't returned.

This can be seen in the following AWS CLI example using the [StartEntitiesDetectionJob \(p. 359\)](#) operation:

```
aws comprehend start-entities-detection-job \
--region region \
--data-access-role-arn "data access role arn" \
--entity-recognizer-arn "entity recognizer arn" \
--input-data-config "S3Uri=s3://Bucket Name/Bucket Path" \
--job-name job name \
--language-code en \
--output-data-config "KmsKeyId=Output S3 KMS key ID" "S3Uri=s3://Bucket Name/Bucket Path/" \
--volumekmskeyid "Volume KMS key ID"
```

#### Note

This example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

### Customer Managed Key (CMK) encryption with API operations

Amazon Comprehend custom model API operations, `CreateEntityRecognizer`, `CreateDocumentClassifier`, and `CreateEndpoint`, support encryption using customer managed keys via the AWS CLI.

You need an IAM policy to allow a principal to use or manage customer managed keys. These keys are specified in the `Resource` element of the policy statement. As best practice, limit customer managed keys to only those that the principals must use in your policy statement.

The following AWS CLI example creates a custom entity recognizer with model encryption using the [CreateEntityRecognizer \(p. 249\)](#) operation:

```
aws comprehend create-entity-recognizer \
 --recognizer-name name \
 --data-access-role-arn data access role arn \
 --language-code en \
 --model-kms-key-id Model KMS Key ID \
 --input-data-config file:///path/input-data-config.json
```

**Note**

This example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

## Protect Jobs by Using an Amazon Virtual Private Cloud

Amazon Comprehend uses a variety of security measures to ensure the safety of your data with our job containers where it's stored while being used by Amazon Comprehend. However, job containers access AWS resources—such as the Amazon S3 buckets where you store data and model artifacts—over the internet.

To control access to your data, we recommend that you create a *virtual private cloud* (VPC) and configure it so that the data and containers aren't accessible over the internet. For information about creating and configuring a VPC, see [Getting Started With Amazon VPC](#) in the *Amazon VPC User Guide*. Using a VPC helps to protect your data because you can configure your VPC so that it is not connected to the internet. Using a VPC also allows you to monitor all network traffic in and out of our job containers by using VPC flow logs. For more information, see [VPC Flow Logs](#) in the *Amazon VPC User Guide*.

You specify your VPC configuration when you create a job, by specifying the subnets and security groups. When you specify the subnets and security groups, Amazon Comprehend creates *elastic network interfaces* (ENIs) that are associated with your security groups in one of the subnets. ENIs allow our job containers to connect to resources in your VPC. For information about ENIs, see [Elastic Network Interfaces](#) in the *Amazon VPC User Guide*.

**Note**

For jobs, you can only configure subnets with a default tenancy VPC in which your instance runs on shared hardware. For more information on the tenancy attribute for VPCs, see [Dedicated Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

## Configure a Job for Amazon VPC Access

To specify subnets and security groups in your VPC, use the `VpcConfig` request parameter of the applicable API, or provide this information when you create a job in the Amazon Comprehend console. Amazon Comprehend uses this information to create ENIs and attach them to our job containers. The ENIs provide our job containers with a network connection within your VPC that is not connected to the internet.

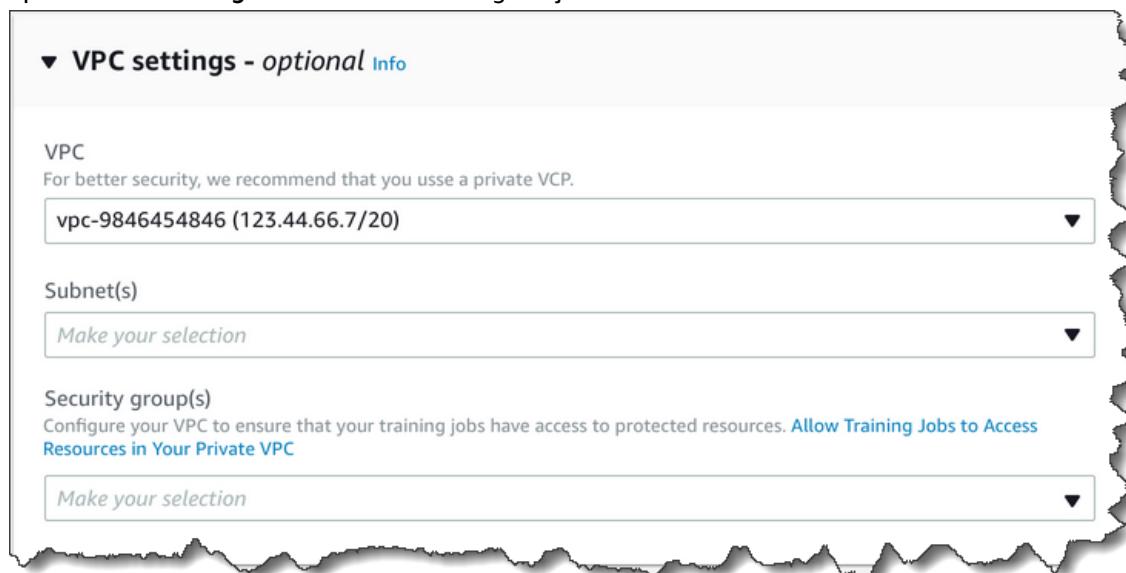
The following APIs contain the `VpcConfig` request parameter:

- Create\* APIs: [CreateDocumentClassifier](#) (p. 240), [CreateEntityRecognizer](#) (p. 249)
- Start\* APIs: [StartDocumentClassificationJob](#) (p. 349), [StartDominantLanguageDetectionJob](#) (p. 354), [StartEntitiesDetectionJob](#) (p. 359), [StartKeyPhrasesDetectionJob](#) (p. 368), [StartSentimentDetectionJob](#) (p. 377), [StartTopicsDetectionJob](#) (p. 382)

The following is an example of the `VpcConfig` parameter that you include in your API call:

```
"VpcConfig": {
 "SecurityGroupIds": [
 "sg-0123456789abcdef0"
],
 "Subnets": [
 "subnet-0123456789abcdef0",
 "subnet-0123456789abcdef1",
 "subnet-0123456789abcdef2"
]
}
```

To configure a VPC from the Amazon Comprehend console, choose the configuration details from the optional **VPC Settings** section when creating the job.



## Configure Your VPC for Amazon Comprehend Job

When configuring the VPC for your Amazon Comprehend jobs, use the following guidelines. For information about setting up a VPC, see [Working with VPCs and Subnets](#) in the *Amazon VPC User Guide*.

### Ensure That Subnets Have Enough IP Addresses

Your VPC subnets should have at least two private IP addresses for each instance in a job. For more information, see [VPC and Subnet Sizing for IPv4](#) in the *Amazon VPC User Guide*.

### Create an Amazon S3 VPC Endpoint

If you configure your VPC so that job containers don't have access to the internet, they can't connect to the Amazon S3 buckets that contain your data unless you create a VPC endpoint that allows access. By creating a VPC endpoint, you allow our job containers to access the model artifacts and your data. We recommend that you also create a custom policy that allows only requests from your VPC to access to your S3 buckets. For more information, see [Endpoints for Amazon S3](#) in the *Amazon VPC User Guide*.

The following policy allows access to S3 buckets. Edit this policy to allow access only the resources that your job needs.

```
{
 "Version": "2008-10-17",
```

```

 "Statement": [
 {
 "Effect": "Allow",
 "Principal": "*",
 "Action": [
 "s3:GetObject",
 "s3:PutObject",
 "s3>ListBucket",
 "s3:GetBucketLocation",
 "s3>DeleteObject",
 "s3>ListMultipartUploadParts",
 "s3:AbortMultipartUpload"
],
 "Resource": "*"
 }
]
}

```

Use default DNS settings for your endpoint route table, so that standard Amazon S3 URLs (for example, `http://s3-aws-region.amazonaws.com/MyBucket`) resolve. If you don't use default DNS settings, ensure that the URLs that you use to specify the locations of the data in your jobs resolve by configuring the endpoint route tables. For information about VPC endpoint route tables, see [Routing for Gateway Endpoints](#) in the *Amazon VPC User Guide*.

The default endpoint policy allows users to install packages from the Amazon Linux and Amazon Linux 2 repositories on our jobs container. If you don't want users to install packages from that repository, create a custom endpoint policy that explicitly denies access to the Amazon Linux and Amazon Linux 2 repositories. Comprehend itself doesn't need any such packages, so there won't be any functionality impact. The following is an example of a policy that denies access to these repositories:

```

{
 "Statement": [
 {
 "Sid": "AmazonLinuxAMIRRepositoryAccess",
 "Principal": "*",
 "Action": [
 "s3:GetObject"
],
 "Effect": "Deny",
 "Resource": [
 "arn:aws:s3:::packages.*.amazonaws.com/*",
 "arn:aws:s3:::repo.*.amazonaws.com/*"
]
 }
]
}

{
 "Statement": [
 {
 "Sid": "AmazonLinux2AMIRRepositoryAccess",
 "Principal": "*",
 "Action": [
 "s3:GetObject"
],
 "Effect": "Deny",
 "Resource": [
 "arn:aws:s3:::amazonlinux.*.amazonaws.com/*"
]
 }
]
}

```

#### Permissions for the `DataAccessRole`

When you use a VPC with your analysis job, the `DataAccessRole` used for the `Create*` and `Start*` operations must also have permissions to the VPC from which the input documents and the output bucket are accessed.

The following policy provides the access needed to the `DataAccessRole` used for the `Create*` and `Start*` operations.

```
{
 "Version": "2008-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ec2:CreateNetworkInterface",
 "ec2:CreateNetworkInterfacePermission",
 "ec2:DeleteNetworkInterface",
 "ec2:DeleteNetworkInterfacePermission",
 "ec2:DescribeNetworkInterfaces",
 "ec2:DescribeVpcs",
 "ec2:DescribeDhcpOptions",
 "ec2:DescribeSubnets",
 "ec2:DescribeSecurityGroups"
],
 "Resource": "*"
 }
]
}
```

### Configure the VPC Security Group

With distributed jobs, you must allow communication between the different job containers in the same job. To do that, configure a rule for your security group that allows inbound connections between members of the same security group. For information, see [Security Group Rules](#) in the *Amazon VPC User Guide*.

### Connect to Resources Outside Your VPC

If you configure your VPC so that it doesn't have internet access, jobs that use that VPC do not have access to resources outside your VPC. If your jobs need access to resources outside your VPC, provide access with one of the following options:

- If your job needs access to an AWS service that supports interface VPC endpoints, create an endpoint to connect to that service. For a list of services that support interface endpoints, see [VPC Endpoints](#) in the *Amazon VPC User Guide*. For information about creating an interface VPC endpoint, see [Interface VPC Endpoints \(AWS PrivateLink\)](#) in the *Amazon VPC User Guide*.
- If your job needs access to an AWS service that doesn't support interface VPC endpoints or to a resource outside of AWS, create a NAT gateway and configure your security groups to allow outbound connections. For information about setting up a NAT gateway for your VPC, see [Scenario 2: VPC with Public and Private Subnets \(NAT\)](#) in the *Amazon VPC User Guide*.

## Amazon Comprehend and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and Amazon Comprehend by creating an *interface VPC endpoint*. Interface endpoints are powered by [AWS PrivateLink](#), a technology that enables you to privately access Amazon Comprehend APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to

communicate with Amazon Comprehend APIs. Traffic between your VPC and Amazon Comprehend does not leave the Amazon network.

Each interface endpoint is represented by one or more [Elastic Network Interfaces](#) in your subnets.

For more information, see [Interface VPC endpoints \(AWS PrivateLink\)](#) in the *Amazon VPC User Guide*.

## Considerations for Amazon Comprehend VPC endpoints

Before you set up an interface VPC endpoint for Amazon Comprehend, ensure that you review [Interface endpoint properties and limitations](#) in the *Amazon VPC User Guide*.

Amazon Comprehend supports making calls to all of its API actions from your VPC.

## Creating an interface VPC endpoint for Amazon Comprehend

You can create a VPC endpoint for the Amazon Comprehend service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see [Creating an interface endpoint](#) in the *Amazon VPC User Guide*.

Create a VPC endpoint for Amazon Comprehend using the following service name:

- com.amazonaws.*region*.comprehend

If you enable private DNS for the endpoint, you can make API requests to Amazon Comprehend using its default DNS name for the Region, for example, *comprehend.us-east-1.amazonaws.com*.

For more information, see [Accessing a service through an interface endpoint](#) in the *Amazon VPC User Guide*.

## Creating a VPC endpoint policy for Amazon Comprehend

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon Comprehend. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see [Controlling access to services with VPC endpoints](#) in the *Amazon VPC User Guide*.

### Example: VPC endpoint policy for Amazon Comprehend actions

The following is an example of an endpoint policy for Amazon Comprehend. When attached to an endpoint, this policy grants access to the Amazon Comprehend `DetectEntities` action for all principals on all resources.

```
{
 "Statement": [
 {
 "Principal": "*",
 "Effect": "Allow",
 "Action": [
 "comprehend:DetectEntities"
],
 "Resource": "*"
 }
]
}
```

]  
}

# Authentication and Access Control for Amazon Comprehend

Access to Amazon Comprehend requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access Amazon Comprehend actions. The following sections provide details on how you can use [AWS Identity and Access Management \(IAM\)](#) and Amazon Comprehend to help secure your resources by controlling who can access them.

- [Authentication \(p. 190\)](#)
- [Access Control \(p. 191\)](#)

## Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.
- **IAM user** – An [IAM user](#) is an identity within your AWS account that has specific custom permissions (for example, permissions to create in Amazon Comprehend). You can use an IAM user name and password to sign in to secure AWS webpages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(CLI\)](#). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. Amazon Comprehend supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 signing process](#) in the [AWS General Reference](#).

- **IAM role** – An [IAM role](#) is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. IAM roles with temporary credentials are useful in the following situations:
  - **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as

federated users. AWS assigns a role to a federated user when access is requested through an [identity provider](#). For more information about federated users, see [Federated users and roles](#) in the *IAM User Guide*.

- **AWS service access** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

## Access Control

You must have valid credentials to authenticate your requests. The credentials must have permissions to call an Amazon Comprehend action.

The following sections describe how to manage permissions for Amazon Comprehend. We recommend that you read the overview first.

- [Overview of Managing Access Permissions to Amazon Comprehend Resources \(p. 191\)](#)
- [Using Identity-Based Policies \(IAM Policies\) for Amazon Comprehend \(p. 193\)](#)

## Overview of Managing Access Permissions to Amazon Comprehend Resources

Permissions to access an action are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) to manage access to actions.

### Note

An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions and the actions they get permissions for.

### Topics

- [Managing Access to Actions \(p. 191\)](#)
- [Specifying Policy Elements: Actions, Effects, and Principals \(p. 192\)](#)
- [Specifying Conditions in a Policy \(p. 193\)](#)

## Managing Access to Actions

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

### Note

This section discusses using IAM in the context of Amazon Comprehend. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What Is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM policies) and policies attached to a resource are referred to as *resource-based* policies. Amazon Comprehend supports only identity-based policies.

## Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user or a group of users permissions to call an Amazon Comprehend action, you can attach a permissions policy to a user or group that the user belongs to.
- **Attach a permissions policy to a role (grant cross-account permissions)** – To grant cross-account permissions, you can attach an identity-based permissions policy to an IAM role. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
  2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. If you want to grant an AWS service permission to assume the role, the principal in the trust policy can also be an AWS service principal.

For more information about using IAM to delegate permissions, see [Access Management](#) in the *IAM User Guide*.

For more information about using identity-based policies with Amazon Comprehend, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Comprehend \(p. 193\)](#). For more information about users, groups, roles, and permissions, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

## Resource-Based Policies

Other services, such as Lambda, support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. Amazon Comprehend doesn't support resource-based policies.

## Specifying Policy Elements: Actions, Effects, and Principals

Amazon Comprehend defines a set of API operations (see [Actions \(p. 218\)](#)). To grant permissions for these API operations, Amazon Comprehend defines a set of actions that you can specify in a policy.

The following are the most basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For Amazon Comprehend, the resource is always "\*".
- **Action** – You use action keywords to identify operations that you want to allow or deny. For example, depending on the specified Effect, `comprehend:DetectEntities` either allows or denies the user permissions to perform the Amazon Comprehend `DetectEntities` operation.

- **Effect** – You specify the effect of the action that occurs when the user requests the specific action —this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure that a user cannot access the resource, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal.

To learn more about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

For a table showing all of the Amazon Comprehend API actions, see [Amazon Comprehend API Permissions: Actions, Resources, and Conditions Reference \(p. 198\)](#).

## Specifying Conditions in a Policy

When you grant permissions, you use the IAM policy language to specify the conditions under which a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see [Condition](#) in the *IAM User Guide*.

AWS provides a set of predefined condition keys for all AWS services that support IAM for access control. For example, you can use the `aws:userid` condition key to require a specific AWS ID when requesting an action. For more information and a complete list of AWS-wide keys, see [Available Keys for Conditions](#) in the *IAM User Guide*.

### Note

Condition keys are case-sensitive.

Amazon Comprehend does not provide any additional condition keys.

## Using Identity-Based Policies (IAM Policies) for Amazon Comprehend

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) and thereby grant permissions to perform Amazon Comprehend actions.

### Important

Before you proceed, we recommend that you review [Overview of Managing Access Permissions to Amazon Comprehend Resources \(p. 191\)](#).

The following is the permissions policy required to use the Amazon Comprehend document analysis actions:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {"Sid": "AllowDetectActions",
 "Effect": "Allow",
 "Action": [
 "comprehend:DetectEntities",
 "comprehend:DetectKeyPhrases",
 "comprehend:DetectDominantLanguage",
 "comprehend:DetectSentiment",
 "comprehend:DetectSyntax"
],
 "Resource": "*"
```

```
]
 }
}
```

The policy has one statement that grants permission to use the `DetectEntities`, `DetectKeyPhrases`, `DetectDominantLanguage` and `DetectSentiment`, and `DetectSyntax` actions. A user with this policy would not be able to perform batch actions or asynchronous actions in your account.

The policy doesn't specify the `Principal` element because you don't specify the principal who gets the permission in an identity-based policy. When you attach a policy to a user, the user is the implicit principal. When you attach a permissions policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the Amazon Comprehend API actions and the resources that they apply to, see [Amazon Comprehend API Permissions: Actions, Resources, and Conditions Reference \(p. 198\)](#).

## Permissions Required to Use KMS Encryption

The permissions reference table lists the Amazon Comprehend API operations and shows the required permissions for each operation. For more information about Amazon Comprehend API permissions, see [Amazon Comprehend API Permissions: Actions, Resources, and Conditions Reference \(p. 198\)](#).

To fully use Amazon Key Management Service (KMS) for data and job encryption in an asynchronous job, you need to grant permissions for the actions shown in the following policy:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "kms:CreateGrant",
 "kms:Decrypt",
 "kms:GenerateDatakey"
],
 "Effect": "Allow",
 "Resource": "*"
 }
]
}
```

When you create an asynchronous job with Amazon Comprehend you use input data stored on Amazon S3. With S3, you have the option to encrypt your stored data, which is encrypted by S3, not by Amazon Comprehend. We can decrypt and read that encrypted input data if you provide `kms:Decrypt` permission for the key with which the original input data was encrypted to the data access role used by the Amazon Comprehend job.

You also have the option of using KMS customer-managed keys (CMK) to encrypt the output results on S3, as well as the storage volume used during job processing. When you do this, you can use the same KMS key for both types of encryption, but this is not necessary. Separate fields are available when creating the job to specify the keys for output encryption and volume encryption and you can even use a KMS key from a different account.

When using KMS encryption, `kms:CreateGrant` permission is required for volume encryption and `kms:GenerateDataKey` permission is needed for output data encryption. For reading encrypted input (as when the input data is already encrypted by Amazon S3), `kms:Decrypt` permission is required. The IAM role needs to give these permissions as needed. However, if the key is from a different account than is currently being used, the KMS key policy for that kms key must also give these permissions to the data access role for the job.

## Permissions Required to Use the Amazon Comprehend Console

The permissions reference table lists the Amazon Comprehend API operations and shows the required permissions for each operation. For more information about Amazon Comprehend API permissions, see [Amazon Comprehend API Permissions: Actions, Resources, and Conditions Reference \(p. 198\)](#).

To use the Amazon Comprehend console, you need to grant permissions for the actions shown in the following policy:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "comprehend:*",
 "iam>ListRoles",
 "iam:GetRole",
 "s3>ListAllMyBuckets",
 "s3>ListBucket",
 "s3:GetBucketLocation"
],
 "Effect": "Allow",
 "Resource": "*"
 }
]
}
```

The Amazon Comprehend console needs these additional permissions for the following reasons:

- `iam` permissions to list the available IAM roles for your account.
- `s3` permissions to access the Amazon S3 buckets and objects that contain the data for topic modeling.

When you create an asynchronous batch job or a topic modeling job using the console, you have the option to have the console create an IAM role for your job. To create an IAM role, users must be granted the following additional permissions to create IAM roles and policies, and to attach policies to roles:

```
{
 "Version": "2012-10-17",
 "Statement": [
 [
 {
 "Action": [
 [
 "iam>CreateRole",
 "iam>CreatePolicy",
 "iam:AttachRolePolicy"
],
 "Effect": "Allow",
 "Resource": "*"
 },
 {
 "Action": [
 [
 "iam:PassRole"
],
 "Effect": "Allow",
 "Resource": "arn:aws:iam::*:role/*Comprehend*"
 }
 }
]
]
]
}
```

The Amazon Comprehend console needs these additional permissions for the following reasons:

- `iam` permissions to create roles and policies and to attach roles and policies. The `iam:PassRole` action enables the console to pass the role to Amazon Comprehend.

## AWS Managed (Predefined) Policies for Amazon Comprehend

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so that you can avoid having to investigate what permissions are needed. For more information, see [AWS Managed Policies](#) in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon Comprehend:

- **ComprehendFullAccess** – Grants full access to Amazon Comprehend resources including running topic modeling jobs. Includes permission to list and get IAM roles.
- **ComprehendReadOnly** – Grants permission to run all Amazon Comprehend actions except `StartDominantLanguageDetectionJob`, `StartEntitiesDetectionJob`, `StartKeyPhrasesDetectionJob`, `StartSentimentDetectionJob`, and `StartTopicsDetectionJob`.

You need to apply the following additional policy to any user that will use Amazon Comprehend:

```
{
 "Version": "2012-10-17",
 "Statement":
 [
 {
 "Action":
 [
 "iam:PassRole"
],
 "Effect": "Allow",
 "Resource": "arn:aws:iam::*:role/*Comprehend*"
 }
]
}
```

You can review the managed permissions policies by signing in to the IAM console and searching for specific policies there.

These policies work when you are using AWS SDKs or the AWS CLI.

You can also create your own custom IAM policies to allow permissions for Amazon Comprehend actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

## Role-Based Permissions Required for Asynchronous Operations

To use the Amazon Comprehend asynchronous operations, you must grant Amazon Comprehend access to the Amazon S3 bucket that contains your document collection. You do this by creating a data access role in your account to trust the Amazon Comprehend service principal. For more information about creating a role, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *AWS Identity and Access Management User Guide*.

The following is the role's trust policy:

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

After you have created the role, you must create an access policy for that role. This should grant the Amazon S3 `GetObject` and `ListBucket` permissions to the Amazon S3 bucket that contains your input data, and the Amazon S3 `PutObject` permission to your Amazon S3 output data bucket.

The following example access policy contains those permissions.

## Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various Amazon Comprehend actions. These policies work when you are using AWS SDKs or the AWS CLI. When you are using the console, you need to grant permissions to all the Amazon Comprehend APIs. This is discussed in [Permissions Required to Use the Amazon Comprehend Console \(p. 195\)](#).

## Note

All examples use the us-east-2 region and contain fictitious account IDs.

## Examples

## Example 1: Allow All Amazon Comprehend Actions

After you sign up for AWS, you create an administrator user to manage your account, including creating users and managing their permissions.

You might choose to create a user who has permissions for all Amazon Comprehend actions (think of this user as a service-specific administrator) for working with Amazon Comprehend. You can attach the following permissions policy to this user.

```
{
 "Version": "2012-10-17",
 "Statement":
 [
 {
 "Sid": "AllowAllComprehendActions",
 "Effect": "Allow",
 "Action":
 [
 "comprehend:*",
 "iam>ListRoles",
 "iam:GetRole",
 "s3>ListAllMyBuckets",
 "s3>ListBucket",
 "s3>GetBucketLocation",
 "iam>CreateRole",
 "iam>CreatePolicy",
 "iam>AttachRolePolicy",
 "kms>CreateGrant",
 "kms>Decrypt".
]
 }
]
}
```

```

 "kms:GenerateDatakey"
],
 "Resource": "*"
},
{
 "Action":
 [
 "iam:PassRole"
],
 "Effect": "Allow",
 "Resource": "arn:aws:iam::*:role/*Comprehend*"
}
]
}

```

These permissions can be modified with regard to encryption in the following ways:

- To enable Amazon Comprehend to analyze documents stored in an encrypted S3 bucket, the IAM role must have the `kms:Decrypt` permission.
- To enable Amazon Comprehend to encrypt documents stored on a storage volume attached to the compute instance that processes the analysis job, the IAM role must have the `kms>CreateGrant` permission.
- To enable Amazon Comprehend to encrypt the output results in their S3 bucket, the IAM role must have the `kms:GenerateDataKey` permission.

## Example 2: Allow Topic Modeling Actions

The following permissions policy grants user permissions to perform the Amazon Comprehend topic modeling operations.

```

{
 "Version": "2012-10-17",
 "Statement": [{
 "Sid": "AllowTopicModelingActions",
 "Effect": "Allow",
 "Action": [
 "comprehend:DescribeTopicsDetectionJob",
 "comprehend>ListTopicsDetectionJobs",
 "comprehend:StartTopicsDetectionJob",
],
 "Resource": "*"
 }]
}

```

## Amazon Comprehend API Permissions: Actions, Resources, and Conditions Reference

Use the following table as a reference when setting up [Access Control \(p. 191\)](#) and writing a permissions policy that you can attach to an IAM identity (an identity-based policy). The list includes each Amazon Comprehend API operation, the corresponding action for which you can grant permissions to perform the action, and the AWS resource for which you can grant the permissions. You specify the actions in the policy's Action field, and you specify the resource value in the policy's Resource field.

To express conditions, you can use AWS-wide condition keys in your Amazon Comprehend policies. For a complete list of AWS-wide keys, see [Available Keys](#) in the *IAM User Guide*.

**Note**

To specify an action, use the `comprehend:` prefix followed by the API operation name, for example, `comprehend:DetectEntities`.

## AWS Managed Policies for Amazon Comprehend

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to [create IAM customer managed policies](#) that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see [AWS managed policies](#) in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the `ReadOnlyAccess` AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see [AWS managed policies for job functions](#) in the *IAM User Guide*.

### AWS managed policy: ComprehendFullAccess

This policy grants full access to Amazon Comprehend resources including running topic modeling jobs. This policy also grants list and get permissions for Amazon S3 buckets and IAM roles.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "comprehend:*",
 "s3>ListAllMyBuckets",
 "s3>ListBucket",
 "s3:GetBucketLocation",
 "iam>ListRoles",
 "iam:GetRole"
],
 "Effect": "Allow",
 "Resource": "*"
 }
]
}
```

### AWS managed policy: ComprehendReadOnly

This policy grants read-only permissions to run all Amazon Comprehend actions **except** the following:

- `StartDominantLanguageDetectionJob`

- StartEntitiesDetectionJob
- StartKeyPhrasesDetectionJob
- StartSentimentDetectionJob
- StartTopicsDetectionJob

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "comprehend:DetectDominantLanguage",
 "comprehend:BatchDetectDominantLanguage",
 "comprehend:DetectEntities",
 "comprehend:BatchDetectEntities",
 "comprehend:DetectKeyPhrases",
 "comprehend:BatchDetectKeyPhrases",
 "comprehend:DetectPiiEntities",
 "comprehend:DetectSentiment",
 "comprehend:BatchDetectSentiment",
 "comprehend:DetectSyntax",
 "comprehend:BatchDetectSyntax",
 "comprehend:ClassifyDocument",
 "comprehend:ContainsPiiEntities",
 "comprehend:DescribeTopicsDetectionJob",
 "comprehend>ListTopicsDetectionJobs",
 "comprehend:DescribeDominantLanguageDetectionJob",
 "comprehend>ListDominantLanguageDetectionJobs",
 "comprehend:DescribeEntitiesDetectionJob",
 "comprehend>ListEntitiesDetectionJobs",
 "comprehend:DescribeKeyPhrasesDetectionJob",
 "comprehend>ListKeyPhrasesDetectionJobs",
 "comprehend:DescribePiiEntitiesDetectionJob",
 "comprehend>ListPiiEntitiesDetectionJobs",
 "comprehend:DescribeSentimentDetectionJob",
 "comprehend>ListSentimentDetectionJobs",
 "comprehend:DescribeDocumentClassifier",
 "comprehend>ListDocumentClassifiers",
 "comprehend>ListDocumentClassifierSummaries",
 "comprehend:DescribeDocumentClassificationJob",
 "comprehend>ListDocumentClassificationJobs",
 "comprehend:DescribeEntityRecognizer",
 "comprehend>ListEntityRecognizers",
 "comprehend>ListEntityRecognizerSummaries",
 "comprehend>ListTagsForResource",
 "comprehend:DescribeEndpoint",
 "comprehend>ListEndpoints"
],
 "Effect": "Allow",
 "Resource": "*"
 }
]
}
```

## Amazon Comprehend updates to AWS managed policies

View details about updates to AWS managed policies for Amazon Comprehend since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon Comprehend [Document history](#) page.

| Change                                                                        | Description                                                                                                                                                                                            | Date               |
|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| <a href="#">ComprehendReadOnly (p. 199)</a><br>– Update to an existing policy | Amazon Comprehend added new permissions to allow the <code>ListDocumentClassifierSummaries</code> and <code>ListEntityRecognizerSummaries</code> actions to the <code>ComprehendReadOnly</code> policy | September 21, 2021 |
| <a href="#">ComprehendReadOnly (p. 199)</a><br>– Update to an existing policy | Amazon Comprehend added new permissions to allow the <code>ContainsPIIEntities</code> action to the <code>ComprehendReadOnly</code> policy                                                             | March 26, 2021     |
| Amazon Comprehend started tracking changes                                    | Amazon Comprehend started tracking changes for its AWS managed policies.                                                                                                                               | March 1, 2021      |

## Logging Amazon Comprehend API Calls with AWS CloudTrail

Amazon Comprehend is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Comprehend. CloudTrail captures API calls for Amazon Comprehend as events. The calls captured include calls from the Amazon Comprehend console and code calls to the Amazon Comprehend API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Comprehend. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in [Event history](#). Using the information collected by CloudTrail, you can determine the request that was made to Amazon Comprehend, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#).

### Amazon Comprehend Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in Amazon Comprehend, that activity is recorded in a CloudTrail event along with other AWS service events in [Event history](#). You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

For an ongoing record of events in your AWS account, including events for Amazon Comprehend, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)

- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Amazon Comprehend supports logging the following actions as events in CloudTrail log files:

- BatchDetectDominantLanguage
- BatchDetectEntities
- BatchDetectKeyPhrases
- BatchDetectSentiment
- BatchDetectSyntax
- ClassifyDocument
- CreateDocumentClassifier
- CreateEndpoint
- CreateEntityRecognizer
- DeleteDocumentClassifier
- DeleteEndpoint
- DeleteEntityRecognizer
- DescribeDocumentClassificationJob
- DescribeDocumentClassifier
- DescribeDominantLanguageDetectionJob
- DescribeEndpoint
- DescribeEntitiesDetectionJob
- DescribeEntityRecognizer
- DescribeKeyPhrasesDetectionJob
- DescribePiiEntitiesDetectionJob
- DescribeSentimentDetectionJob
- DescribeTopicsDetectionJob
- DetectDominantLanguage
- DetectEntities
- DetectKeyPhrases
- DetectPiiEntities
- DetectSentiment
- DetectSyntax
- ListDocumentClassificationJobs
- ListDocumentClassifiers
- ListDominantLanguageDetectionJobs
- ListEndpoints
- ListEntitiesDetectionJobs
- ListEntityRecognizers
- ListKeyPhrasesDetectionJobs
- ListPiiEntitiesDetectionJobs
- ListSentimentDetectionJobs
- ListTagsForResource
- ListTopicsDetectionJobs
- StartDocumentClassificationJob
- StartDominantLanguageDetectionJob

- [StartEntitiesDetectionJob](#)
- [StartKeyPhrasesDetectionJob](#)
- [StartPiiEntitiesDetectionJob](#)
- [StartSentimentDetectionJob](#)
- [StartTopicsDetectionJob](#)
- [StopDominantLanguageDetectionJob](#)
- [StopEntitiesDetectionJob](#)
- [StopKeyPhrasesDetectionJob](#)
- [StopPiiEntitiesDetectionJob](#)
- [StopSentimentDetectionJob](#)
- [StopTrainingDocumentClassifier](#)
- [StopTrainingEntityRecognizer](#)
- [TagResource](#)
- [UntagResource](#)
- [UpdateEndpoint](#)

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity Element](#).

## Examples: Amazon Comprehend Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the [ListKeyPhrasesDetectionJobs](#) action.

```
{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "AROAICFHPEXAMPLE:kapil",
 "arn": "arn:aws:sts::12345678910:assumed-role/UserRole/kapil",
 "accountId": "12345678910",
 "accessKeyId": "ASIA3VZEXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AROAICFHPEXAMPLE",
 "arn": "arn:aws:iam::12345678910:role/UserRole",
 "accountId": "12345678910",
 "userName": "UserRole"
 },
 }
 },
}
```

```

 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-29T15:46:04Z"
 }
 },
 "eventTime": "2020-04-29T17:57:06Z",
 "eventSource": "comprehend.amazonaws.com",
 "eventName": "ListKeyPhrasesDetectionJobs",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "3.22.248.29",
 "userAgent": "aws-internal/3 aws-sdk-java/1.11.761 Linux/4.14.165-102.205.amzn2.x86_64
OpenJDK_64-Bit_Server_VM/25.201-b09 java/1.8.0_201 vendor/Oracle_Corporation, canary-
generated exec-env/AWS_Lambda_java8",
 "requestParameters": {
 "filter": {
 "submitTimeAfter": "Apr 29, 2020 5:54:04 PM"
 }
 },
 "responseElements": null,
 "requestID": "59bee2bc-e45c-436e-aae7-493af0328a8c",
 "eventId": "eabb7b59-edf3-44c8-8833-416bbdb16eae",
 "readOnly": true,
 "eventType": "AwsApiCall",
 "recipientAccountId": "802699264198"
}

```

The following example shows a CloudTrail log entry that demonstrates the StartKeyPhrasesDetectionJob action.

```

{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "AROAICFHPEXAMPLE:kapil",
 "arn": "arn:aws:sts::12345678910:assumed-role/UserRole/kapil",
 "accountId": "12345678910",
 "accessKeyId": "ASIA3VZEXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AROAICFHPEXAMPLE",
 "arn": "arn:aws:iam::12345678910:role/UserRole",
 "accountId": "12345678910",
 "userName": "UserRole"
 },
 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-29T15:46:04Z"
 }
 }
 },
 "eventTime": "2020-04-29T16:42:39Z",
 "eventSource": "comprehend.amazonaws.com",
 "eventName": "StartKeyPhrasesDetectionJob",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "3.20.236.234",
 "userAgent": "aws-internal/3 aws-sdk-java/1.11.761 Linux/4.14.165-102.205.amzn2.x86_64
OpenJDK_64-Bit_Server_VM/25.201-b09 java/1.8.0_201 vendor/Oracle_Corporation, canary-
generated exec-env/AWS_Lambda_java8",
 "requestParameters": {
 "inputDataConfig": {

```

```

 "s3Uri": "s3://dataset-prod-us-east-2/ONE_DOC_PER_LINE/KP/FOUR_KB",
 "inputFormat": "ONE_DOC_PER_LINE"
 },
 "outputDataConfig": {
 "s3Uri": "s3://datasets3bucke-y6icltvagurj/JSON/00f706b3-6d84-4b09-
bea3-88084f9cb6b0",
 "kmsKeyId": "arn:aws:kms:us-
east-2:12345678910:key/2e107fe3-6ab3-4bab-90ab-966e85b14678"
 },
 "dataAccessRoleArn": "arn:aws:iam::12345678910:role/DataAccessRoleXYZ",
 "languageCode": "en",
 "clientRequestToken": "d9e4f777-8b4e-4249-8ebb-b7495acb800b",
 "volumeKmsKeyId": "arn:aws:kms:us-east-2:12345678910:key/7b9b8aff-c459-4d20-
bb39-64d6e3756e85"
},
"responseElements": {
 "jobId": "2102905798f0d44a04da91fbd12ed60e",
 "jobStatus": "SUBMITTED"
},
"requestID": "f202914c-f5cf-4f2d-a4c7-5f30d101fe19",
"eventID": "692e3a17-40a2-491a-a4ce-a275c4c4ce76",
"readOnly": false,
"resources": [
{
 "accountId": "12345678910",
 "type": "AWS::KMS::Key",
 "ARN": "arn:aws:kms:us-
east-2:12345678910:key/2e107fe3-6ab3-4bab-90ab-966e85b14678"
},
{
 "accountId": "12345678910",
 "type": "AWS::KMS::Key",
 "ARN": "arn:aws:kms:us-east-2:12345678910:key/7b9b8aff-c459-4d20-
bb39-64d6e3756e85"
},
{
 "accountId": "12345678910",
 "type": "AWS::S3::Object",
 "ARN": "s3://dataset-prod-us-east-2/ONE_DOC_PER_LINE/KP/FOUR_KB"
},
{
 "accountId": "12345678910",
 "type": "AWS::S3::Object",
 "ARN": "s3://datasets3bucke-y6icltvagurj/JSON/00f706b3-6d84-4b09-
bea3-88084f9cb6b0"
},
{
 "accountId": "12345678910",
 "type": "AWS::IAM::Role",
 "ARN": "arn:aws:iam::12345678910:role/DataAccessRoleXYZ"
}
],
"eventType": "AwsApiCall",
"recipientAccountId": "12345678910"
}

```

The following example shows a CloudTrail log entry that demonstrates the `DescribeDocumentClassifier` action.

```
{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",

```

```

 "principalId": "AROAICFHPEXAMPLE:SomeServiceCa-CheckAsyncJobStatusLambd-
TICR23C65S42",
 "arn": "arn:aws:sts::12345678910:assumed-role/SomeServiceCa-SomeLambdaR-
ADLIAGB0VT9X/SomeServiceCa-CheckAsyncJobStatusLambd-TICR23C65S42",
 "accountId": "12345678910",
 "accessKeyId": "ASIA3VZEXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AROAICFHPEXAMPLE",
 "arn": "arn:aws:iam::12345678910:role/SomeServiceCa-SomeLambdaR-
ADLIAGB0VT9X",
 "accountId": "12345678910",
 "userName": "SomeServiceCa-SomeLambdaR-ADLIAGB0VT9X"
 },
 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-29T15:30:41Z"
 }
 }
 },
 "eventTime": "2020-04-29T17:02:44Z",
 "eventSource": "comprehend.amazonaws.com",
 "eventName": "DescribeDocumentClassifier",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "18.191.40.153",
 "userAgent": "aws-internal/3 aws-sdk-java/1.11.761 Linux/4.14.165-102.205.amzn2.x86_64
OpenJDK_64-Bit_Server_VM/25.201-b09 java/1.8.0_201 vendor/Oracle_Corporation, canary-
generated exec-env/AWS_Lambda_java8",
 "errorCode": "ResourceNotFoundException",
 "errorMessage": "RESOURCE_NOT_FOUND: Could not find specified resource.",
 "requestParameters": {
 "documentClassifierArn": "arn:aws:comprehend:us-east-2:12345678910:document-
classifier/DocumentClassifier-FOUR-KB-MULTI-LABEL-1588162272420"
 },
 "responseElements": null,
 "requestID": "e5bf2476-a894-4204-af77-4a84857ff2f8",
 "eventID": "bc77875f-9fe5-4b65-8384-c9f82c392f46",
 "readOnly": true,
 "resources": [
 {
 "accountId": "12345678910",
 "type": "AWS::Comprehend::DocumentClassifier",
 "ARN": "arn:aws:comprehend:us-east-2:12345678910:document-classifier/
DocumentClassifier-FOUR-KB-MULTI-LABEL-1588162272420"
 }
],
 "eventType": "AwsApiCall",
 "recipientAccountId": "12345678910"
}

```

The following example shows a CloudTrail log entry that demonstrates the `CreateDocumentClassifier` action.

```
{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "AROAICFHPEXAMPLE:SomeServiceCanary-
StartAsyncJobLambda-1STGUG0X870YM",
 "arn": "arn:aws:sts::12345678910:assumed-role/SomeServiceCa-SomeLambdaR-
ADLIAGB0VT9X/SomeServiceCanary-StartAsyncJobLambda-1STGUG0X870YM",
 "accountId": "12345678910",

```

```

 "accessKeyId": "ASIA3VZEXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AROAICFHPEXAMPLE",
 "arn": "arn:aws:iam::12345678910:role/SomeServiceCa-SomeLambdaR-
ADLIAGB0VT9X",
 "accountId": "12345678910",
 "userName": "SomeServiceCa-SomeLambdaR-ADLIAGB0VT9X"
 },
 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-29T15:46:04Z"
 }
 }
},
"eventTime": "2020-04-29T17:01:33Z",
"eventSource": "comprehend.amazonaws.com",
"eventName": "CreateDocumentClassifier",
"awsRegion": "us-east-2",
"sourceIPAddress": "3.20.236.234",
"userAgent": "aws-internal/3 aws-sdk-java/1.11.761 Linux/4.14.165-102.205.amzn2.x86_64
OpenJDK_64-Bit_Server_VM/25.201-b09 java/1.8.0_201 vendor/Oracle_Corporation, canary-
generated exec-env/AWS_Lambda_java8",
"requestParameters": {
 "documentClassifierName": "DocumentClassifier-FOUR-KB-MULTI-CLASS-1588179693504",
 "dataAccessRoleArn": "arn:aws:iam::12345678910:role/SomeServiceCanary-CMH-p-
DataAccessRole-1DI9T5MJ973BK",
 "tags": [
 {
 "key": "CreationTag",
 "value": "TagValue"
 }
],
 "inputDataConfig": {
 "s3Uri": "s3://aws-deepinsight-async-canary-datasets-prod-us-east-2/
ONE_DOC_PER_LINE/DOCUMENT_CLASSIFIER/FOUR_KB"
 },
 "clientRequestToken": "eb545dbf-52bd-474f-b0a2-b21010371e1b",
 "languageCode": "en",
 "volumeKmsKeyId": "arn:aws:kms:us-east-2:12345678910:key/7b9b8aff-c459-4d20-
bb39-64d6e3756e85",
 "mode": "MULTI_CLASS"
},
"responseElements": {
 "documentClassifierArn": "arn:aws:comprehend:us-east-2:12345678910:document-
classifier/DocumentClassifier-FOUR-KB-MULTI-CLASS-1588179693504"
},
"requestID": "20300c29-29bf-48fd-8275-e5a805ee8999",
"eventID": "3444051d-3616-438f-b80e-a3be9bbc0341",
"readOnly": false,
"resources": [
 {
 "accountId": "12345678910",
 "type": "AWS::Comprehend::DocumentClassifier",
 "ARN": "arn:aws:comprehend:us-east-2:12345678910:document-classifier/
DocumentClassifier-FOUR-KB-MULTI-CLASS-1588179693504"
 },
 {
 "accountId": "12345678910",
 "type": "AWS::KMS::Key",
 "ARN": "arn:aws:kms:us-east-2:12345678910:key/7b9b8aff-c459-4d20-
bb39-64d6e3756e85"
 },
 {

```

```

 "accountId": "12345678910",
 "type": "AWS::S3::Object",
 "ARN": "s3://aws-deepinsight-async-canary-datasets-prod-us-east-2/
ONE_DOC_PER_LINE/DOCUMENT_CLASSIFIER/FOUR_KB"
 },
 {
 "accountId": "12345678910",
 "type": "AWS::IAM::Role",
 "ARN": "arn:aws:iam::12345678910:role/SomeServiceCanary-CMH-p-
DataAccessRole-1DI9T5MJ973BK"
 }
],
"eventType": "AwsApiCall",
"recipientAccountId": "12345678910"
}

```

The following example shows a CloudTrail log entry that demonstrates the `DeleteEntityRecognizer` action.

```

{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "AROAICFHPEXAMPLE:SomeServiceCanary-
StartAsyncJobLambda-1STGUG0X870YM",
 "arn": "arn:aws:sts::12345678910:assumed-role/SomeServiceCa-SomeLambdaR-
ADLIAGB0VT9X/SomeServiceCanary-StartAsyncJobLambda-1STGUG0X870YM",
 "accountId": "12345678910",
 "accessKeyId": "ASIA3VZEXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AROAICFHPEXAMPLE",
 "arn": "arn:aws:iam::12345678910:role/SomeServiceCa-SomeLambdaR-
ADLIAGB0VT9X",
 "accountId": "12345678910",
 "userName": "SomeServiceCa-SomeLambdaR-ADLIAGB0VT9X"
 },
 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-29T15:46:04Z"
 }
 }
 },
 "eventTime": "2020-04-29T17:01:13Z",
 "eventSource": "comprehend.amazonaws.com",
 "eventName": "DeleteEntityRecognizer",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "3.20.236.234",
 "userAgent": "aws-internal/3 aws-sdk-java/1.11.761 Linux/4.14.165-102.205.amzn2.x86_64
OpenJDK_64-Bit_Server_VM/25.201-b09 java/1.8.0_201 vendor/Oracle_Corporation, canary-
generated exec-env/AWS_Lambda_java8",
 "requestParameters": {
 "entityRecognizerArn": "arn:aws:comprehend:us-east-2:12345678910:entity-recognizer/
EntityRecognizer-DOCS-2K-ONE-DOC-PER-LINE-1588161691587"
 },
 "responseElements": null,
 "requestID": "ab49a08d-3579-4f10-a62f-eb72e359e516",
 "eventID": "fb9860a1-b158-4764-a306-d19cfbace7fc",
 "readOnly": false,
 "resources": [
 {
 "accountId": "12345678910",

```

```

 "type": "AWS::Comprehend::EntityRecognizer",
 "ARN": "arn:aws:comprehend:us-east-2:12345678910:entity-recognizer/
EntityRecognizer-DOCS-2K-ONE-DOC-PER-LINE-1588161691587"
 }
],
"eventType": "AwsApiCall",
"recipientAccountId": "12345678910"
}

```

The following example shows a CloudTrail log entry that demonstrates the `ClassifyDocument` action.

```

{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "AROAICFHPEXAMPLE:SomeServiceCa-ValidateEndpointOutputLa-
C5F50672CNDK",
 "arn": "arn:aws:sts::12345678910:assumed-role/SomeServiceCa-SomeLambdaR-
ADLIAGB0VT9X/SomeServiceCa-ValidateEndpointOutputLa-C5F50672CNDK",
 "accountId": "12345678910",
 "accessKeyId": "ASIA3VZEXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AROAICFHPEXAMPLE",
 "arn": "arn:aws:iam::12345678910:role/SomeServiceCa-SomeLambdaR-
ADLIAGB0VT9X",
 "accountId": "12345678910",
 "userName": "SomeServiceCa-SomeLambdaR-ADLIAGB0VT9X"
 },
 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-29T16:15:11Z"
 }
 }
 },
 "eventTime": "2020-04-29T17:10:26Z",
 "eventSource": "comprehend.amazonaws.com",
 "eventName": "ClassifyDocument",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "3.21.185.237",
 "userAgent": "aws-internal/3 aws-sdk-java/1.11.761 Linux/4.14.165-102.205.amzn2.x86_64
OpenJDK_64-Bit_Server_VM/25.201-b09 java/1.8.0_201 vendor/Oracle_Corporation, canary-
generated exec-env/AWS_Lambda_java8",
 "requestParameters": {
 "endpointArn": "arn:aws:comprehend:us-east-2:12345678910:document-classifier-
endpoint/canary86644"
 },
 "responseElements": null,
 "requestID": "fd916e66-caac-46c9-a1fc-81a0ef33e61b",
 "eventID": "535ca22b-b3a3-4c13-b2c5-bf51ab082794",
 "readOnly": true,
 "resources": [
 {
 "accountId": "12345678910",
 "type": "AWS::Comprehend::DocumentClassifierEndpoint",
 "ARN": "arn:aws:comprehend:us-east-2:12345678910:document-classifier-endpoint/
canary86644"
 }
],
 "eventType": "AwsApiCall",
 "recipientAccountId": "12345678910"
}

```

## Compliance Validation for Amazon Comprehend

Third-party auditors assess the security and compliance of Amazon Comprehend as part of multiple AWS compliance programs. These include PCI, FedRAMP, HIPAA, and others. You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using Amazon Comprehend is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Config](#) – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

For a list of AWS services in scope of specific compliance programs, see [AWS Services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

## Resilience in Amazon Comprehend

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

## Infrastructure Security in Amazon Comprehend

As a managed service, Amazon Comprehend adheres to the [AWS Best Practices for Security, Identity, and Compliance](#).

To access Amazon Comprehend through the network, you use AWS published API calls. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems, such as Java 7 and later, support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an AWS Identity and Access Management (IAM) principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

# Permissions Required for a Custom Asynchronous Analysis Job

## Important

If you have an IAM policy which restricts model access, you won't be able to complete an inference job with a custom model. Your IAM policy should be updated to having a wildcard resource for a custom async analysis job.

If you are using the [StartDocumentClassificationJob](#) and [StartEntitiesDetectionJob](#) APIs, you need to update your IAM policy unless you are currently using wildcards as resources. If you are using a [StartEntitiesDetectionJob](#) using a pretrained model this does not impact you and you don't need to make any changes.

The following example policy contains an **outdated** reference.

```
{
 "Action": [
 "comprehend:StartDocumentClassificationJob",
 "comprehend:StartEntitiesDetectionJob",
],
 "Resource": [
 "arn:aws:comprehend:us-east-1:123456789012:document-classifier/myClassifier",
 "arn:aws:comprehend:us-east-1:123456789012:entity-recognizer/myRecognizer"
],
 "Effect": "Allow"
}
```

This is the **updated** policy you need to use to sucessfully run StartDocumentClassificationJob and StartEntitiesDetectionJob.

```
{
 "Action": [
 "comprehend:StartDocumentClassificationJob",
 "comprehend:StartEntitiesDetectionJob",
],
 "Resource": [
 "arn:aws:comprehend:us-east-1:123456789012:document-classifier/myClassifier",
 "arn:aws:comprehend:us-east-1:123456789012:document-classification-job/*",
 "arn:aws:comprehend:us-east-1:123456789012:entity-recognizer/myRecognizer",
 "arn:aws:comprehend:us-east-1:123456789012:entities-detection-job/*"
],
 "Effect": "Allow"
}
```

# Guidelines and Quotas

Many of the Amazon Comprehend quotas shown here can be increased if needed for your applications. For information about service quotas and to request a quota increase, see [AWS Service Quotas](#).

Remember the following information when using Amazon Comprehend.

## Supported Regions

For a list of AWS Regions where Amazon Comprehend is available, see [AWS Regions and Endpoints](#) in the [Amazon Web Services General Reference](#).

## Overall Quotas

All operations except asynchronous operations and topic modeling operations have the following quotas:

| Description                      | Quota/Guideline |
|----------------------------------|-----------------|
| Character encoding               | UTF-8           |
| Document size (UTF-8 characters) | 5,000 bytes     |

Synchronous requests to label documents with PII using the [ContainsPiiEntities \(p. 238\)](#) operation have the following quotas:

| Description                      | Quota/Guideline |
|----------------------------------|-----------------|
| Character encoding               | UTF-8           |
| Document size (UTF-8 characters) | 50,000 bytes    |

Amazon Comprehend may store your content to continuously improve the quality of its analysis models. See the [Amazon Comprehend FAQ](#) to learn more.

## Throttling When Using Single Transactions

You may be able to avoid throttling by using the batch operations instead of the single transaction operations. For more information, see [Multiple Document Operations \(p. 212\)](#).

## Multiple Document Operations

The [BatchDetectDominantLanguage \(p. 220\)](#), [BatchDetectEntities \(p. 223\)](#), [BatchDetectKeyPhrases \(p. 226\)](#), and [BatchDetectSentiment \(p. 229\)](#) operations have the following quotas:

| Description           | Quota/Guideline |
|-----------------------|-----------------|
| Documents per request | 25              |

If you plan to send more than 20 requests per second, you should consider using the batch operations. Batch operations enable you to send more documents in each request which may result in higher throughput. For example, when you use the `DetectDominantLanguage` operation, you can send up to 20 documents per second. However, if you use the `BatchRequestDominantLanguage` operation, you can send up to 250 documents per second, but processing speed may be lower. For more information about throttling quotas see [Amazon Comprehend Quotas](#) in the *Amazon Web Services General Reference*. For more information about using the multiple document APIs, see [Multiple Document Synchronous Processing \(p. 122\)](#).

## Asynchronous Operations

Asynchronous batch jobs are run with the [StartDominantLanguageDetectionJob \(p. 354\)](#), [StartEntitiesDetectionJob \(p. 359\)](#), [StartKeyPhrasesDetectionJob \(p. 368\)](#), [StartPiiEntitiesDetectionJob \(p. 373\)](#), and [StartSentimentDetectionJob \(p. 377\)](#) operations. These jobs have the following limits:

| Description                                                                                 | Quota/Guideline |
|---------------------------------------------------------------------------------------------|-----------------|
| Maximum size of each document in jobs that detect entities, key phrases, PII, and languages | 1 MB            |
| Maximum size of each document in jobs that detect sentiment                                 | 5 KB            |
| Total size of all files in batch                                                            | 5 GB            |
| Maximum number of files, one document per file                                              | 1,000,000       |

You should use the asynchronous operations:

- To analyze more than 25 documents at a time
- To analyze documents larger than 5,000 bytes for keywords and entities

For more information, see [Asynchronous Batch Processing \(p. 117\)](#).

## Document Classification

Document classifier training jobs started with the [CreateDocumentClassifier \(p. 240\)](#) operation, asynchronous document classification jobs started with the [StartDocumentClassificationJob \(p. 349\)](#), and synchronous document classification requests started with the [ClassifyDocument \(p. 235\)](#) operation have the following limits:

### General

| Description        | Quota/Guideline |
|--------------------|-----------------|
| Character encoding | UTF-8           |

| Description                                                                 | Quota/Guideline  |
|-----------------------------------------------------------------------------|------------------|
| Maximum number of classes (multi-class mode)                                | 1,000            |
| Maximum number of classes (multi-label mode)                                | 100              |
| Maximum length of class name                                                | 5,000 characters |
| Minimum number of training documents per class (multi-class mode)           | 50               |
| Minimum number of training documents per class, (multi-label mode)          | 10               |
| Total size of all files in training job                                     | 5 GB             |
| Total size of all files in asynchronous job                                 | 5 GB             |
| Maximum file size for one file, one document per file                       | 10 MB            |
| Maximum number of files, one document per file                              | 1,000,000        |
| Maximum number of lines, one document per line (for all files in request)   | 1,000,000        |
| Maximum number of documents per synchronous request                         | 1                |
| Maximum number of augmented manifest files for training a custom classifier | 5                |
| Maximum number of attribute names for each augmented manifest file          | 5                |
| Maximum length of attribute name                                            | 63 characters    |

### Real-time analysis

| Description                                        | Quota/Guideline |
|----------------------------------------------------|-----------------|
| Maximum number of inference units per account      | 100             |
| Maximum number of inference units per endpoint     | 10              |
| Maximum throughput per inference unit (characters) | 100/second      |
| Maximum throughput per inference unit (documents)  | 2/second        |

## Language Detection

The [BatchDetectDominantLanguage \(p. 220\)](#), [DetectDominantLanguage \(p. 291\)](#) operations and asynchronous jobs started with the [StartDominantLanguageDetectionJob \(p. 354\)](#) operation have the following limitations:

- They don't support phonetic language detection. For example, they will not detect "arigato" as Japanese nor "nihao" as Chinese.
- They may have trouble distinguishing close language pairs, such as Indonesian and Malay; or Bosnian, Croatian, and Serbian.

- For best results the input text should be at least 20 characters long.

## Events

Events detection jobs created with the [StartEventsDetectionJob \(p. 364\)](#) operation have the following limits:

| Description                                                               | Quotas |
|---------------------------------------------------------------------------|--------|
| Character encoding                                                        | UTF-8  |
| Total size of all files in a job                                          | 50 MB  |
| Maximum size of each document in a job                                    | 10 KB  |
| Maximum number of files, one document per file                            | 5,000  |
| Maximum number of lines, one document per line (for all files in request) | 5,000  |

## Topic Modeling

Topic detection jobs created with the [StartTopicsDetectionJob \(p. 382\)](#) operation have the following limits:

| Description                                                               | Quota/Guideline |
|---------------------------------------------------------------------------|-----------------|
| Character encoding                                                        | UTF-8           |
| Maximum number of topics to return                                        | 100             |
| Maximum total size of all files in request                                | 5 GB            |
| Minimum total size of all files in request                                | 500 bytes       |
| Maximum file size for one file, one document per file                     | 100 MB          |
| Maximum number of files, one document per file                            | 1,000,000       |
| Maximum number of lines, one document per line (for all files in request) | 1,000,000       |

For best results, you should include at least 1,000 input documents.

## Entity Recognition

Entity recognizer training jobs started with the [CreateEntityRecognizer \(p. 249\)](#) operation, asynchronous entity recognition jobs started with the [StartEntitiesDetectionJob \(p. 359\)](#) operation, and synchronous entity recognition requests started with the [DetectEntities \(p. 294\)](#) operation have the following limits:

### Plain text entity recognition - training ([CreateEntityRecognizer](#))

| Description                                               | Quota/Guideline |
|-----------------------------------------------------------|-----------------|
| Number of entities per model/custom entity recognizer     | 1 - 25          |
| Document size (UTF-8)                                     | 1 - 5,000 byte  |
| Number of documents                                       | 250 - 120,000   |
| Document corpus size (all docs in plain text combined)    | 5 KB - 100MB    |
| Minimum number of annotations per entity                  | 100             |
| Number of items in entity list                            | 1 - 1 million   |
| Length of individual entry (post-strip) in entry list     | 1 - 5,000       |
| Entity list corpus size (all docs in plain text combined) | 5KB - 100 MB    |

### PDF or Word text entity recognition - training ([CreateEntityRecognizer](#))

| Description                                            | Quota/Guideline |
|--------------------------------------------------------|-----------------|
| Number of entities per model/custom entity recognizer  | 1 - 25          |
| Maximum annotation file size (UTF-8 JSON)              | 5 MB            |
| Number of documents                                    | 250 - 10,000    |
| Document corpus size (all docs in plain text combined) | 5 KB - 1 GB     |
| Minimum number of annotations per entity               | 100             |

### Augmented manifest files

| Description                                                                        | Quota/Guideline |
|------------------------------------------------------------------------------------|-----------------|
| Maximum number of augmented manifest files for training a custom entity recognizer | 5               |
| Maximum number of attribute names for each augmented manifest file                 | 5               |
| Maximum length of attribute name                                                   | 63 characters   |

### Plain text entity recognition - inference ([StartEntitiesDetectionJob](#))

| Description                                                               | Quota/Guideline |
|---------------------------------------------------------------------------|-----------------|
| Document size (UTF-8)                                                     | 1 byte - 1 MB   |
| Maximum number of files, one document per file                            | 1,000,000       |
| Maximum number of lines, one document per line (for all files in request) | 1,000,000       |
| Document corpus size (all docs in plain text combined)                    | 1 byte - 5 GB   |

### **PDF or Word text entity recognition - inference (StartEntitiesDetectionJob)**

| Description                                                                                                  | Quota/Guideline |
|--------------------------------------------------------------------------------------------------------------|-----------------|
| Document size (PDF)                                                                                          | 1 byte - 50 MB  |
| Document size (Docx)                                                                                         | 1 byte - 5 MB   |
| Document size (UTF-8)                                                                                        | 1 byte - 1 MB   |
| Maximum number of files, one document per file (one document per line not allowed for PDF or Word documents) | 500             |
| Maximum number of pages for single PDF or Docx file                                                          | 100             |
| Document corpus size (all docs in plain text combined)                                                       | 1 byte - 5 GB   |

### **Real-time analysis**

| Description                                         | Quota/Guideline |
|-----------------------------------------------------|-----------------|
| Maximum number of inference units per account       | 100             |
| Maximum number of inference units per endpoint      | 10              |
| Maximum throughput per inference unit (characters)  | 100/second      |
| Maximum throughput per inference unit (documents)   | 2/second        |
| Maximum number of documents per synchronous request | 1               |

# API Reference

This section provides documentation for the Amazon Comprehend API operations.

## Actions

The following actions are supported:

- [BatchDetectDominantLanguage \(p. 220\)](#)
- [BatchDetectEntities \(p. 223\)](#)
- [BatchDetectKeyPhrases \(p. 226\)](#)
- [BatchDetectSentiment \(p. 229\)](#)
- [BatchDetectSyntax \(p. 232\)](#)
- [ClassifyDocument \(p. 235\)](#)
- [ContainsPiiEntities \(p. 238\)](#)
- [CreateDocumentClassifier \(p. 240\)](#)
- [CreateEndpoint \(p. 245\)](#)
- [CreateEntityRecognizer \(p. 249\)](#)
- [DeleteDocumentClassifier \(p. 254\)](#)
- [DeleteEndpoint \(p. 256\)](#)
- [DeleteEntityRecognizer \(p. 258\)](#)
- [DescribeDocumentClassificationJob \(p. 260\)](#)
- [DescribeDocumentClassifier \(p. 263\)](#)
- [DescribeDominantLanguageDetectionJob \(p. 266\)](#)
- [DescribeEndpoint \(p. 269\)](#)
- [DescribeEntitiesDetectionJob \(p. 271\)](#)
- [DescribeEntityRecognizer \(p. 274\)](#)
- [DescribeEventsDetectionJob \(p. 277\)](#)
- [DescribeKeyPhrasesDetectionJob \(p. 279\)](#)
- [DescribePiiEntitiesDetectionJob \(p. 282\)](#)
- [DescribeSentimentDetectionJob \(p. 285\)](#)
- [DescribeTopicsDetectionJob \(p. 288\)](#)
- [DetectDominantLanguage \(p. 291\)](#)
- [DetectEntities \(p. 294\)](#)
- [DetectKeyPhrases \(p. 298\)](#)
- [DetectPiiEntities \(p. 301\)](#)
- [DetectSentiment \(p. 303\)](#)
- [DetectSyntax \(p. 306\)](#)
- [ListDocumentClassificationJobs \(p. 309\)](#)
- [ListDocumentClassifiers \(p. 312\)](#)
- [ListDocumentClassifierSummaries \(p. 315\)](#)
- [ListDominantLanguageDetectionJobs \(p. 317\)](#)
- [ListEndpoints \(p. 320\)](#)

- [ListEntitiesDetectionJobs \(p. 323\)](#)
- [ListEntityRecognizers \(p. 326\)](#)
- [ListEntityRecognizerSummaries \(p. 330\)](#)
- [ListEventsDetectionJobs \(p. 332\)](#)
- [ListKeyPhrasesDetectionJobs \(p. 335\)](#)
- [ListPiiEntitiesDetectionJobs \(p. 338\)](#)
- [ListSentimentDetectionJobs \(p. 341\)](#)
- [ListTagsForResource \(p. 344\)](#)
- [ListTopicsDetectionJobs \(p. 346\)](#)
- [StartDocumentClassificationJob \(p. 349\)](#)
- [StartDominantLanguageDetectionJob \(p. 354\)](#)
- [StartEntitiesDetectionJob \(p. 359\)](#)
- [StartEventsDetectionJob \(p. 364\)](#)
- [StartKeyPhrasesDetectionJob \(p. 368\)](#)
- [StartPiiEntitiesDetectionJob \(p. 373\)](#)
- [StartSentimentDetectionJob \(p. 377\)](#)
- [StartTopicsDetectionJob \(p. 382\)](#)
- [StopDominantLanguageDetectionJob \(p. 387\)](#)
- [StopEntitiesDetectionJob \(p. 389\)](#)
- [StopEventsDetectionJob \(p. 391\)](#)
- [StopKeyPhrasesDetectionJob \(p. 393\)](#)
- [StopPiiEntitiesDetectionJob \(p. 395\)](#)
- [StopSentimentDetectionJob \(p. 397\)](#)
- [StopTrainingDocumentClassifier \(p. 399\)](#)
- [StopTrainingEntityRecognizer \(p. 401\)](#)
- [TagResource \(p. 403\)](#)
- [UntagResource \(p. 405\)](#)
- [UpdateEndpoint \(p. 407\)](#)

## BatchDetectDominantLanguage

Determines the dominant language of the input text for a batch of documents. For a list of languages that Amazon Comprehend can detect, see [Amazon Comprehend Supported Languages](#).

### Request Syntax

```
{
 "TextList": ["string"]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### TextList (p. 220)

A list containing the text of the input documents. The list can contain a maximum of 25 documents. Each document should contain at least 20 characters and must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: Array of strings

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "ErrorList": [
 {
 "ErrorCode": "string",
 "ErrorMessage": "string",
 "Index": number
 }
],
 "ResultList": [
 {
 "Index": number,
 "Languages": [
 {
 "LanguageCode": "string",
 "Score": number
 }
]
 }
]
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### **ErrorList (p. 220)**

A list containing one [BatchItemError \(p. 418\)](#) object for each document that contained an error. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If there are no errors in the batch, the `ErrorList` is empty.

Type: Array of [BatchItemError \(p. 418\)](#) objects

#### **ResultList (p. 220)**

A list of [BatchDetectDominantLanguageItemResult \(p. 413\)](#) objects containing the results of the operation. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If all of the documents contain an error, the `ResultList` is empty.

Type: Array of [BatchDetectDominantLanguageItemResult \(p. 413\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

#### **BatchSizeLimitExceededException**

The number of documents in the request exceeds the limit of 25. Try your request again with fewer documents.

HTTP Status Code: 400

#### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

#### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

#### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)

- [AWS SDK for Ruby V3](#)

## BatchDetectEntities

Inspects the text of a batch of documents for named entities and returns information about them. For more information about named entities, see [Detect Entities \(p. 94\)](#)

### Request Syntax

```
{
 "LanguageCode": "string",
 "TextList": ["string"]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **LanguageCode (p. 223)**

The language of the input documents. You can specify any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: Yes

#### **TextList (p. 223)**

A list containing the text of the input documents. The list can contain a maximum of 25 documents. Each document must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: Array of strings

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "ErrorList": [
 {
 "ErrorCode": "string",
 "ErrorMessage": "string",
 "Index": number
 }
],
 "ResultList": [
 {
 "Entities": [
 {
 "BeginOffset": number,
 "EndOffset": number,
 "Score": number,
 "Text": "string",
 "Type": "string"
 }
]
 }
]
}
```

```
 "Type": "string"
 }
],
"Index": number
}
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### ErrorList (p. 223)

A list containing one [BatchItemError \(p. 418\)](#) object for each document that contained an error. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If there are no errors in the batch, the `ErrorList` is empty.

Type: Array of [BatchItemError \(p. 418\)](#) objects

### ResultList (p. 223)

A list of [BatchDetectEntitiesItemResult \(p. 414\)](#) objects containing the results of the operation. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If all of the documents contain an error, the `ResultList` is empty.

Type: Array of [BatchDetectEntitiesItemResult \(p. 414\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### BatchSizeLimitExceededException

The number of documents in the request exceeds the limit of 25. Try your request again with fewer documents.

HTTP Status Code: 400

### InternalServerError

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### InvalidRequestException

The request is invalid.

HTTP Status Code: 400

### TextSizeLimitExceededException

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### UnsupportedLanguageException

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## BatchDetectKeyPhrases

Detects the key noun phrases found in a batch of documents.

### Request Syntax

```
{
 "LanguageCode": "string",
 "TextList": ["string"]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### LanguageCode (p. 226)

The language of the input documents. You can specify any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: Yes

#### TextList (p. 226)

A list containing the text of the input documents. The list can contain a maximum of 25 documents. Each document must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: Array of strings

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "ErrorList": [
 {
 "ErrorCode": "string",
 "ErrorMessage": "string",
 "Index": number
 }
],
 "ResultList": [
 {
 "Index": number,
 "KeyPhrases": [
 {
 "BeginOffset": number,
 "EndOffset": number,
 "Score": number,
 "Text": "string"
 }
]
 }
]
}
```

```
 "Text": "string"
 }
]
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [ErrorList \(p. 226\)](#)

A list containing one [BatchItemError \(p. 418\)](#) object for each document that contained an error. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If there are no errors in the batch, the `ErrorList` is empty.

Type: Array of [BatchItemError \(p. 418\)](#) objects

### [ResultList \(p. 226\)](#)

A list of [BatchDetectKeyPhrasesItemResult \(p. 415\)](#) objects containing the results of the operation. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If all of the documents contain an error, the `ResultList` is empty.

Type: Array of [BatchDetectKeyPhrasesItemResult \(p. 415\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **BatchSizeLimitExceededException**

The number of documents in the request exceeds the limit of 25. Try your request again with fewer documents.

HTTP Status Code: 400

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### **UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## BatchDetectSentiment

Inspects a batch of documents and returns an inference of the prevailing sentiment, POSITIVE, NEUTRAL, MIXED, or NEGATIVE, in each one.

### Request Syntax

```
{
 "LanguageCode": "string",
 "TextList": ["string"]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### LanguageCode ([p. 229](#))

The language of the input documents. You can specify any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: Yes

#### TextList ([p. 229](#))

A list containing the text of the input documents. The list can contain a maximum of 25 documents. Each document must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: Array of strings

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "ErrorList": [
 {
 "ErrorCode": "string",
 "ErrorMessage": "string",
 "Index": number
 }
],
 "ResultList": [
 {
 "Index": number,
 "Sentiment": "string",
 "SentimentScore": {
 "Mixed": number,
 "Negative": number,
 "Positive": number,
 "Neutral": number
 }
 }
]
}
```

```
 "Neutral": number,
 "Positive": number
 }
]
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### **ErrorList** (p. 229)

A list containing one [BatchItemError](#) (p. 418) object for each document that contained an error. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If there are no errors in the batch, the `ErrorList` is empty.

Type: Array of [BatchItemError](#) (p. 418) objects

### **ResultList** (p. 229)

A list of [BatchDetectSentimentItemResult](#) (p. 416) objects containing the results of the operation. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If all of the documents contain an error, the `ResultList` is empty.

Type: Array of [BatchDetectSentimentItemResult](#) (p. 416) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **BatchSizeLimitExceededException**

The number of documents in the request exceeds the limit of 25. Try your request again with fewer documents.

HTTP Status Code: 400

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### **UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## BatchDetectSyntax

Inspects the text of a batch of documents for the syntax and part of speech of the words in the document and returns information about them. For more information, see [Analyze Syntax \(p. 112\)](#).

### Request Syntax

```
{
 "LanguageCode": "string",
 "TextList": ["string"]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [LanguageCode \(p. 232\)](#)

The language of the input documents. You can specify any of the following languages supported by Amazon Comprehend: German ("de"), English ("en"), Spanish ("es"), French ("fr"), Italian ("it"), or Portuguese ("pt"). All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt

Required: Yes

#### [TextList \(p. 232\)](#)

A list containing the text of the input documents. The list can contain a maximum of 25 documents. Each document must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: Array of strings

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "ErrorList": [
 {
 "ErrorCode": "string",
 "ErrorMessage": "string",
 "Index": number
 }
],
 "ResultList": [
 {
 "Index": number,
 "SyntaxTokens": [
 {
 "BeginOffset": number,
 "EndOffset": number,
 "Text": "string"
 }
]
 }
]
}
```

```
 "EndOffset": number,
 "PartOfSpeech": {
 "Score": number,
 "Tag": "string"
 },
 "Text": "string",
 "TokenId": number
 }
}
]
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [ErrorList \(p. 232\)](#)

A list containing one [BatchItemError \(p. 418\)](#) object for each document that contained an error. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If there are no errors in the batch, the `ErrorList` is empty.

Type: Array of [BatchItemError \(p. 418\)](#) objects

### [ResultList \(p. 232\)](#)

A list of [BatchDetectSyntaxItemResult \(p. 417\)](#) objects containing the results of the operation. The results are sorted in ascending order by the `Index` field and match the order of the documents in the input list. If all of the documents contain an error, the `ResultList` is empty.

Type: Array of [BatchDetectSyntaxItemResult \(p. 417\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **BatchSizeLimitExceededException**

The number of documents in the request exceeds the limit of 25. Try your request again with fewer documents.

HTTP Status Code: 400

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### **UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ClassifyDocument

Creates a new document classification request to analyze a single document in real-time, using a previously created and trained custom model and an endpoint.

### Request Syntax

```
{
 "EndpointArn": "string",
 "Text": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [EndpointArn \(p. 235\)](#)

The Amazon Resource Number (ARN) of the endpoint.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:document-classifier-endpoint/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

Required: Yes

#### [Text \(p. 235\)](#)

The document text to be analyzed.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "Classes": [
 {
 "Name": "string",
 "Score": number
 }
],
 "Labels": [
 {
 "Name": "string",
 "Score": number
 }
]
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [Classes \(p. 235\)](#)

The classes used by the document being analyzed. These are used for multi-class trained models. Individual classes are mutually exclusive and each document is expected to have only a single class assigned to it. For example, an animal can be a dog or a cat, but not both at the same time.

Type: Array of [DocumentClass \(p. 422\)](#) objects

### [Labels \(p. 235\)](#)

The labels used the document being analyzed. These are used for multi-label trained models. Individual labels represent different categories that are related in some manner and are not mutually exclusive. For example, a movie can be just an action movie, or it can be an action movie, a science fiction movie, and a comedy, all at the same time.

Type: Array of [DocumentLabel \(p. 437\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceUnavailableException**

The specified resource is not available. Check the resource and try your request again.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ContainsPiiEntities

Analyzes input text for the presence of personally identifiable information (PII) and returns the labels of identified PII entity types such as name, address, bank account number, or phone number.

### Request Syntax

```
{
 "LanguageCode": "string",
 "Text": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [LanguageCode \(p. 238\)](#)

The language of the input documents.

Type: String

Valid Values: en

Required: Yes

#### [Text \(p. 238\)](#)

Creates a new document classification request to analyze a single document in real-time, returning personally identifiable information (PII) entity labels.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "Labels": [
 {
 "Name": "string",
 "Score": number
 }
]
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

## [Labels \(p. 238\)](#)

The labels used in the document being analyzed. Individual labels represent personally identifiable information (PII) entity types.

Type: Array of [EntityLabel \(p. 454\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### **UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## CreateDocumentClassifier

Creates a new document classifier that you can use to categorize documents. To create a classifier, you provide a set of training documents that labeled with the categories that you want to use. After the classifier is trained you can use it to categorize a set of labeled documents into the categories. For more information, see [Custom Classification \(p. 125\)](#).

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "DocumentClassifierName": "string",
 "InputDataConfig": {
 "AugmentedManifests": [
 {
 "AnnotationDataS3Uri": "string",
 "AttributeNames": ["string"],
 "DocumentType": "string",
 "S3Uri": "string",
 "SourceDocumentsS3Uri": "string",
 "Split": "string"
 }
],
 "DataFormat": "string",
 "LabelDelimiter": "string",
 "S3Uri": "string",
 "TestS3Uri": "string"
 },
 "LanguageCode": "string",
 "Mode": "string",
 "ModelKmsKeyId": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "VersionName": "string",
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **ClientRequestToken (p. 240)**

A unique identifier for the request. If you don't set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[ a-zA-Z0-9-]+\$

Required: No

#### [DataAccessRoleArn \(p. 240\)](#)

The Amazon Resource Name (ARN) of the AWS Identity and Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam:::[0-9]{12}:role/.+

Required: Yes

#### [DocumentClassifierName \(p. 240\)](#)

The name of the document classifier.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[ a-zA-Z0-9 ](-\*[ a-zA-Z0-9 ])\*\$

Required: Yes

#### [InputDataConfig \(p. 240\)](#)

Specifies the format and location of the input data for the job.

Type: [DocumentClassifierInputDataConfig \(p. 428\)](#) object

Required: Yes

#### [LanguageCode \(p. 240\)](#)

The language of the input documents. You can specify any of the following languages supported by Amazon Comprehend: German ("de"), English ("en"), Spanish ("es"), French ("fr"), Italian ("it"), or Portuguese ("pt"). All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt

Required: Yes

#### [Mode \(p. 240\)](#)

Indicates the mode in which the classifier will be trained. The classifier can be trained in multi-class mode, which identifies one and only one class for each document, or multi-label mode, which identifies one or more labels for each document. In multi-label mode, multiple labels for an individual document are separated by a delimiter. The default delimiter between labels is a pipe (|).

Type: String

Valid Values: MULTI\_CLASS | MULTI\_LABEL

Required: No

### [ModelKmsKeyId \(p. 240\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt trained custom models. The ModelKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

### [OutputDataConfig \(p. 240\)](#)

Enables the addition of output results configuration parameters for custom classifier jobs.

Type: [DocumentClassifierOutputDataConfig \(p. 430\)](#) object

Required: No

### [Tags \(p. 240\)](#)

Tags to be associated with the document classifier being created. A tag is a key-value pair that adds as a metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

### [VersionName \(p. 240\)](#)

The version name given to the newly created classifier. Version names can have a maximum of 256 characters. Alphanumeric characters, hyphens (-) and underscores (\_) are allowed. The version name must be unique among all models with the same classifier name in the account/AWS Region.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

### [VolumeKmsKeyId \(p. 240\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### [VpcConfig \(p. 240\)](#)

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for your custom classifier. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## Response Syntax

```
{
 "DocumentClassifierArn": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [DocumentClassifierArn \(p. 243\)](#)

The Amazon Resource Name (ARN) that identifies the document classifier.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:document-classifier/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

#### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

#### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

#### **KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

#### **ResourceInUseException**

The specified resource name is already in use. Use a different name and try your request again.

HTTP Status Code: 400

**ResourceLimitExceeded**

The maximum number of resources per account has been exceeded. Review the resources, and then try your request again.

HTTP Status Code: 400

**TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

**TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

**UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

# CreateEndpoint

Creates a model-specific endpoint for synchronous inference for a previously trained custom model

## Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "DesiredInferenceUnits": number,
 "EndpointName": "string",
 "ModelArn": "string",
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
]
}
```

## Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

### **ClientRequestToken** (p. 245)

An idempotency token provided by the customer. If this token matches a previous endpoint creation request, Amazon Comprehend will not return a `ResourceInUseException`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

### **DataAccessRoleArn** (p. 245)

The Amazon Resource Name (ARN) of the AWS identity and Access Management (IAM) role that grants Amazon Comprehend read access to trained custom models encrypted with a customer managed key (`ModelKmsKeyId`).

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: No

### **DesiredInferenceUnits** (p. 245)

The desired number of inference units to be used by the model using this endpoint. Each inference unit represents of a throughput of 100 characters per second.

Type: Integer

Valid Range: Minimum value of 1.

Required: Yes

#### [EndpointName \(p. 245\)](#)

This is the descriptive suffix that becomes part of the `EndpointArn` used for all subsequent requests to this resource.

Type: String

Length Constraints: Maximum length of 40.

Pattern: `^[a-zA-Z0-9](-*[a-zA-Z0-9])*$`

Required: Yes

#### [ModelArn \(p. 245\)](#)

The Amazon Resource Number (ARN) of the model to which the endpoint will be attached.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:(document-classifier|entity-recognizer)/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*)?`

Required: Yes

#### [Tags \(p. 245\)](#)

Tags associated with the endpoint being created. A tag is a key-value pair that adds metadata to the endpoint. For example, a tag with "Sales" as the key might be added to an endpoint to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

## Response Syntax

```
{
 "EndpointArn": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [EndpointArn \(p. 246\)](#)

The Amazon Resource Number (ARN) of the endpoint being created.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier-endpoint|entity-recognizer-endpoint)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceInUseException**

The specified resource name is already in use. Use a different name and try your request again.

HTTP Status Code: 400

### **ResourceLimitExceededException**

The maximum number of resources per account has been exceeded. Review the resources, and then try your request again.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **ResourceUnavailableException**

The specified resource is not available. Check the resource and try your request again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

### **TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## CreateEntityRecognizer

Creates an entity recognizer using submitted files. After your `CreateEntityRecognizer` request is submitted, you can check job status using the [DescribeEntityRecognizer \(p. 274\)](#) API.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "InputDataConfig": {
 "Annotations": {
 "S3Uri": "string",
 "TestS3Uri": "string"
 },
 "AugmentedManifests": [
 {
 "AnnotationDataS3Uri": "string",
 "AttributeNames": ["string"],
 "DocumentType": "string",
 "S3Uri": "string",
 "SourceDocumentsS3Uri": "string",
 "Split": "string"
 }
],
 "DataFormat": "string",
 "Documents": {
 "InputFormat": "string",
 "S3Uri": "string",
 "TestS3Uri": "string"
 },
 "EntityList": {
 "S3Uri": "string"
 },
 "EntityTypes": [
 {
 "Type": "string"
 }
]
 },
 "LanguageCode": "string",
 "ModelKmsKeyId": "string",
 "RecognizerName": "string",
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "VersionName": "string",
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

**[ClientRequestToken \(p. 249\)](#)**

A unique identifier for the request. If you don't set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

**[DataAccessRoleArn \(p. 249\)](#)**

The Amazon Resource Name (ARN) of the AWS Identity and Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: Yes

**[InputDataConfig \(p. 249\)](#)**

Specifies the format and location of the input data. The S3 bucket containing the input data must be located in the same region as the entity recognizer being created.

Type: [EntityRecognizerInputDataConfig \(p. 460\)](#) object

Required: Yes

**[LanguageCode \(p. 249\)](#)**

You can specify any of the following languages supported by Amazon Comprehend: English ("en"), Spanish ("es"), French ("fr"), Italian ("it"), German ("de"), or Portuguese ("pt"). All documents must be in the same language.

Type: String

Valid Values: en | es | fr | it | de | pt

Required: Yes

**[ModelKmsKeyId \(p. 249\)](#)**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt trained custom models. The ModelKmsKeyId can be either of the following formats

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

### [RecognizerName \(p. 249\)](#)

The name given to the newly created recognizer. Recognizer names can be a maximum of 256 characters. Alphanumeric characters, hyphens (-) and underscores (\_) are allowed. The name must be unique in the account/region.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: Yes

### [Tags \(p. 249\)](#)

Tags to be associated with the entity recognizer being created. A tag is a key-value pair that adds as a metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

### [VersionName \(p. 249\)](#)

The version name given to the newly created recognizer. Version names can be a maximum of 256 characters. Alphanumeric characters, hyphens (-) and underscores (\_) are allowed. The version name must be unique among all models with the same recognizer name in the account/ AWS Region.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

### [VolumeKmsKeyId \(p. 249\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

### [VpcConfig \(p. 249\)](#)

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for your custom entity recognizer. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## Response Syntax

```
{
 "EntityRecognizerArn": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### **EntityRecognizerArn** (p. 252)

The Amazon Resource Name (ARN) that identifies the entity recognizer.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:entity-recognizer/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

### **ResourceInUseException**

The specified resource name is already in use. Use a different name and try your request again.

HTTP Status Code: 400

### **ResourceLimitExceededException**

The maximum number of resources per account has been exceeded. Review the resources, and then try your request again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

**TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

**UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DeleteDocumentClassifier

Deletes a previously created document classifier

Only those classifiers that are in terminated states (IN\_ERROR, TRAINED) will be deleted. If an active inference job is using the model, a `ResourceInUseException` will be returned.

This is an asynchronous action that puts the classifier into a DELETING state, and it is then removed by a background job. Once removed, the classifier disappears from your account and is no longer available for use.

### Request Syntax

```
{
 "DocumentClassifierArn": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **DocumentClassifierArn** (p. 254)

The Amazon Resource Name (ARN) that identifies the document classifier.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:document-classifier/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*?)?`

Required: Yes

### Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

### Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

#### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

#### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceInUseException**

The specified resource name is already in use. Use a different name and try your request again.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **ResourceUnavailableException**

The specified resource is not available. Check the resource and try your request again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DeleteEndpoint

Deletes a model-specific endpoint for a previously-trained custom model. All endpoints must be deleted in order for the model to be deleted.

### Request Syntax

```
{
 "EndpointArn": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### EndpointArn (p. 256)

The Amazon Resource Number (ARN) of the endpoint being deleted.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier-endpoint|entity-recognizer-endpoint)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

Required: Yes

### Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

### Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

#### InternalServerError

An internal server error occurred. Retry your request.

HTTP Status Code: 500

#### InvalidRequestException

The request is invalid.

HTTP Status Code: 400

#### ResourceInUseException

The specified resource name is already in use. Use a different name and try your request again.

HTTP Status Code: 400

#### ResourceNotFoundException

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### TooManyRequestsException

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DeleteEntityRecognizer

Deletes an entity recognizer.

Only those recognizers that are in terminated states (IN\_ERROR, TRAINED) will be deleted. If an active inference job is using the model, a `ResourceInUseException` will be returned.

This is an asynchronous action that puts the recognizer into a DELETING state, and it is then removed by a background job. Once removed, the recognizer disappears from your account and is no longer available for use.

### Request Syntax

```
{
 "EntityRecognizerArn": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **EntityRecognizerArn** (p. 258)

The Amazon Resource Name (ARN) that identifies the entity recognizer.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:entity-recognizer/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*?)?`

Required: Yes

### Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

### Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

#### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

#### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceInUseException**

The specified resource name is already in use. Use a different name and try your request again.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **ResourceUnavailableException**

The specified resource is not available. Check the resource and try your request again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeDocumentClassificationJob

Gets the properties associated with a document classification job. Use this operation to get the status of a classification job.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### JobId (p. 260)

The identifier that Amazon Comprehend generated for the job. The [StartDocumentClassificationJob \(p. 349\)](#) operation returns this identifier in its response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:+\\-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "DocumentClassificationJobProperties": {
 "DataAccessRoleArn": "string",
 "DocumentClassifierArn": "string",
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 }
}
```

```
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [DocumentClassificationJobProperties \(p. 260\)](#)

An object that describes the properties associated with the document classification job.

Type: [DocumentClassificationJobProperties \(p. 424\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeDocumentClassifier

Gets the properties associated with a document classifier.

## Request Syntax

```
{
 "DocumentClassifierArn": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

## DocumentClassifierArn (p. 263)

The Amazon Resource Name (ARN) that identifies the document classifier. The [CreateDocumentClassifier](#) (p. 240) operation returns this identifier in its response.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:document-classifier/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9]))\*)?

Required: Yes

## Response Syntax

```
{
 "DocumentClassifierProperties": {
 "ClassifierMetadata": {
 "EvaluationMetrics": {
 "Accuracy": "number",
 "F1Score": "number",
 "HammingLoss": "number",
 "MicroF1Score": "number",
 "MicroPrecision": "number",
 "MicroRecall": "number",
 "Precision": "number",
 "Recall": "number"
 },
 "NumberOfLabels": "number",
 "NumberOfTestDocuments": "number",
 "NumberOfTrainedDocuments": "number"
 },
 "DataAccessRoleArn": "string",
 "DocumentClassifierArn": "string",
 "EndTime": "number",
 "InputDataConfig": {
 "AugmentedManifests": [
 {
 "AnnotationDataS3Uri": "string",
 "AttributeNames": ["string"
]
]
 }
 }
}
```

```
 "DocumentType": "string",
 "S3Uri": "string",
 "SourceDocumentsS3Uri": "string",
 "Split": "string"
 }
],
"DataFormat": "string",
"LabelDelimiter": "string",
"S3Uri": "string",
"TestS3Uri": "string"
},
"LanguageCode": "string",
"Message": "string",
"Mode": "string",
"ModelKmsKeyId": "string",
"OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
},
"Status": "string",
"SubmitTime": number,
"TrainingEndTime": number,
"TrainingStartTime": number,
"VersionName": "string",
"VolumeKmsKeyId": "string",
"VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
}
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### **DocumentClassifierProperties** (p. 263)

An object that contains the properties associated with a document classifier.

Type: [DocumentClassifierProperties](#) (p. 431) object

## Errors

For information about the errors that are common to all actions, see [Common Errors](#) (p. 503).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### TooManyRequestsException

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeDominantLanguageDetectionJob

Gets the properties associated with a dominant language detection job. Use this operation to get the status of a detection job.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### JobId (p. 266)

The identifier that Amazon Comprehend generated for the job. The [StartDominantLanguageDetectionJob \(p. 354\)](#) operation returns this identifier in its response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/:+\\-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "DominantLanguageDetectionJobProperties": {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 }
```

```
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [DominantLanguageDetectionJobProperties \(p. 266\)](#)

An object that contains the properties associated with a dominant language detection job.

Type: [DominantLanguageDetectionJobProperties \(p. 441\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)

- [AWS SDK for Ruby V3](#)

## DescribeEndpoint

Gets the properties associated with a specific endpoint. Use this operation to get the status of an endpoint.

### Request Syntax

```
{
 "EndpointArn": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### EndpointArn (p. 269)

The Amazon Resource Number (ARN) of the endpoint being described.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier-endpoint|entity-recognizer-endpoint)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

Required: Yes

### Response Syntax

```
{
 "EndpointProperties": {
 "CreationTime": number,
 "CurrentInferenceUnits": number,
 "DataAccessRoleArn": "string",
 "DesiredDataAccessRoleArn": "string",
 "DesiredInferenceUnits": number,
 "DesiredModelArn": "string",
 "EndpointArn": "string",
 "LastModifiedTime": number,
 "Message": "string",
 "ModelArn": "string",
 "Status": "string"
 }
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### EndpointProperties (p. 269)

Describes information associated with the specific endpoint.

Type: [EndpointProperties \(p. 445\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeEntitiesDetectionJob

Gets the properties associated with an entities detection job. Use this operation to get the status of a detection job.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### JobId (p. 271)

The identifier that Amazon Comprehend generated for the job. The [StartEntitiesDetectionJob \(p. 359\)](#) operation returns this identifier in its response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+=\-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "EntitiesDetectionJobProperties": {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "EntityRecognizerArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "StartTime": number
 }
}
```

```
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [EntitiesDetectionJobProperties \(p. 271\)](#)

An object that contains the properties associated with an entities detection job.

Type: [EntitiesDetectionJobProperties \(p. 449\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeEntityRecognizer

Provides details about an entity recognizer including status, S3 buckets containing training data, recognizer metadata, metrics, and so on.

### Request Syntax

```
{
 "EntityRecognizerArn": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### EntityRecognizerArn (p. 274)

The Amazon Resource Name (ARN) that identifies the entity recognizer.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:entity-recognizer/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: Yes

### Response Syntax

```
{
 "EntityRecognizerProperties": {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "EntityRecognizerArn": "string",
 "InputDataConfig": {
 "Annotations": {
 "S3Uri": "string",
 "TestS3Uri": "string"
 },
 "AugmentedManifests": [
 {
 "AnnotationDataS3Uri": "string",
 "AttributeNames": ["string"],
 "DocumentType": "string",
 "S3Uri": "string",
 "SourceDocumentsS3Uri": "string",
 "Split": "string"
 }
],
 "DataFormat": "string",
 "Documents": {
 "InputFormat": "string",
 "S3Uri": "string",
 }
 }
 }
}
```

```
 "TestS3Uri": "string"
 },
 "EntityList": {
 "S3Uri": "string"
 },
 "EntityTypes": [
 {
 "Type": "string"
 }
],
 "LanguageCode": "string",
 "Message": "string",
 "ModelKnsKeyId": "string",
 "RecognizerMetadata": {
 "EntityTypes": [
 {
 "EvaluationMetrics": {
 "F1Score": number,
 "Precision": number,
 "Recall": number
 },
 "NumberOfTrainMentions": number,
 "Type": "string"
 }
],
 "EvaluationMetrics": {
 "F1Score": number,
 "Precision": number,
 "Recall": number
 },
 "NumberOfTestDocuments": number,
 "NumberOfTrainedDocuments": number
 },
 "Status": "string",
 "SubmitTime": number,
 "TrainingEndTime": number,
 "TrainingStartTime": number,
 "VersionName": "string",
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [EntityRecognizerProperties \(p. 274\)](#)

Describes information associated with an entity recognizer.

Type: [EntityRecognizerProperties \(p. 464\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeEventsDetectionJob

Gets the status and details of an events detection job.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **JobId** (p. 277)

The identifier of the events detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/:=+-\%@]\*\$

Required: Yes

### Response Syntax

```
{
 "EventsDetectionJobProperties": {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "TargetEventTypes": ["string"]
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [EventsDetectionJobProperties \(p. 277\)](#)

An object that contains the properties associated with an event detection job.

Type: [EventsDetectionJobProperties \(p. 472\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeKeyPhrasesDetectionJob

Gets the properties associated with a key phrases detection job. Use this operation to get the status of a detection job.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### JobId (p. 279)

The identifier that Amazon Comprehend generated for the job. The [StartKeyPhrasesDetectionJob \(p. 368\)](#) operation returns this identifier in its response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:+\\-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "KeyPhrasesDetectionJobProperties": {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 }
}
```

```
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [KeyPhrasesDetectionJobProperties \(p. 279\)](#)

An object that contains the properties associated with a key phrases detection job.

Type: [KeyPhrasesDetectionJobProperties \(p. 479\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribePiiEntitiesDetectionJob

Gets the properties associated with a PII entities detection job. For example, you can use this operation to get the job status.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### JobId (p. 282)

The identifier that Amazon Comprehend generated for the job. The [StartPiiEntitiesDetectionJob \(p. 373\)](#) operation returns this identifier in its response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:+\\-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "PiiEntitiesDetectionJobProperties": {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "Message": "string",
 "Mode": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "RedactionConfig": {
 }
```

```
 "MaskCharacter": "string",
 "MaskMode": "string",
 "PiiEntityTypes": ["string"]
 },
 "SubmitTime": number
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [PiiEntitiesDetectionJobProperties \(p. 282\)](#)

Provides information about a PII entities detection job.

Type: [PiiEntitiesDetectionJobProperties \(p. 485\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeSentimentDetectionJob

Gets the properties associated with a sentiment detection job. Use this operation to get the status of a detection job.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### JobId (p. 285)

The identifier that Amazon Comprehend generated for the job. The [StartSentimentDetectionJob \(p. 377\)](#) operation returns this identifier in its response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:+\\-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "SentimentDetectionJobProperties": {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 }
}
```

```
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [SentimentDetectionJobProperties \(p. 285\)](#)

An object that contains the properties associated with a sentiment detection job.

Type: [SentimentDetectionJobProperties \(p. 493\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeTopicsDetectionJob

Gets the properties associated with a topic detection job. Use this operation to get the status of a detection job.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### JobId (p. 288)

The identifier assigned by the user to the detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+=\-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "TopicsDetectionJobProperties": {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "Message": "string",
 "NumberOfTopics": number,
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 }
```

```
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [TopicsDetectionJobProperties \(p. 288\)](#)

The list of properties for the requested job.

Type: [TopicsDetectionJobProperties \(p. 500\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)

- [AWS SDK for Ruby V3](#)

## DetectDominantLanguage

Determines the dominant language of the input text. For a list of languages that Amazon Comprehend can detect, see [Amazon Comprehend Supported Languages](#).

### Request Syntax

```
{
 "Text": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Text \(p. 291\)](#)

A UTF-8 text string. Each string should contain at least 20 characters and must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "Languages": [
 {
 "LanguageCode": "string",
 "Score": number
 }
]
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [Languages \(p. 291\)](#)

The languages that Amazon Comprehend detected in the input text. For each language, the response returns the RFC 5646 language code and the level of confidence that Amazon Comprehend has in the accuracy of its inference. For more information about RFC 5646, see [Tags for Identifying Languages](#) on the [IETF Tools](#) web site.

Type: Array of [DominantLanguage \(p. 439\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

## Examples

### Detect dominant language

If the input text is "Bob lives in Seattle. He is a software engineer at Amazon.", the operation returns the following:

```
{
 "Languages": [
 {
 "LanguageCode": "en",
 "Score": 0.9774383902549744
 },
 {
 "LanguageCode": "de",
 "Score": 0.010717987082898617
 }
]
}
```

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)



## DetectEntities

Inspects text for named entities, and returns information about them. For more information, about named entities, see [Detect Entities \(p. 94\)](#).

### Request Syntax

```
{
 "EndpointArn": "string",
 "LanguageCode": "string",
 "Text": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [EndpointArn \(p. 294\)](#)

The Amazon Resource Name of an endpoint that is associated with a custom entity recognition model. Provide an endpoint if you want to detect entities by using your own custom model instead of the default model that is used by Amazon Comprehend.

If you specify an endpoint, Amazon Comprehend uses the language of your custom model, and it ignores any language code that you provide in your request.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:entity-recognizer-endpoint/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

Required: No

#### [LanguageCode \(p. 294\)](#)

The language of the input documents. You can specify any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

If your request includes the endpoint for a custom entity recognition model, Amazon Comprehend uses the language of your custom model, and it ignores any language code that you specify here.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: No

#### [Text \(p. 294\)](#)

A UTF-8 text string. Each string must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

## Response Syntax

```
{
 "Entities": [
 {
 "BeginOffset": number,
 "EndOffset": number,
 "Score": number,
 "Text": "string",
 "Type": "string"
 }
]
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### **Entities** ([p. 295](#))

A collection of entities identified in the input text. For each entity, the response provides the entity text, entity type, where the entity text begins and ends, and the level of confidence that Amazon Comprehend has in the detection.

If your request uses a custom entity recognition model, Amazon Comprehend detects the entities that the model is trained to recognize. Otherwise, it detects the default entity types. For a list of default entity types, see [Detect Entities \(p. 94\)](#).

Type: Array of [Entity \(p. 452\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceUnavailableException**

The specified resource is not available. Check the resource and try your request again.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### UnsupportedLanguageException

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## Examples

### Detect entities

If the input text is "Bob ordered two sandwiches and three ice cream cones today from a store in Seattle.", the operation returns the following:

```
{
 "Entities": [
 {
 "Text": "Bob",
 "Score": 1.0,
 "Type": "PERSON",
 "BeginOffset": 0,
 "EndOffset": 3
 },
 {
 "Text": "two",
 "Score": 1.0,
 "Type": "QUANTITY",
 "BeginOffset": 12,
 "EndOffset": 15
 },
 {
 "Text": "three",
 "Score": 1.0,
 "Type": "QUANTITY",
 "BeginOffset": 32,
 "EndOffset": 37
 },
 {
 "Text": "Today",
 "Score": 1.0,
 "Type": "DATE",
 "BeginOffset": 54,
 "EndOffset": 59
 },
 {
 "Text": "Seattle",
 "Score": 1.0,
 "Type": "LOCATION",
 "BeginOffset": 76,
 "EndOffset": 83
 }
]
}
```

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DetectKeyPhrases

Detects the key noun phrases found in the text.

### Request Syntax

```
{
 "LanguageCode": "string",
 "Text": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [LanguageCode \(p. 298\)](#)

The language of the input documents. You can specify any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: Yes

#### [Text \(p. 298\)](#)

A UTF-8 text string. Each string must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "KeyPhrases": [
 {
 "BeginOffset": number,
 "EndOffset": number,
 "Score": number,
 "Text": "string"
 }
]
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

## KeyPhrases (p. 298)

A collection of key phrases that Amazon Comprehend identified in the input text. For each key phrase, the response provides the text of the key phrase, where the key phrase begins and ends, and the level of confidence that Amazon Comprehend has in the accuracy of the detection.

Type: Array of [KeyPhrase \(p. 477\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### InternalServerError

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### InvalidRequestException

The request is invalid.

HTTP Status Code: 400

### TextSizeLimitExceededException

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### UnsupportedLanguageException

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## Examples

### Detect phrases

If the input text is "Bob lives in Seattle. He is a software engineer at Amazon.", the API returns the following:

```
{
 "KeyPhrases": [
 {
 "Text": "Bob",
 "Score": 1.0,
 "BeginOffset": 0,
 "EndOffset": 3
 },
 {
 "Text": "Seattle",
 "Score": 1.0,
 "BeginOffset": 13,
 "EndOffset": 20
 }
]
}
```

```
 "Text": "a software engineer",
 "Score": 1.0,
 "BeginOffset": 28,
 "EndOffset": 39
 },
 {
 "Text": "Amazon",
 "Score": 1.0,
 "BeginOffset": 43,
 "EndOffset": 49
 }
}]
```

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DetectPiiEntities

Inspects the input text for entities that contain personally identifiable information (PII) and returns information about them.

### Request Syntax

```
{
 "LanguageCode": "string",
 "Text": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [LanguageCode \(p. 301\)](#)

The language of the input documents.

Type: String

Valid Values: en

Required: Yes

#### [Text \(p. 301\)](#)

A UTF-8 text string. Each string must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "Entities": [
 {
 "BeginOffset": number,
 "EndOffset": number,
 "Score": number,
 "Type": "string"
 }
]
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [Entities \(p. 301\)](#)

A collection of PII entities identified in the input text. For each entity, the response provides the entity type, where the entity text begins and ends, and the level of confidence that Amazon Comprehend has in the detection.

Type: Array of [PiiEntity \(p. 488\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### **UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DetectSentiment

Inspects text and returns an inference of the prevailing sentiment (POSITIVE, NEUTRAL, MIXED, or NEGATIVE).

### Request Syntax

```
{
 "LanguageCode": "string",
 "Text": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [LanguageCode \(p. 303\)](#)

The language of the input documents. You can specify any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: Yes

#### [Text \(p. 303\)](#)

A UTF-8 text string. Each string must contain fewer than 5,000 bytes of UTF-8 encoded characters.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "Sentiment": "string",
 "SentimentScore": {
 "Mixed": number,
 "Negative": number,
 "Neutral": number,
 "Positive": number
 }
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [Sentiment \(p. 303\)](#)

The inferred sentiment that Amazon Comprehend has the highest level of confidence in.

Type: String

Valid Values: POSITIVE | NEGATIVE | NEUTRAL | MIXED

### [SentimentScore \(p. 303\)](#)

An object that lists the sentiments, and their corresponding confidence levels.

Type: [SentimentScore \(p. 496\)](#) object

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### **UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## Examples

### Detect sentiment

If the input text is "Today is my birthday, I am so happy.", the operation returns the following response:

```
{
 "SentimentScore": {
 "Mixed": 0.0033542951568961143,
 "Positive": 0.9869875907897949,
 "Neutral": 0.008563132025301456,
 "Negative": 0.0010949420975521207
 },
 "Sentiment": "POSITIVE",
}
```

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DetectSyntax

Inspects text for syntax and the part of speech of words in the document. For more information, [Analyze Syntax \(p. 112\)](#).

### Request Syntax

```
{
 "LanguageCode": "string",
 "Text": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [LanguageCode \(p. 306\)](#)

The language code of the input documents. You can specify any of the following languages supported by Amazon Comprehend: German ("de"), English ("en"), Spanish ("es"), French ("fr"), Italian ("it"), or Portuguese ("pt").

Type: String

Valid Values: en | es | fr | de | it | pt

Required: Yes

#### [Text \(p. 306\)](#)

A UTF-8 string. Each string must contain fewer than 5,000 bytes of UTF encoded characters.

Type: String

Length Constraints: Minimum length of 1.

Required: Yes

### Response Syntax

```
{
 "SyntaxTokens": [
 {
 "BeginOffset": number,
 "EndOffset": number,
 "PartOfSpeech": {
 "Score": number,
 "Tag": "string"
 },
 "Text": "string",
 "TokenId": number
 }
]
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### **SyntaxTokens (p. 306)**

A collection of syntax tokens describing the text. For each token, the response provides the text, the token type, where the text begins and ends, and the level of confidence that Amazon Comprehend has that the token is correct. For a list of token types, see [Analyze Syntax \(p. 112\)](#).

Type: Array of [SyntaxToken \(p. 497\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TextSizeLimitExceededException**

The size of the input text exceeds the limit. Use a smaller document.

HTTP Status Code: 400

### **UnsupportedLanguageException**

Amazon Comprehend can't process the language of the input text. For custom entity recognition APIs, only English, Spanish, French, Italian, German, or Portuguese are accepted. For a list of supported languages, see [Languages Supported in Amazon Comprehend \(p. 5\)](#).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)



## ListDocumentClassificationJobs

Gets a list of the documentation classification jobs that you have submitted.

### Request Syntax

```
{
 "Filter": {
 "JobName": "string",
 "JobStatus": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Filter \(p. 309\)](#)

Filters the jobs that are returned. You can filter jobs on their names, status, or the date and time that they were submitted. You can only set one filter at a time.

Type: [DocumentClassificationJobFilter \(p. 423\)](#) object

Required: No

#### [MaxResults \(p. 309\)](#)

The maximum number of results to return in each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 309\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "DocumentClassificationJobPropertiesList": [
 {
 "DataAccessRoleArn": "string",
 "DocumentClassifierArn": "string",
 "DocumentType": "string",
 "FailureReason": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LastModified": "string",
 "SubmitTime": "string",
 "StatusDetails": {
 "Status": "string",
 "StatusReason": "string"
 }
 }
]
}
```

```
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
},
],
"NextToken": "string
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [DocumentClassificationJobPropertiesList \(p. 309\)](#)

A list containing the properties of each job returned.

Type: Array of [DocumentClassificationJobProperties \(p. 424\)](#) objects

### [NextToken \(p. 309\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

**InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

**TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListDocumentClassifiers

Gets a list of the document classifiers that you have created.

### Request Syntax

```
{
 "Filter": {
 "DocumentClassifierName": "string",
 "Status": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Filter \(p. 312\)](#)

Filters the jobs that are returned. You can filter jobs on their name, status, or the date and time that they were submitted. You can only set one filter at a time.

Type: [DocumentClassifierFilter \(p. 427\)](#) object

Required: No

#### [MaxResults \(p. 312\)](#)

The maximum number of results to return in each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 312\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "DocumentClassifierPropertiesList": [
 {
 "ClassifierMetadata": {
 "EvaluationMetrics": {
 "Metric": "string",
 "Value": number
 }
 }
 }
]
}
```

```

 "Accuracy": number,
 "F1Score": number,
 "HammingLoss": number,
 "MicroF1Score": number,
 "MicroPrecision": number,
 "MicroRecall": number,
 "Precision": number,
 "Recall": number
 },
 "NumberOfLabels": number,
 "NumberOfTestDocuments": number,
 "NumberOfTrainedDocuments": number
},
"DataAccessRoleArn": "string",
"DocumentClassifierArn": "string",
"EndTime": number,
"InputDataConfig": {
 "AugmentedManifests": [
 {
 "AnnotationDataS3Uri": "string",
 "AttributeNames": ["string"],
 "DocumentType": "string",
 "S3Uri": "string",
 "SourceDocumentsS3Uri": "string",
 "Split": "string"
 }
],
 "DataFormat": "string",
 "LabelDelimiter": "string",
 "S3Uri": "string",
 "TestS3Uri": "string"
},
"LanguageCode": "string",
"Message": "string",
"Mode": "string",
"ModelKmsKeyId": "string",
"OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
},
"Status": "string",
"SubmitTime": number,
"TrainingEndTime": number,
"TrainingStartTime": number,
"VersionName": "string",
"VolumeKmsKeyId": "string",
"VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
}
},
],
"NextToken": "string"
}

```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [DocumentClassifierPropertiesList \(p. 312\)](#)

A list containing the properties of each job returned.

Type: Array of [DocumentClassifierProperties \(p. 431\)](#) objects  
**NextToken (p. 312)**

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListDocumentClassifierSummaries

Gets a list of summaries of the document classifiers that you have created

### Request Syntax

```
{
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [MaxResults \(p. 315\)](#)

The maximum number of results to return on each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 315\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "DocumentClassifierSummariesList": [
 {
 "DocumentClassifierName": "string",
 "LatestVersionCreatedAt": number,
 "LatestVersionName": "string",
 "LatestVersionStatus": "string",
 "NumberOfVersions": number
 }
],
 "NextToken": "string"
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [DocumentClassifierSummariesList \(p. 315\)](#)

The list of summaries of document classifiers.

Type: Array of [DocumentClassifierSummary \(p. 435\)](#) objects

### [NextToken \(p. 315\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListDominantLanguageDetectionJobs

Gets a list of the dominant language detection jobs that you have submitted.

### Request Syntax

```
{
 "Filter": {
 "JobName": "string",
 "JobStatus": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Filter \(p. 317\)](#)

Filters that jobs that are returned. You can filter jobs on their name, status, or the date and time that they were submitted. You can only set one filter at a time.

Type: [DominantLanguageDetectionJobFilter \(p. 440\)](#) object

Required: No

#### [MaxResults \(p. 317\)](#)

The maximum number of results to return in each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 317\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "DominantLanguageDetectionJobPropertiesList": [
 {
 "DataAccessRoleArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "SubmitTime": number
 }
]
}
```

```
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
 }
],
"NextToken": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [DominantLanguageDetectionJobPropertiesList \(p. 317\)](#)

A list containing the properties of each job that is returned.

Type: Array of [DominantLanguageDetectionJobProperties \(p. 441\)](#) objects

### [NextToken \(p. 317\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

**InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

**TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListEndpoints

Gets a list of all existing endpoints that you've created.

### Request Syntax

```
{
 "Filter": {
 "CreationTimeAfter": number,
 "CreationTimeBefore": number,
 "ModelArn": "string",
 "Status": "string"
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **Filter (p. 320)**

Filters the endpoints that are returned. You can filter endpoints on their name, model, status, or the date and time that they were created. You can only set one filter at a time.

Type: [EndpointFilter \(p. 444\)](#) object

Required: No

#### **MaxResults (p. 320)**

The maximum number of results to return in each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### **NextToken (p. 320)**

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "EndpointPropertiesList": [
 {
 "CreationTime": number,
 "LastModifiedTime": number,
 "ModelArn": "string",
 "Name": "string",
 "Status": "string"
 }
]
}
```

```
 "CurrentInferenceUnits": number,
 "DataAccessRoleArn": "string",
 "DesiredDataAccessRoleArn": "string",
 "DesiredInferenceUnits": number,
 "DesiredModelArn": "string",
 "EndpointArn": "string",
 "LastModifiedTime": number,
 "Message": "string",
 "ModelArn": "string",
 "Status": "string"
 }
],
"NextToken": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [EndpointPropertiesList \(p. 320\)](#)

Displays a list of endpoint properties being retrieved by the service in response to the request.

Type: Array of [EndpointProperties \(p. 445\)](#) objects

### [NextToken \(p. 320\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListEntitiesDetectionJobs

Gets a list of the entity detection jobs that you have submitted.

## Request Syntax

```
{
 "Filter": {
 "JobName": "string",
 "JobStatus": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

## Filter (p. 323)

Filters the jobs that are returned. You can filter jobs on their name, status, or the date and time that they were submitted. You can only set one filter at a time.

Type: EntitiesDetectionJobFilter (p. 448) object

Required: No

## MaxResults (p. 323)

The maximum number of results to return in each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

## NextToken (p. 323)

**Identifies the next page of results to return.**

Type: String

Length Constraints: Minimum length of 1.

Required: No

## Response Syntax

```
{
 "EntitiesDetectionJobPropertiesList": [
 {
```

```
"DataAccessRoleArn": "string",
"EndTime": number,
"EntityRecognizerArn": "string",
"InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
},
"JobArn": "string",
"JobId": "string",
"JobName": "string",
"JobStatus": "string",
"LanguageCode": "string",
"Message": "string",
"OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
},
"SubmitTime": number,
"VolumeKmsKeyId": "string",
"VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
}
},
"NextToken": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [EntitiesDetectionJobPropertiesList \(p. 323\)](#)

A list containing the properties of each job that is returned.

Type: Array of [EntitiesDetectionJobProperties \(p. 449\)](#) objects

### [NextToken \(p. 323\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListEntityRecognizers

Gets a list of the properties of all entity recognizers that you created, including recognizers currently in training. Allows you to filter the list of recognizers based on criteria such as status and submission time. This call returns up to 500 entity recognizers in the list, with a default number of 100 recognizers in the list.

The results of this list are not in any particular order. Please get the list and sort locally if needed.

### Request Syntax

```
{
 "Filter": {
 "RecognizerName": "string",
 "Status": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Filter \(p. 326\)](#)

Filters the list of entities returned. You can filter on `Status`, `SubmitTimeBefore`, or `SubmitTimeAfter`. You can only set one filter at a time.

Type: [EntityRecognizerFilter \(p. 459\)](#) object

Required: No

#### [MaxResults \(p. 326\)](#)

The maximum number of results to return on each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 326\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
```

```

"EntityRecognizerPropertiesList": [
 {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "EntityRecognizerArn": "string",
 "InputDataConfig": {
 "Annotations": {
 "S3Uri": "string",
 "TestS3Uri": "string"
 },
 "AugmentedManifests": [
 {
 "AnnotationDataS3Uri": "string",
 "AttributeNames": ["string"],
 "DocumentType": "string",
 "S3Uri": "string",
 "SourceDocumentsS3Uri": "string",
 "Split": "string"
 }
],
 "DataFormat": "string",
 "Documents": {
 "InputFormat": "string",
 "S3Uri": "string",
 "TestS3Uri": "string"
 },
 "EntityList": {
 "S3Uri": "string"
 },
 "EntityTypes": [
 {
 "Type": "string"
 }
]
 },
 "LanguageCode": "string",
 "Message": "string",
 "ModelKmsKeyId": "string",
 "RecognizerMetadata": {
 "EntityTypes": [
 {
 "EvaluationMetrics": {
 "F1Score": number,
 "Precision": number,
 "Recall": number
 },
 "NumberOfTrainMentions": number,
 "Type": "string"
 }
],
 "EvaluationMetrics": {
 "F1Score": number,
 "Precision": number,
 "Recall": number
 },
 "NumberOfTestDocuments": number,
 "NumberOfTrainedDocuments": number
 },
 "Status": "string",
 "SubmitTime": number,
 "TrainingEndTime": number,
 "TrainingStartTime": number,
 "VersionName": "string",
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],

```

```
 "Subnets": ["string"]
 }
],
"NextToken": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### **EntityRecognizerPropertiesList** (p. 326)

The list of properties of an entity recognizer.

Type: Array of [EntityRecognizerProperties](#) (p. 464) objects

### **NextToken** (p. 326)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListEntityRecognizerSummaries

Gets a list of summaries for the entity recognizers that you have created.

### Request Syntax

```
{
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [MaxResults \(p. 330\)](#)

The maximum number of results to return on each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 330\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "EntityRecognizerSummariesList": [
 {
 "LatestVersionCreatedAt": number,
 "LatestVersionName": "string",
 "LatestVersionStatus": "string",
 "NumberOfVersions": number,
 "RecognizerName": "string"
 }
],
 "NextToken": "string"
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [EntityRecognizerSummariesList \(p. 330\)](#)

The list entity recognizer summaries.

Type: Array of [EntityRecognizerSummary \(p. 467\)](#) objects

### [NextToken \(p. 330\)](#)

The list entity recognizer summaries.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListEventsDetectionJobs

Gets a list of the events detection jobs that you have submitted.

### Request Syntax

```
{
 "Filter": {
 "JobName": "string",
 "JobStatus": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Filter \(p. 332\)](#)

Filters the jobs that are returned. You can filter jobs on their name, status, or the date and time that they were submitted. You can only set one filter at a time.

Type: [EventsDetectionJobFilter \(p. 471\)](#) object

Required: No

#### [MaxResults \(p. 332\)](#)

The maximum number of results to return in each page.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 332\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "EventsDetectionJobPropertiesList": [
 {
 "DataAccessRoleArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LastModified": "string",
 "SubmitTime": "string"
 }
]
}
```

```
"EndTime": number,
"InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
},
"JobArn": "string",
"JobId": "string",
"JobName": "string",
"JobStatus": "string",
"LanguageCode": "string",
"Message": "string",
"OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
},
"SubmitTime": number,
"TargetEventTypes": ["string"]
},
],
"NextToken": "string
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [EventsDetectionJobPropertiesList \(p. 332\)](#)

A list containing the properties of each job that is returned.

Type: Array of [EventsDetectionJobProperties \(p. 472\)](#) objects

### [NextToken \(p. 332\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListKeyPhrasesDetectionJobs

Get a list of key phrase detection jobs that you have submitted.

### Request Syntax

```
{
 "Filter": {
 "JobName": "string",
 "JobStatus": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Filter \(p. 335\)](#)

Filters the jobs that are returned. You can filter jobs on their name, status, or the date and time that they were submitted. You can only set one filter at a time.

Type: [KeyPhrasesDetectionJobFilter \(p. 478\)](#) object

Required: No

#### [MaxResults \(p. 335\)](#)

The maximum number of results to return in each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 335\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "KeyPhrasesDetectionJobPropertiesList": [
 {
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "SubmitTime": number
 }
]
}
```

```
"InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
},
"JobArn": "string",
"JobId": "string",
"JobName": "string",
"JobStatus": "string",
"LanguageCode": "string",
"Message": "string",
"OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
},
"SubmitTime": number,
"VolumeKmsKeyId": "string",
"VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
}
}
],
"NextToken": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [KeyPhrasesDetectionJobPropertiesList \(p. 335\)](#)

A list containing the properties of each job that is returned.

Type: Array of [KeyPhrasesDetectionJobProperties \(p. 479\)](#) objects

### [NextToken \(p. 335\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

**InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

**TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListPiiEntitiesDetectionJobs

Gets a list of the PII entity detection jobs that you have submitted.

### Request Syntax

```
{
 "Filter": {
 "JobName": "string",
 "JobStatus": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **Filter (p. 338)**

Filters the jobs that are returned. You can filter jobs on their name, status, or the date and time that they were submitted. You can only set one filter at a time.

Type: [PiiEntitiesDetectionJobFilter \(p. 484\)](#) object

Required: No

#### **MaxResults (p. 338)**

The maximum number of results to return in each page.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### **NextToken (p. 338)**

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "NextToken": "string",
 "PiiEntitiesDetectionJobPropertiesList": [
 ...
]
}
```

```
{
 "DataAccessRoleArn": "string",
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "Message": "string",
 "Mode": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "RedactionConfig": {
 "MaskCharacter": "string",
 "MaskMode": "string",
 "PiiEntityTypes": ["string"]
 },
 "SubmitTime": number
}
]
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [NextToken \(p. 338\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

### [PiiEntitiesDetectionJobPropertiesList \(p. 338\)](#)

A list containing the properties of each job that is returned.

Type: Array of [PiiEntitiesDetectionJobProperties \(p. 485\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListSentimentDetectionJobs

Gets a list of sentiment detection jobs that you have submitted.

### Request Syntax

```
{
 "Filter": {
 "JobName": "string",
 "JobStatus": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Filter \(p. 341\)](#)

Filters the jobs that are returned. You can filter jobs on their name, status, or the date and time that they were submitted. You can only set one filter at a time.

Type: [SentimentDetectionJobFilter \(p. 492\)](#) object

Required: No

#### [MaxResults \(p. 341\)](#)

The maximum number of results to return in each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 341\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "NextToken": "string",
 "SentimentDetectionJobPropertiesList": [
 {
 "DataAccessRoleArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "SentimentType": "string",
 "SubmitTime": number
 }
]
}
```

```
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "LanguageCode": "string",
 "Message": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [NextToken \(p. 341\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

### [SentimentDetectionJobPropertiesList \(p. 341\)](#)

A list containing the properties of each job that is returned.

Type: Array of [SentimentDetectionJobProperties \(p. 493\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

**InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

**TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListTagsForResource

Lists all tags associated with a given Amazon Comprehend resource.

### Request Syntax

```
{
 "ResourceArn": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ResourceArn \(p. 344\)](#)

The Amazon Resource Name (ARN) of the given Amazon Comprehend resource you are querying.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[ ^:]\*)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: Yes

### Response Syntax

```
{
 "ResourceArn": "string",
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
]
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [ResourceArn \(p. 344\)](#)

The Amazon Resource Name (ARN) of the given Amazon Comprehend resource you are querying.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

### Tags (p. 344)

Tags associated with the Amazon Comprehend resource being queried. A tag is a key-value pair that adds as a metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### InternalServerError

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### InvalidRequestException

The request is invalid.

HTTP Status Code: 400

### ResourceNotFoundException

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## ListTopicsDetectionJobs

Gets a list of the topic detection jobs that you have submitted.

### Request Syntax

```
{
 "Filter": {
 "JobName": "string",
 "JobStatus": "string",
 "SubmitTimeAfter": number,
 "SubmitTimeBefore": number
 },
 "MaxResults": number,
 "NextToken": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [Filter \(p. 346\)](#)

Filters the jobs that are returned. Jobs can be filtered on their name, status, or the date and time that they were submitted. You can set only one filter at a time.

Type: [TopicsDetectionJobFilter \(p. 499\)](#) object

Required: No

#### [MaxResults \(p. 346\)](#)

The maximum number of results to return in each page. The default is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

#### [NextToken \(p. 346\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

Required: No

### Response Syntax

```
{
 "NextToken": "string",
 "TopicsDetectionJobPropertiesList": [
 {
 "DataAccessRoleArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "Language": "string",
 "SubmitTime": number
 }
]
}
```

```
 "EndTime": number,
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobArn": "string",
 "JobId": "string",
 "JobName": "string",
 "JobStatus": "string",
 "Message": "string",
 "NumberOfTopics": number,
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "SubmitTime": number,
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
 }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [NextToken \(p. 346\)](#)

Identifies the next page of results to return.

Type: String

Length Constraints: Minimum length of 1.

### [TopicsDetectionJobPropertiesList \(p. 346\)](#)

A list containing the properties of each job that is returned.

Type: Array of [TopicsDetectionJobProperties \(p. 500\)](#) objects

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidFilterException**

The filter specified for the operation is invalid. Specify a different filter.

HTTP Status Code: 400

**InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

**TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StartDocumentClassificationJob

Starts an asynchronous document classification job. Use the [DescribeDocumentClassificationJob \(p. 260\)](#) operation to track the progress of the job.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "DocumentClassifierArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobName": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **ClientRequestToken** ([p. 349](#))

A unique identifier for the request. If you do not set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

#### **DataAccessRoleArn** ([p. 349](#))

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?::iam::[0-9]{12}:role/.+

Required: Yes

#### [DocumentClassifierArn \(p. 349\)](#)

The Amazon Resource Name (ARN) of the document classifier to use to process the job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?::comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:document-classifier/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*?)?

Required: Yes

#### [InputDataConfig \(p. 349\)](#)

Specifies the format and location of the input data for the job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: Yes

#### [JobName \(p. 349\)](#)

The identifier of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:=+\-%@]\*\$

Required: No

#### [OutputDataConfig \(p. 349\)](#)

Specifies where to send the output files.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: Yes

#### [Tags \(p. 349\)](#)

Tags to be associated with the document classification job. A tag is a key-value pair that adds metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

#### [VolumeKmsKeyId \(p. 349\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### [VpcConfig \(p. 349\)](#)

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for your document classification job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## Response Syntax

```
{
 "JobArn": "string",
 "JobId": "string",
 "JobStatus": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [JobArn \(p. 351\)](#)

The Amazon Resource Name (ARN) of the document classification job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

`arn:<partition>:comprehend:<region>:<account-id>:document-classification-job/<job-id>`

The following is an example job ARN:

`arn:aws:comprehend:us-west-2:111122223333:document-classification-job/1234abcd12ab34cd56ef1234567890ab`

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*)?`

#### [JobId \(p. 351\)](#)

The identifier generated for the job. To get the status of the job, use this identifier with the [DescribeDocumentClassificationJob \(p. 260\)](#) operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-+\\-%@]\*\$

### **JobStatus (p. 351)**

The status of the job:

- SUBMITTED - The job has been received and queued for processing.
- IN\_PROGRESS - Amazon Comprehend is processing the job.
- COMPLETED - The job was successfully completed and the output is available.
- FAILED - The job did not complete. For details, use the [DescribeDocumentClassificationJob \(p. 260\)](#) operation.
- STOP\_REQUESTED - Amazon Comprehend has received a stop request for the job and is processing the request.
- STOPPED - The job was successfully stopped without completing.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **ResourceUnavailableException**

The specified resource is not available. Check the resource and try your request again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

### TooManyTagsException

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StartDominantLanguageDetectionJob

Starts an asynchronous dominant language detection job for a collection of documents. Use the [DescribeDominantLanguageDetectionJob \(p. 266\)](#) operation to track the status of a job.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobName": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ClientRequestToken \(p. 354\)](#)

A unique identifier for the request. If you do not set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

#### [DataAccessRoleArn \(p. 354\)](#)

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data. For more information, see <https://>

[docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions](https://docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions).

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: Yes

#### [InputDataConfig \(p. 354\)](#)

Specifies the format and location of the input data for the job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: Yes

#### [JobName \(p. 354\)](#)

An identifier for the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:=+\-%@]\*\$

Required: No

#### [OutputDataConfig \(p. 354\)](#)

Specifies where to send the output files.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: Yes

#### [Tags \(p. 354\)](#)

Tags to be associated with the dominant language detection job. A tag is a key-value pair that adds metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

#### [VolumeKmsKeyId \(p. 354\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

### **VpcConfig (p. 354)**

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for your dominant language detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## Response Syntax

```
{
 "JobArn": "string",
 "JobId": "string",
 "JobStatus": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### **JobArn (p. 356)**

The Amazon Resource Name (ARN) of the dominant language detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

```
arn:<partition>:comprehend:<region>:<account-id>:dominant-language-detection-job/<job-id>
```

The following is an example job ARN:

```
arn:aws:comprehend:us-west-2:111122223333:dominant-language-detection-job/1234abcd12ab34cd56ef1234567890ab
```

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

### **JobId (p. 356)**

The identifier generated for the job. To get the status of a job, use this identifier with the [DescribeDominantLanguageDetectionJob \(p. 266\)](#) operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+=\-%@]\*)\$

### **JobStatus (p. 356)**

The status of the job.

- SUBMITTED - The job has been received and is queued for processing.

- IN\_PROGRESS - Amazon Comprehend is processing the job.
- COMPLETED - The job was successfully completed and the output is available.
- FAILED - The job did not complete. To get details, use the [DescribeDominantLanguageDetectionJob \(p. 266\)](#) operation.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### InternalServerError

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### InvalidRequestException

The request is invalid.

HTTP Status Code: 400

### KmsKeyValidationException

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

### TooManyRequestsException

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

### TooManyTagsException

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)



## StartEntitiesDetectionJob

Starts an asynchronous entity detection job for a collection of documents. Use the [DescribeEntitiesDetectionJob \(p. 271\)](#) operation to track the status of a job.

This API can be used for either standard entity detection or custom entity recognition. In order to be used for custom entity recognition, the optional `EntityRecognizerArn` must be used in order to provide access to the recognizer being used to detect the custom entity.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "EntityRecognizerArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobName": "string",
 "LanguageCode": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **ClientRequestToken (p. 359)**

A unique identifier for the request. If you don't set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

#### [DataAccessRoleArn \(p. 359\)](#)

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data. For more information, see <https://docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions>.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?::iam::[0-9]{12}:role/.+

Required: Yes

#### [EntityRecognizerArn \(p. 359\)](#)

The Amazon Resource Name (ARN) that identifies the specific entity recognizer to be used by the StartEntitiesDetectionJob. This ARN is optional and is only used for a custom entity recognition job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?::comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:entity-recognizer/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

#### [InputDataConfig \(p. 359\)](#)

Specifies the format and location of the input data for the job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: Yes

#### [JobName \(p. 359\)](#)

The identifier of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/=+\-%@]\*)\$

Required: No

#### [LanguageCode \(p. 359\)](#)

The language of the input documents. All documents must be in the same language. You can specify any of the languages supported by Amazon Comprehend. If custom entities recognition is used, this parameter is ignored and the language used for training the model is used instead.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: Yes

#### [OutputDataConfig \(p. 359\)](#)

Specifies where to send the output files.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: Yes

#### [Tags \(p. 359\)](#)

Tags to be associated with the entities detection job. A tag is a key-value pair that adds metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

#### [VolumeKmsKeyId \(p. 359\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### [VpcConfig \(p. 359\)](#)

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for your entity detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## Response Syntax

```
{
 "JobArn": "string",
 "JobId": "string",
 "JobStatus": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [JobArn \(p. 361\)](#)

The Amazon Resource Name (ARN) of the entities detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

`arn:<partition>:comprehend:<region>:<account-id>:entities-detection-job/<job-id>`

The following is an example job ARN:

`arn:aws:comprehend:us-west-2:111122223333:entities-detection-job/1234abcd12ab34cd56ef1234567890ab`

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^\:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*)?`

### **JobId (p. 361)**

The identifier generated for the job. To get the status of job, use this identifier with the [DescribeEntitiesDetectionJob \(p. 271\)](#) operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: `^([\p{L}\p{Z}]\p{N}_.:/:=+\-%@]*$`

### **JobStatus (p. 361)**

The status of the job.

- SUBMITTED - The job has been received and is queued for processing.
- IN\_PROGRESS - Amazon Comprehend is processing the job.
- COMPLETED - The job was successfully completed and the output is available.
- FAILED - The job did not complete. To get details, use the [DescribeEntitiesDetectionJob \(p. 271\)](#) operation.
- STOP\_REQUESTED - Amazon Comprehend has received a stop request for the job and is processing the request.
- STOPPED - The job was successfully stopped without completing.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

**ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

**ResourceUnavailableException**

The specified resource is not available. Check the resource and try your request again.

HTTP Status Code: 400

**TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

**TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StartEventsDetectionJob

Starts an asynchronous event detection job for a collection of documents.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobName": "string",
 "LanguageCode": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "TargetEventTypes": ["string"]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ClientRequestToken \(p. 364\)](#)

An unique identifier for the request. If you don't set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

#### [DataAccessRoleArn \(p. 364\)](#)

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: Yes

#### [InputDataConfig \(p. 364\)](#)

Specifies the format and location of the input data for the job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: Yes

#### [JobName \(p. 364\)](#)

The identifier of the events detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/=+\-%@]\*\$

Required: No

#### [LanguageCode \(p. 364\)](#)

The language code of the input documents.

Type: String

Valid Values: en

Required: Yes

#### [OutputDataConfig \(p. 364\)](#)

Specifies where to send the output files.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: Yes

#### [Tags \(p. 364\)](#)

Tags to be associated with the events detection job. A tag is a key-value pair that adds metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

#### [TargetEventTypes \(p. 364\)](#)

The types of events to detect in the input documents.

Type: Array of strings

Array Members: Minimum number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: [A-Z\_]\*

Required: Yes

## Response Syntax

```
{
 "JobArn": "string",
 "JobId": "string",
 "JobStatus": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### **JobArn** (p. 366)

The Amazon Resource Name (ARN) of the events detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

`arn:<partition>:comprehend:<region>:<account-id>:events-detection-job/<job-id>`

The following is an example job ARN:

`arn:aws:comprehend:us-west-2:111122223333:events-detection-job/1234abcd12ab34cd56ef1234567890ab`

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^\:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*)?`

### **JobId** (p. 366)

An unique identifier for the request. If you don't set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: `^([\\p{L}\\p{Z}]\\p{N}_.:+/-%@]*$`

### **JobStatus** (p. 366)

The status of the events detection job.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

**InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

**KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

**TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

**TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StartKeyPhrasesDetectionJob

Starts an asynchronous key phrase detection job for a collection of documents. Use the [DescribeKeyPhrasesDetectionJob \(p. 279\)](#) operation to track the status of a job.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobName": "string",
 "LanguageCode": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ClientRequestToken \(p. 368\)](#)

A unique identifier for the request. If you don't set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

#### [DataAccessRoleArn \(p. 368\)](#)

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data. For more information, see <https://>

[docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions](https://docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions).

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: Yes

#### [InputDataConfig \(p. 368\)](#)

Specifies the format and location of the input data for the job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: Yes

#### [JobName \(p. 368\)](#)

The identifier of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

Required: No

#### [LanguageCode \(p. 368\)](#)

The language of the input documents. You can specify any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: Yes

#### [OutputDataConfig \(p. 368\)](#)

Specifies where to send the output files.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: Yes

#### [Tags \(p. 368\)](#)

Tags to be associated with the key phrases detection job. A tag is a key-value pair that adds metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

#### [VolumeKmsKeyId \(p. 368\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### [VpcConfig \(p. 368\)](#)

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for your key phrases detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## Response Syntax

```
{
 "JobArn": "string",
 "JobId": "string",
 "JobStatus": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [JobArn \(p. 370\)](#)

The Amazon Resource Name (ARN) of the key phrase detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

```
arn:<partition>:comprehend:<region>:<account-id>:key-phrases-detection-job/
<job-id>
```

The following is an example job ARN:

```
arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-
job/1234abcd12ab34cd56ef1234567890ab
```

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]  
{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*?)?

#### [JobId \(p. 370\)](#)

The identifier generated for the job. To get the status of a job, use this identifier with the [DescribeKeyPhrasesDetectionJob \(p. 279\)](#) operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

#### [JobStatus \(p. 370\)](#)

The status of the job.

- SUBMITTED - The job has been received and is queued for processing.
- IN\_PROGRESS - Amazon Comprehend is processing the job.
- COMPLETED - The job was successfully completed and the output is available.
- FAILED - The job did not complete. To get details, use the [DescribeKeyPhrasesDetectionJob \(p. 279\)](#) operation.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

### **TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)

- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StartPiiEntitiesDetectionJob

Starts an asynchronous PII entity detection job for a collection of documents.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobName": "string",
 "LanguageCode": "string",
 "Mode": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "RedactionConfig": {
 "MaskCharacter": "string",
 "MaskMode": "string",
 "PiiEntityTypes": ["string"]
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ClientRequestToken \(p. 373\)](#)

A unique identifier for the request. If you don't set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

#### [DataAccessRoleArn \(p. 373\)](#)

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: Yes

#### [InputDataConfig \(p. 373\)](#)

The input properties for a PII entities detection job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: Yes

#### [JobName \(p. 373\)](#)

The identifier of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+=\-%@]\*\$

Required: No

#### [LanguageCode \(p. 373\)](#)

The language of the input documents.

Type: String

Valid Values: en

Required: Yes

#### [Mode \(p. 373\)](#)

Specifies whether the output provides the locations (offsets) of PII entities or a file in which PII entities are redacted.

Type: String

Valid Values: ONLY\_REDACTION | ONLY\_OFFSETS

Required: Yes

#### [OutputDataConfig \(p. 373\)](#)

Provides configuration parameters for the output of PII entity detection jobs.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: Yes

#### [RedactionConfig \(p. 373\)](#)

Provides configuration parameters for PII entity redaction.

This parameter is required if you set the Mode parameter to ONLY\_REDACTION. In that case, you must provide a RedactionConfig definition that includes the PiiEntityTypes parameter.

Type: [RedactionConfig \(p. 491\)](#) object

Required: No

### [Tags \(p. 373\)](#)

Tags to be associated with the PII entities detection job. A tag is a key-value pair that adds metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

## Response Syntax

```
{
 "JobArn": "string",
 "JobId": "string",
 "JobStatus": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

### [JobArn \(p. 375\)](#)

The Amazon Resource Name (ARN) of the PII entity detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

```
arn:<partition>:comprehend:<region>:<account-id>:pii-entities-detection-job/
<job-id>
```

The following is an example job ARN:

```
arn:aws:comprehend:us-west-2:111122223333:pii-entities-detection-
job/1234abcd12ab34cd56ef1234567890ab
```

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]  
{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*?)?

### [JobId \(p. 375\)](#)

The identifier generated for the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/=+\-%@]\*\$

### [JobStatus \(p. 375\)](#)

The status of the job.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

### **TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StartSentimentDetectionJob

Starts an asynchronous sentiment detection job for a collection of documents. Use the [DescribeSentimentDetectionJob \(p. 285\)](#) operation to track the status of a job.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobName": "string",
 "LanguageCode": "string",
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ClientRequestToken \(p. 377\)](#)

A unique identifier for the request. If you don't set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

#### [DataAccessRoleArn \(p. 377\)](#)

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data. For more information, see <https://>

[docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions](https://docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions).

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: Yes

#### [InputDataConfig \(p. 377\)](#)

Specifies the format and location of the input data for the job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: Yes

#### [JobName \(p. 377\)](#)

The identifier of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:=+\-%@]\*\$

Required: No

#### [LanguageCode \(p. 377\)](#)

The language of the input documents. You can specify any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: Yes

#### [OutputDataConfig \(p. 377\)](#)

Specifies where to send the output files.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: Yes

#### [Tags \(p. 377\)](#)

Tags to be associated with the sentiment detection job. A tag is a key-value pair that adds metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

#### [VolumeKmsKeyId \(p. 377\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### [VpcConfig \(p. 377\)](#)

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for your sentiment detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## Response Syntax

```
{
 "JobArn": "string",
 "JobId": "string",
 "JobStatus": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [JobArn \(p. 379\)](#)

The Amazon Resource Name (ARN) of the sentiment detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

```
arn:<partition>:comprehend:<region>:<account-id>:sentiment-detection-job/
<job-id>
```

The following is an example job ARN:

```
arn:aws:comprehend:us-west-2:111122223333:sentiment-detection-
job/1234abcd12ab34cd56ef1234567890ab
```

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]  
{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*?)?

#### [JobId \(p. 379\)](#)

The identifier generated for the job. To get the status of a job, use this identifier with the [DescribeSentimentDetectionJob \(p. 285\)](#) operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

#### **JobStatus (p. 379)**

The status of the job.

- SUBMITTED - The job has been received and is queued for processing.
- IN\_PROGRESS - Amazon Comprehend is processing the job.
- COMPLETED - The job was successfully completed and the output is available.
- FAILED - The job did not complete. To get details, use the [DescribeSentimentDetectionJob \(p. 285\)](#) operation.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

### **TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)

- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StartTopicsDetectionJob

Starts an asynchronous topic detection job. Use the `DescribeTopicDetectionJob` operation to track the status of a job.

### Request Syntax

```
{
 "ClientRequestToken": "string",
 "DataAccessRoleArn": "string",
 "InputDataConfig": {
 "DocumentReaderConfig": {
 "DocumentReadAction": "string",
 "DocumentReadMode": "string",
 "FeatureTypes": ["string"]
 },
 "InputFormat": "string",
 "S3Uri": "string"
 },
 "JobName": "string",
 "NumberOfTopics": number,
 "OutputDataConfig": {
 "KmsKeyId": "string",
 "S3Uri": "string"
 },
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "VolumeKmsKeyId": "string",
 "VpcConfig": {
 "SecurityGroupIds": ["string"],
 "Subnets": ["string"]
 }
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ClientRequestToken \(p. 382\)](#)

A unique identifier for the request. If you do not set the client request token, Amazon Comprehend generates one.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-]+\$

Required: No

#### [DataAccessRoleArn \(p. 382\)](#)

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data. For more information, see <https://>

[docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions](https://docs.aws.amazon.com/comprehend/latest/dg/access-control-managing-permissions.html#auth-role-permissions).

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: Yes

#### [InputDataConfig \(p. 382\)](#)

Specifies the format and location of the input data for the job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: Yes

#### [JobName \(p. 382\)](#)

The identifier of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/:=+\-%@]\*\$

Required: No

#### [NumberOfTopics \(p. 382\)](#)

The number of topics to detect.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

#### [OutputDataConfig \(p. 382\)](#)

Specifies where to send the output files. The output is a compressed archive with two files, `topic-terms.csv` that lists the terms associated with each topic, and `doc-topics.csv` that lists the documents associated with each topic

Type: [OutputDataConfig \(p. 482\)](#) object

Required: Yes

#### [Tags \(p. 382\)](#)

Tags to be associated with the topics detection job. A tag is a key-value pair that adds metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

Type: Array of [Tag \(p. 498\)](#) objects

Required: No

#### [VolumeKmsKeyId \(p. 382\)](#)

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### VpcConfig (p. 382)

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for your topic detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## Response Syntax

```
{
 "JobArn": "string",
 "JobId": "string",
 "JobStatus": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### JobArn (p. 384)

The Amazon Resource Name (ARN) of the topics detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

arn:<partition>:comprehend:<region>:<account-id>:topics-detection-job/<job-id>

The following is an example job ARN:

```
arn:aws:comprehend:us-west-2:111122223333:document-classification-job/1234abcd12ab34cd56ef1234567890ab
```

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*?)?

#### JobId (p. 384)

The identifier generated for the job. To get the status of the job, use this identifier with the [DescribeTopicDetectionJob](#) operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

#### [JobStatus \(p. 384\)](#)

The status of the job:

- SUBMITTED - The job has been received and is queued for processing.
- IN\_PROGRESS - Amazon Comprehend is processing the job.
- COMPLETED - The job was successfully completed and the output is available.
- FAILED - The job did not complete. To get details, use the [DescribeTopicDetectionJob](#) operation.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **KmsKeyValidationException**

The KMS customer managed key (CMK) entered cannot be validated. Verify the key and re-enter it.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

### **TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)

- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StopDominantLanguageDetectionJob

Stops a dominant language detection job in progress.

If the job state is `IN_PROGRESS` the job is marked for termination and put into the `STOP_REQUESTED` state. If the job completes before it can be stopped, it is put into the `COMPLETED` state; otherwise the job is stopped and put into the `STOPPED` state.

If the job is in the `COMPLETED` or `FAILED` state when you call the `StopDominantLanguageDetectionJob` operation, the operation returns a 400 Internal Request Exception.

When a job is stopped, any documents already processed are written to the output location.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [JobId \(p. 387\)](#)

The identifier of the dominant language detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:=+-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "JobId": "string",
 "JobStatus": "string"
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [JobId \(p. 387\)](#)

The identifier of the dominant language detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-+\\-%@]\*\$

#### [JobStatus \(p. 387\)](#)

Either STOP\_REQUESTED if the job is currently running, or STOPPED if the job was previously stopped with the StopDominantLanguageDetectionJob operation.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StopEntitiesDetectionJob

Stops an entities detection job in progress.

If the job state is `IN_PROGRESS` the job is marked for termination and put into the `STOP_REQUESTED` state. If the job completes before it can be stopped, it is put into the `COMPLETED` state; otherwise the job is stopped and put into the `STOPPED` state.

If the job is in the `COMPLETED` or `FAILED` state when you call the `StopDominantLanguageDetectionJob` operation, the operation returns a 400 Internal Request Exception.

When a job is stopped, any documents already processed are written to the output location.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [JobId \(p. 389\)](#)

The identifier of the entities detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:=+-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "JobId": "string",
 "JobStatus": "string"
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [JobId \(p. 389\)](#)

The identifier of the entities detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-+\\-%@]\*\$

#### [JobStatus \(p. 389\)](#)

Either STOP\_REQUESTED if the job is currently running, or STOPPED if the job was previously stopped with the StopEntitiesDetectionJob operation.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StopEventsDetectionJob

Stops an events detection job in progress.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **JobId** (p. 391)

The identifier of the events detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "JobId": "string",
 "JobStatus": "string"
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### **JobId** (p. 391)

The identifier of the events detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\*\$

#### **JobStatus** (p. 391)

The status of the events detection job.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StopKeyPhrasesDetectionJob

Stops a key phrases detection job in progress.

If the job state is `IN_PROGRESS` the job is marked for termination and put into the `STOP_REQUESTED` state. If the job completes before it can be stopped, it is put into the `COMPLETED` state; otherwise the job is stopped and put into the `STOPPED` state.

If the job is in the `COMPLETED` or `FAILED` state when you call the `StopDominantLanguageDetectionJob` operation, the operation returns a 400 Internal Request Exception.

When a job is stopped, any documents already processed are written to the output location.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [JobId \(p. 393\)](#)

The identifier of the key phrases detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:=+-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "JobId": "string",
 "JobStatus": "string"
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [JobId \(p. 393\)](#)

The identifier of the key phrases detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-+\\-%@]\*\$

### [JobStatus \(p. 393\)](#)

Either STOP\_REQUESTED if the job is currently running, or STOPPED if the job was previously stopped with the StopKeyPhrasesDetectionJob operation.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StopPiiEntitiesDetectionJob

Stops a PII entities detection job in progress.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### **JobId** (p. 395)

The identifier of the PII entities detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/+/-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "JobId": "string",
 "JobStatus": "string"
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### **JobId** (p. 395)

The identifier of the PII entities detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/+/-%@]\*\$

#### **JobStatus** (p. 395)

The status of the PII entities detection job.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StopSentimentDetectionJob

Stops a sentiment detection job in progress.

If the job state is `IN_PROGRESS` the job is marked for termination and put into the `STOP_REQUESTED` state. If the job completes before it can be stopped, it is put into the `COMPLETED` state; otherwise the job is stopped and put into the `STOPPED` state.

If the job is in the `COMPLETED` or `FAILED` state when you call the `StopDominantLanguageDetectionJob` operation, the operation returns a 400 Internal Request Exception.

When a job is stopped, any documents already processed are written to the output location.

### Request Syntax

```
{
 "JobId": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [JobId \(p. 397\)](#)

The identifier of the sentiment detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:/:=+-%@]\*\$

Required: Yes

### Response Syntax

```
{
 "JobId": "string",
 "JobStatus": "string"
}
```

### Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

#### [JobId \(p. 397\)](#)

The identifier of the sentiment detection job to stop.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-+\\-%@]\*\$

#### [JobStatus \(p. 397\)](#)

Either STOP\_REQUESTED if the job is currently running, or STOPPED if the job was previously stopped with the StopSentimentDetectionJob operation.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **JobNotFoundException**

The specified job was not found. Check the job ID and try again.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## StopTrainingDocumentClassifier

Stops a document classifier training job while in progress.

If the training job state is `TRAINING`, the job is marked for termination and put into the `STOP_REQUESTED` state. If the training job completes before it can be stopped, it is put into the `TRAINED`; otherwise the training job is stopped and put into the `STOPPED` state and the service sends back an HTTP 200 response with an empty HTTP body.

### Request Syntax

```
{
 "DocumentClassifierArn": "string"
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [DocumentClassifierArn \(p. 399\)](#)

The Amazon Resource Name (ARN) that identifies the document classifier currently being trained.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:document-classifier/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*)?`

Required: Yes

### Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

### Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

#### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

#### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

#### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### TooManyRequestsException

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

# StopTrainingEntityRecognizer

Stops an entity recognizer training job while in progress.

If the training job state is `TRAINING`, the job is marked for termination and put into the `STOP_REQUESTED` state. If the training job completes before it can be stopped, it is put into the `TRAINED`; otherwise the training job is stopped and put into the `STOPPED` state and the service sends back an HTTP 200 response with an empty HTTP body.

## Request Syntax

```
{
 "EntityRecognizerArn": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

### **EntityRecognizerArn** (p. 401)

The Amazon Resource Name (ARN) that identifies the entity recognizer currently being trained.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:entity-recognizer/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*)?`

Required: Yes

## Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### TooManyRequestsException

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## TagResource

Associates a specific tag with an Amazon Comprehend resource. A tag is a key-value pair that adds as a metadata to a resource used by Amazon Comprehend. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department.

### Request Syntax

```
{
 "ResourceArn": "string",
 "Tags": [
 {
 "Key": "string",
 "Value": "string"
 }
]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ResourceArn \(p. 403\)](#)

The Amazon Resource Name (ARN) of the given Amazon Comprehend resource to which you want to associate the tags.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: Yes

#### [Tags \(p. 403\)](#)

Tags being associated with a specific Amazon Comprehend resource. There can be a maximum of 50 tags (both existing and pending) associated with a specific resource.

Type: Array of [Tag \(p. 498\)](#) objects

Required: Yes

## Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

### Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **ConcurrentModificationException**

Concurrent modification of the tags associated with an Amazon Comprehend resource is not supported.

HTTP Status Code: 400

### **InternalServerException**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **TooManyTagsException**

The request contains more tags than can be associated with a resource (50 tags per resource). The maximum number of tags includes both existing tags and those included in your current request.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## UntagResource

Removes a specific tag associated with an Amazon Comprehend resource.

### Request Syntax

```
{
 "ResourceArn": "string",
 "TagKeys": ["string"]
}
```

### Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

#### [ResourceArn \(p. 405\)](#)

The Amazon Resource Name (ARN) of the given Amazon Comprehend resource from which you want to remove the tags.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: Yes

#### [TagKeys \(p. 405\)](#)

The initial part of a key-value pair that forms a tag being removed from a given resource. For example, a tag with "Sales" as the key might be added to a resource to indicate its use by the sales department. Keys must be unique and cannot be duplicated for a particular resource.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: Yes

## Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

#### **ConcurrentModificationException**

Concurrent modification of the tags associated with an Amazon Comprehend resource is not supported.

HTTP Status Code: 400

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **TooManyTagKeysException**

The request contains more tag keys than can be associated with a resource (50 tag keys per resource).

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

# UpdateEndpoint

Updates information about the specified endpoint.

## Request Syntax

```
{
 "DesiredDataAccessRoleArn": "string",
 "DesiredInferenceUnits": number,
 "DesiredModelArn": "string",
 "EndpointArn": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 505\)](#).

The request accepts the following data in JSON format.

### [DesiredDataAccessRoleArn \(p. 407\)](#)

Data access role ARN to use in case the new model is encrypted with a customer CMK.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: No

### [DesiredInferenceUnits \(p. 407\)](#)

The desired number of inference units to be used by the model using this endpoint. Each inference unit represents of a throughput of 100 characters per second.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

### [DesiredModelArn \(p. 407\)](#)

The ARN of the new model to use when updating an existing endpoint.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier|entity-recognizer)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

### [EndpointArn \(p. 407\)](#)

The Amazon Resource Number (ARN) of the endpoint being updated.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier-endpoint|entity-recognizer-endpoint)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

Required: Yes

## Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

## Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 503\)](#).

### **InternalServerError**

An internal server error occurred. Retry your request.

HTTP Status Code: 500

### **InvalidRequestException**

The request is invalid.

HTTP Status Code: 400

### **ResourceInUseException**

The specified resource name is already in use. Use a different name and try your request again.

HTTP Status Code: 400

### **ResourceLimitExceeded**

The maximum number of resources per account has been exceeded. Review the resources, and then try your request again.

HTTP Status Code: 400

### **ResourceNotFoundException**

The specified resource ARN was not found. Check the ARN and try your request again.

HTTP Status Code: 400

### **ResourceUnavailableException**

The specified resource is not available. Check the resource and try your request again.

HTTP Status Code: 400

### **TooManyRequestsException**

The number of requests exceeds the limit. Resubmit your request later.

HTTP Status Code: 400

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

## Data Types

The following data types are supported:

- [AugmentedManifestsListItem](#) (p. 411)
- [BatchDetectDominantLanguageItemResult](#) (p. 413)
- [BatchDetectEntitiesItemResult](#) (p. 414)
- [BatchDetectKeyPhrasesItemResult](#) (p. 415)
- [BatchDetectSentimentItemResult](#) (p. 416)
- [BatchDetectSyntaxItemResult](#) (p. 417)
- [BatchItemError](#) (p. 418)
- [ClassifierEvaluationMetrics](#) (p. 419)
- [ClassifierMetadata](#) (p. 421)
- [DocumentClass](#) (p. 422)
- [DocumentClassificationJobFilter](#) (p. 423)
- [DocumentClassificationJobProperties](#) (p. 424)
- [DocumentClassifierFilter](#) (p. 427)
- [DocumentClassifierInputDataConfig](#) (p. 428)
- [DocumentClassifierOutputDataConfig](#) (p. 430)
- [DocumentClassifierProperties](#) (p. 431)
- [DocumentClassifierSummary](#) (p. 435)
- [DocumentLabel](#) (p. 437)
- [DocumentReaderConfig](#) (p. 438)
- [DominantLanguage](#) (p. 439)
- [DominantLanguageDetectionJobFilter](#) (p. 440)
- [DominantLanguageDetectionJobProperties](#) (p. 441)
- [EndpointFilter](#) (p. 444)
- [EndpointProperties](#) (p. 445)
- [EntitiesDetectionJobFilter](#) (p. 448)
- [EntitiesDetectionJobProperties](#) (p. 449)
- [Entity](#) (p. 452)
- [EntityLabel](#) (p. 454)
- [EntityRecognizerAnnotations](#) (p. 455)
- [EntityRecognizerDocuments](#) (p. 456)
- [EntityRecognizerEntityList](#) (p. 457)

- [EntityRecognizerEvaluationMetrics \(p. 458\)](#)
- [EntityRecognizerFilter \(p. 459\)](#)
- [EntityRecognizerInputDataConfig \(p. 460\)](#)
- [EntityRecognizerMetadata \(p. 462\)](#)
- [EntityRecognizerMetadataEntityTypesListItem \(p. 463\)](#)
- [EntityRecognizerProperties \(p. 464\)](#)
- [EntityRecognizerSummary \(p. 467\)](#)
- [EntityTypesEvaluationMetrics \(p. 469\)](#)
- [EntityTypesListItem \(p. 470\)](#)
- [EventsDetectionJobFilter \(p. 471\)](#)
- [EventsDetectionJobProperties \(p. 472\)](#)
- [InputDataConfig \(p. 475\)](#)
- [KeyPhrase \(p. 477\)](#)
- [KeyPhrasesDetectionJobFilter \(p. 478\)](#)
- [KeyPhrasesDetectionJobProperties \(p. 479\)](#)
- [OutputDataConfig \(p. 482\)](#)
- [PartOfSpeechTag \(p. 483\)](#)
- [PiiEntitiesDetectionJobFilter \(p. 484\)](#)
- [PiiEntitiesDetectionJobProperties \(p. 485\)](#)
- [PiiEntity \(p. 488\)](#)
- [PiiOutputDataConfig \(p. 490\)](#)
- [RedactionConfig \(p. 491\)](#)
- [SentimentDetectionJobFilter \(p. 492\)](#)
- [SentimentDetectionJobProperties \(p. 493\)](#)
- [SentimentScore \(p. 496\)](#)
- [SyntaxToken \(p. 497\)](#)
- [Tag \(p. 498\)](#)
- [TopicsDetectionJobFilter \(p. 499\)](#)
- [TopicsDetectionJobProperties \(p. 500\)](#)
- [VpcConfig \(p. 503\)](#)

## AugmentedManifestsListItem

An augmented manifest file that provides training data for your custom model. An augmented manifest file is a labeled dataset that is produced by Amazon SageMaker Ground Truth.

### Contents

#### AnnotationDataS3Uri

The S3 prefix to the annotation files that are referred in the augmented manifest file.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-zA-Z0-9][\.\-\a-zA-Z0-9]{1,61}[a-zA-Z0-9](/.\*)?

Required: No

#### AttributeNames

The JSON attribute that contains the annotations for your training documents. The number of attribute names that you specify depends on whether your augmented manifest file is the output of a single labeling job or a chained labeling job.

If your file is the output of a single labeling job, specify the LabelAttributeName key that was used when the job was created in Ground Truth.

If your file is the output of a chained labeling job, specify the LabelAttributeName key for one or more jobs in the chain. Each LabelAttributeName key provides the annotations from an individual job.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

Required: Yes

#### DocumentType

The type of augmented manifest. PlainTextDocument or SemiStructuredDocument. If you don't specify, the default is PlainTextDocument.

- PLAIN\_TEXT\_DOCUMENT A document type that represents any unicode text that is encoded in UTF-8.
- SEMI\_STRUCTURED\_DOCUMENT A document type with positional and structural context, like a PDF. For training with Amazon Comprehend, only PDFs are supported. For inference, Amazon Comprehend support PDFs, DOCX and TXT.

Type: String

Valid Values: PLAIN\_TEXT\_DOCUMENT | SEMI\_STRUCTURED\_DOCUMENT

Required: No

#### S3Uri

The Amazon S3 location of the augmented manifest file.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: `s3://[a-z0-9][\.\-\_a-z0-9]{1,61}[a-z0-9](/.*)?`

Required: Yes

#### **SourceDocumentsS3Uri**

The S3 prefix to the source files (PDFs) that are referred to in the augmented manifest file.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: `s3://[a-z0-9][\.\-\_a-z0-9]{1,61}[a-z0-9](/.*)?`

Required: No

#### **Split**

The purpose of the data you've provided in the augmented manifest. You can either train or test this data. If you don't specify, the default is train.

TRAIN - all of the documents in the manifest will be used for training. If no test documents are provided, Amazon Comprehend will automatically reserve a portion of the training documents for testing.

TEST - all of the documents in the manifest will be used for testing.

Type: String

Valid Values: TRAIN | TEST

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## BatchDetectDominantLanguageItemResult

The result of calling the [BatchDetectDominantLanguage \(p. 220\)](#) operation. The operation returns one object for each document that is successfully processed by the operation.

### Contents

#### Index

The zero-based index of the document in the input list.

Type: Integer

Required: No

#### Languages

One or more [DominantLanguage \(p. 439\)](#) objects describing the dominant languages in the document.

Type: Array of [DominantLanguage \(p. 439\)](#) objects

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## BatchDetectEntitiesItemResult

The result of calling the [BatchDetectEntities \(p. 223\)](#) operation. The operation returns one object for each document that is successfully processed by the operation.

### Contents

#### Entities

One or more [Entity \(p. 452\)](#) objects, one for each entity detected in the document.

Type: Array of [Entity \(p. 452\)](#) objects

Required: No

#### Index

The zero-based index of the document in the input list.

Type: Integer

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## BatchDetectKeyPhrasesItemResult

The result of calling the [BatchDetectKeyPhrases](#) (p. 226) operation. The operation returns one object for each document that is successfully processed by the operation.

### Contents

#### Index

The zero-based index of the document in the input list.

Type: Integer

Required: No

#### KeyPhrases

One or more [KeyPhrase](#) (p. 477) objects, one for each key phrase detected in the document.

Type: Array of [KeyPhrase](#) (p. 477) objects

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## BatchDetectSentimentItemResult

The result of calling the [BatchDetectSentiment \(p. 229\)](#) operation. The operation returns one object for each document that is successfully processed by the operation.

### Contents

#### Index

The zero-based index of the document in the input list.

Type: Integer

Required: No

#### Sentiment

The sentiment detected in the document.

Type: String

Valid Values: POSITIVE | NEGATIVE | NEUTRAL | MIXED

Required: No

#### SentimentScore

The level of confidence that Amazon Comprehend has in the accuracy of its sentiment detection.

Type: [SentimentScore \(p. 496\)](#) object

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## BatchDetectSyntaxItemResult

The result of calling the [BatchDetectSyntax](#) (p. 232) operation. The operation returns one object that is successfully processed by the operation.

### Contents

#### Index

The zero-based index of the document in the input list.

Type: Integer

Required: No

#### SyntaxTokens

The syntax tokens for the words in the document, one token for each word.

Type: Array of [SyntaxToken](#) (p. 497) objects

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## BatchItemError

Describes an error that occurred while processing a document in a batch. The operation returns one `BatchItemError` object for each document that contained an error.

### Contents

#### **ErrorCode**

The numeric error code of the error.

Type: String

Length Constraints: Minimum length of 1.

Required: No

#### **ErrorMessage**

A text description of the error.

Type: String

Length Constraints: Minimum length of 1.

Required: No

#### **Index**

The zero-based index of the document in the input list.

Type: Integer

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# ClassifierEvaluationMetrics

Describes the result metrics for the test data associated with an documentation classifier.

## Contents

### Accuracy

The fraction of the labels that were correct recognized. It is computed by dividing the number of labels in the test documents that were correctly recognized by the total number of labels in the test documents.

Type: Double

Required: No

### F1Score

A measure of how accurate the classifier results are for the test data. It is derived from the Precision and Recall values. The F1Score is the harmonic average of the two scores. The highest score is 1, and the worst score is 0.

Type: Double

Required: No

### HammingLoss

Indicates the fraction of labels that are incorrectly predicted. Also seen as the fraction of wrong labels compared to the total number of labels. Scores closer to zero are better.

Type: Double

Required: No

### MicroF1Score

A measure of how accurate the classifier results are for the test data. It is a combination of the Micro Precision and Micro Recall values. The Micro F1Score is the harmonic mean of the two scores. The highest score is 1, and the worst score is 0.

Type: Double

Required: No

### MicroPrecision

A measure of the usefulness of the recognizer results in the test data. High precision means that the recognizer returned substantially more relevant results than irrelevant ones. Unlike the Precision metric which comes from averaging the precision of all available labels, this is based on the overall score of all precision scores added together.

Type: Double

Required: No

### MicroRecall

A measure of how complete the classifier results are for the test data. High recall means that the classifier returned most of the relevant results. Specifically, this indicates how many of the correct categories in the text that the model can predict. It is a percentage of correct categories in the text that can be found. Instead of averaging the recall scores of all labels (as with Recall), micro Recall is based on the overall score of all recall scores added together.

Type: Double

Required: No

#### Precision

A measure of the usefulness of the classifier results in the test data. High precision means that the classifier returned substantially more relevant results than irrelevant ones.

Type: Double

Required: No

#### Recall

A measure of how complete the classifier results are for the test data. High recall means that the classifier returned most of the relevant results.

Type: Double

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# ClassifierMetadata

Provides information about a document classifier.

## Contents

### EvaluationMetrics

Describes the result metrics for the test data associated with an documentation classifier.

Type: [ClassifierEvaluationMetrics \(p. 419\)](#) object

Required: No

### NumberOfLabels

The number of labels in the input data.

Type: Integer

Required: No

### NumberOfTestDocuments

The number of documents in the input data that were used to test the classifier. Typically this is 10 to 20 percent of the input documents, up to 10,000 documents.

Type: Integer

Required: No

### NumberOfTrainedDocuments

The number of documents in the input data that were used to train the classifier. Typically this is 80 to 90 percent of the input documents.

Type: Integer

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## DocumentClass

Specifies the class that categorizes the document being analyzed

### Contents

#### Name

The name of the class.

Type: String

Length Constraints: Minimum length of 1.

Required: No

#### Score

The confidence score that Amazon Comprehend has this class correctly attributed.

Type: Float

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## DocumentClassificationJobFilter

Provides information for filtering a list of document classification jobs. For more information, see the [ListDocumentClassificationJobs \(p. 309\)](#) operation. You can provide only one filter parameter in each request.

### Contents

#### JobName

Filters on the name of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}]\p{N}\_.:=/+\\-%@]\*\$

Required: No

#### JobStatus

Filters the list based on job status. Returns only jobs with the specified status.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### SubmitTimeAfter

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted after the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

#### SubmitTimeBefore

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted before the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# DocumentClassificationJobProperties

Provides information about a document classification job.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) of the AWS identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+`

Required: No

### DocumentClassifierArn

The Amazon Resource Name (ARN) that identifies the document classifier.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:document-classifier/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*)?`

Required: No

### EndTime

The time that the document classification job completed.

Type: Timestamp

Required: No

### InputDataConfig

The input data configuration that you supplied when you created the document classification job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: No

### JobArn

The Amazon Resource Name (ARN) of the document classification job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

`arn:<partition>:comprehend:<region>:<account-id>:document-classification-job/<job-id>`

The following is an example job ARN:

`arn:aws:comprehend:us-west-2:111122223333:document-classification-job/1234abcd12ab34cd56ef1234567890ab`

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[ ^:]\*)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

#### **JobId**

The identifier assigned to the document classification job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\*)\$

Required: No

#### **JobName**

The name that you assigned to the document classification job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\*)\$

Required: No

#### **JobStatus**

The current status of the document classification job. If the status is FAILED, the Message field shows the reason for the failure.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **Message**

A description of the status of the job.

Type: String

Required: No

#### **OutputDataConfig**

The output data configuration that you supplied when you created the document classification job.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: No

#### **SubmitTime**

The time that the document classification job was submitted for processing.

Type: Timestamp

Required: No

#### **VolumeKmsKeyId**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### **VpcConfig**

Configuration parameters for a private Virtual Private Cloud (VPC) containing the resources you are using for your document classification job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# DocumentClassifierFilter

Provides information for filtering a list of document classifiers. You can only specify one filtering parameter in a request. For more information, see the [ListDocumentClassifiers \(p. 312\)](#) operation.

## Contents

### DocumentClassifierName

The name that you assigned to the document classifier

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

### Status

Filters the list of classifiers based on status.

Type: String

Valid Values: SUBMITTED | TRAINING | DELETING | STOP\_REQUESTED | STOPPED | IN\_ERROR | TRAINED

Required: No

### SubmitTimeAfter

Filters the list of classifiers based on the time that the classifier was submitted for processing. Returns only classifiers submitted after the specified time. Classifiers are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

### SubmitTimeBefore

Filters the list of classifiers based on the time that the classifier was submitted for processing. Returns only classifiers submitted before the specified time. Classifiers are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# DocumentClassifierInputDataConfig

The input properties for training a document classifier.

For more information on how the input file is formatted, see [Creating Training Data \(p. 127\)](#).

## Contents

### AugmentedManifests

A list of augmented manifest files that provide training data for your custom model. An augmented manifest file is a labeled dataset that is produced by Amazon SageMaker Ground Truth.

This parameter is required if you set `DataFormat` to `AUGMENTED_MANIFEST`.

Type: Array of [AugmentedManifestsListItem \(p. 411\)](#) objects

Required: No

### DataFormat

The format of your training data:

- `COMPREHEND_CSV`: A two-column CSV file, where labels are provided in the first column, and documents are provided in the second. If you use this value, you must provide the `S3Uri` parameter in your request.
- `AUGMENTED_MANIFEST`: A labeled dataset that is produced by Amazon SageMaker Ground Truth. This file is in JSON lines format. Each line is a complete JSON object that contains a training document and its associated labels.

If you use this value, you must provide the `AugmentedManifests` parameter in your request.

If you don't specify a value, Amazon Comprehend uses `COMPREHEND_CSV` as the default.

Type: String

Valid Values: `COMPREHEND_CSV` | `AUGMENTED_MANIFEST`

Required: No

### LabelDelimiter

Indicates the delimiter used to separate each label for training a multi-label classifier. The default delimiter between labels is a pipe (|). You can use a different character as a delimiter (if it's an allowed character) by specifying it under `Delimiter` for labels. If the training documents use a delimiter other than the default or the delimiter you specify, the labels on that line will be combined to make a single unique label, such as LABEL1LABEL2LABEL.

Type: String

Length Constraints: Fixed length of 1.

Pattern: ^[ ~!@#\$%^\*-\_+=|\\";:\t>?/]\$

Required: No

### S3Uri

The Amazon S3 URI for the input data. The S3 bucket must be in the same region as the API endpoint that you are calling. The URI can point to a single input file or it can provide the prefix for a collection of input files.

For example, if you use the URI `s3://bucketName/prefix`, if the prefix is a single file, Amazon Comprehend uses that file as input. If more than one file begins with the prefix, Amazon Comprehend uses all of them as input.

This parameter is required if you set `DataFormat` to `COMPREHEND_CSV`.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: `s3://[a-z0-9][\.\-\a-z0-9]{1,61}[a-z0-9](/.*?)?`

Required: No

#### TestS3Uri

The Amazon S3 URI for the input data. The Amazon S3 bucket must be in the same AWS Region as the API endpoint that you are calling. The URI can point to a single input file or it can provide the prefix for a collection of input files.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: `s3://[a-z0-9][\.\-\a-z0-9]{1,61}[a-z0-9](/.*?)?`

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# DocumentClassifierOutputDataConfig

Provides output results configuration parameters for custom classifier jobs.

## Contents

### KmsKeyId

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt the output results from an analysis job. The KmsKeyId can be one of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
- KMS Key Alias: "alias/ExampleAlias"
- ARN of a KMS Key Alias: "arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

### S3Uri

When you use the `OutputDataConfig` object while creating a custom classifier, you specify the Amazon S3 location where you want to write the confusion matrix. The URI must be in the same region as the API endpoint that you are calling. The location is used as the prefix for the actual location of this output file.

When the custom classifier job is finished, the service creates the output file in a directory specific to the job. The `S3Uri` field contains the location of the output file, called `output.tar.gz`. It is a compressed archive that contains the confusion matrix.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-z0-9][\.-a-z0-9]{1,61}[a-z0-9](/.\*)?

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# DocumentClassifierProperties

Provides information about a document classifier.

## Contents

### ClassifierMetadata

Information about the document classifier, including the number of documents used for training the classifier, the number of documents used for test the classifier, and an accuracy rating.

Type: [ClassifierMetadata \(p. 421\)](#) object

Required: No

### DataAccessRoleArn

The Amazon Resource Name (ARN) of the AWS Identity and Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: No

### DocumentClassifierArn

The Amazon Resource Name (ARN) that identifies the document classifier.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:document-classifier/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*?)?

Required: No

### EndTime

The time that training the document classifier completed.

Type: Timestamp

Required: No

### InputDataConfig

The input data configuration that you supplied when you created the document classifier for training.

Type: [DocumentClassifierInputDataConfig \(p. 428\)](#) object

Required: No

### LanguageCode

The language code for the language of the documents that the classifier was trained on.

Type: String

Valid Values: en | es | fr | de | it | pt

Required: No

#### Message

Additional information about the status of the classifier.

Type: String

Required: No

#### Mode

Indicates the mode in which the specific classifier was trained. This also indicates the format of input documents and the format of the confusion matrix. Each classifier can only be trained in one mode and this cannot be changed once the classifier is trained.

Type: String

Valid Values: MULTI\_CLASS | MULTI\_LABEL

Required: No

#### ModelKmsKeyId

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt trained custom models. The ModelKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### OutputDataConfig

Provides output results configuration parameters for custom classifier jobs.

Type: [DocumentClassifierOutputDataConfig \(p. 430\)](#) object

Required: No

#### Status

The status of the document classifier. If the status is TRAINED the classifier is ready to use. If the status is FAILED you can see additional information about why the classifier wasn't trained in the Message field.

Type: String

Valid Values: SUBMITTED | TRAINING | DELETING | STOP\_REQUESTED | STOPPED | IN\_ERROR | TRAINED

Required: No

#### SubmitTime

The time that the document classifier was submitted for training.

Type: Timestamp

Required: No

**TrainingEndTime**

The time that training of the document classifier was completed. Indicates the time when the training completes on documentation classifiers. You are billed for the time interval between this time and the value of TrainingStartTime.

Type: Timestamp

Required: No

**TrainingStartTime**

Indicates the time when the training starts on documentation classifiers. You are billed for the time interval between this time and the value of TrainingEndTime.

Type: Timestamp

Required: No

**VersionName**

The version name that you assigned to the document classifier.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

**VolumeKmsKeyId**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

**VpcConfig**

Configuration parameters for a private Virtual Private Cloud (VPC) containing the resources you are using for your custom classifier. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## DocumentClassifierSummary

Describes information about a document classifier and its versions.

### Contents

#### **DocumentClassifierName**

The name that you assigned the document classifier.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

#### **LatestVersionCreatedAt**

The time that the latest document classifier version was submitted for processing.

Type: Timestamp

Required: No

#### **LatestVersionName**

The version name you assigned to the latest document classifier version.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

#### **LatestVersionStatus**

Provides the status of the latest document classifier version.

Type: String

Valid Values: SUBMITTED | TRAINING | DELETING | STOP\_REQUESTED | STOPPED | IN\_ERROR | TRAINED

Required: No

#### **NumberOfVersions**

The number of versions you created.

Type: Integer

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## DocumentLabel

Specifies one of the label or labels that categorize the document being analyzed.

### Contents

#### Name

The name of the label.

Type: String

Length Constraints: Minimum length of 1.

Required: No

#### Score

The confidence score that Amazon Comprehend has this label correctly attributed.

Type: Float

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# DocumentReaderConfig

The input properties for a topic detection job.

## Contents

### DocumentReadAction

This enum field will start with two values which will apply to PDFs:

- `TEXTRACT_DETECT_DOCUMENT_TEXT` - The service calls DetectDocumentText for PDF documents per page.
- `TEXTRACT_ANALYZE_DOCUMENT` - The service calls AnalyzeDocument for PDF documents per page.

Type: String

Valid Values: `TEXTRACT_DETECT_DOCUMENT_TEXT` | `TEXTRACT_ANALYZE_DOCUMENT`

Required: Yes

### DocumentReadMode

This enum field provides two values:

- `SERVICE_DEFAULT` - use service defaults for Document reading. For Digital PDF it would mean using an internal parser instead of Textract APIs
- `FORCE_DOCUMENT_READ_ACTION` - Always use specified action for DocumentReadAction, including Digital PDF.

Type: String

Valid Values: `SERVICE_DEFAULT` | `FORCE_DOCUMENT_READ_ACTION`

Required: No

### FeatureTypes

Specifies how the text in an input file should be processed:

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 2 items.

Valid Values: `TABLES` | `FORMS`

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## DominantLanguage

Returns the code for the dominant language in the input text and the level of confidence that Amazon Comprehend has in the accuracy of the detection.

### Contents

#### LanguageCode

The RFC 5646 language code for the dominant language. For more information about RFC 5646, see [Tags for Identifying Languages](#) on the *IETF Tools* web site.

Type: String

Length Constraints: Minimum length of 1.

Required: No

#### Score

The level of confidence that Amazon Comprehend has in the accuracy of the detection.

Type: Float

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## DominantLanguageDetectionJobFilter

Provides information for filtering a list of dominant language detection jobs. For more information, see the [ListDominantLanguageDetectionJobs \(p. 317\)](#) operation.

### Contents

#### **JobName**

Filters on the name of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\*\$

Required: No

#### **JobStatus**

Filters the list of jobs based on job status. Returns only jobs with the specified status.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **SubmitTimeAfter**

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted after the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

#### **SubmitTimeBefore**

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted before the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# DominantLanguageDetectionJobProperties

Provides information about a dominant language detection job.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) that gives Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+`

Required: No

### EndTime

The time that the dominant language detection job completed.

Type: Timestamp

Required: No

### InputDataConfig

The input data configuration that you supplied when you created the dominant language detection job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: No

### JobArn

The Amazon Resource Name (ARN) of the dominant language detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

`arn:<partition>:comprehend:<region>:<account-id>:dominant-language-detection-job/<job-id>`

The following is an example job ARN:

`arn:aws:comprehend:us-west-2:111122223333:dominant-language-detection-job/1234abcd12ab34cd56ef1234567890ab`

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*?)?`

Required: No

### JobId

The identifier assigned to the dominant language detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-@]\*\$

Required: No

#### **JobName**

The name that you assigned to the dominant language detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-@]\*\$

Required: No

#### **JobStatus**

The current status of the dominant language detection job. If the status is FAILED, the Message field shows the reason for the failure.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **Message**

A description for the status of a job.

Type: String

Required: No

#### **OutputDataConfig**

The output data configuration that you supplied when you created the dominant language detection job.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: No

#### **SubmitTime**

The time that the dominant language detection job was submitted for processing.

Type: Timestamp

Required: No

#### **VolumeKmsKeyId**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### VpcConfig

Configuration parameters for a private Virtual Private Cloud (VPC) containing the resources you are using for your dominant language detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## EndpointFilter

The filter used to determine which endpoints are returned. You can filter jobs on their name, model, status, or the date and time that they were created. You can only set one filter at a time.

### Contents

#### **CreationTimeAfter**

Specifies a date after which the returned endpoint or endpoints were created.

Type: Timestamp

Required: No

#### **CreationTimeBefore**

Specifies a date before which the returned endpoint or endpoints were created.

Type: Timestamp

Required: No

#### **ModelArn**

The Amazon Resource Number (ARN) of the model to which the endpoint is attached.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier|entity-recognizer)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

#### **Status**

Specifies the status of the endpoint being returned. Possible values are: Creating, Ready, Updating, Deleting, Failed.

Type: String

Valid Values: CREATING | DELETING | FAILED | IN\_SERVICE | UPDATING

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EndpointProperties

Specifies information about the specified endpoint.

## Contents

### **CreationTime**

The creation date and time of the endpoint.

Type: Timestamp

Required: No

### **CurrentInferenceUnits**

The number of inference units currently used by the model using this endpoint.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

### **DataAccessRoleArn**

The Amazon Resource Name (ARN) of the AWS identity and Access Management (IAM) role that grants Amazon Comprehend read access to trained custom models encrypted with a customer managed key (ModelKmsKeyId).

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+`

Required: No

### **DesiredDataAccessRoleArn**

Data access role ARN to use in case the new model is encrypted with a customer KMS key.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+`

Required: No

### **DesiredInferenceUnits**

The desired number of inference units to be used by the model using this endpoint. Each inference unit represents of a throughput of 100 characters per second.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

### **DesiredModelArn**

ARN of the new model to use for updating an existing endpoint. This ARN is going to be different from the model ARN when the update is in progress

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]\*)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier|entity-recognizer)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9]))?

Required: No

#### EndpointArn

The Amazon Resource Number (ARN) of the endpoint.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]\*)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier-endpoint|entity-recognizer-endpoint)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

Required: No

#### LastModifiedTime

The date and time that the endpoint was last modified.

Type: Timestamp

Required: No

#### Message

Specifies a reason for failure in cases of Failed status.

Type: String

Required: No

#### ModelArn

The Amazon Resource Number (ARN) of the model to which the endpoint is attached.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]\*)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:(document-classifier|entity-recognizer)/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9]))?

Required: No

#### Status

Specifies the status of the endpoint. Because the endpoint updates and creation are asynchronous, so customers will need to wait for the endpoint to be Ready status before making inference requests.

Type: String

Valid Values: CREATING | DELETING | FAILED | IN\_SERVICE | UPDATING

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## EntitiesDetectionJobFilter

Provides information for filtering a list of dominant language detection jobs. For more information, see the [ListEntitiesDetectionJobs \(p. 323\)](#) operation.

### Contents

#### JobName

Filters on the name of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

Required: No

#### JobStatus

Filters the list of jobs based on job status. Returns only jobs with the specified status.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### SubmitTimeAfter

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted after the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

#### SubmitTimeBefore

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted before the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntitiesDetectionJobProperties

Provides information about an entities detection job.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) that gives Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: No

### EndTime

The time that the entities detection job completed

Type: Timestamp

Required: No

### EntityRecognizerArn

The Amazon Resource Name (ARN) that identifies the entity recognizer.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:entity-recognizer/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

### InputDataConfig

The input data configuration that you supplied when you created the entities detection job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: No

### JobArn

The Amazon Resource Name (ARN) of the entities detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

arn:<partition>:comprehend:<region>:<account-id>:entities-detection-job/<job-id>

The following is an example job ARN:

arn:aws:comprehend:us-west-2:111122223333:entities-detection-job/1234abcd12ab34cd56ef1234567890ab

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

#### **JobId**

The identifier assigned to the entities detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/=+\-%@]\*)\$

Required: No

#### **JobName**

The name that you assigned the entities detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/=+\-%@]\*)\$

Required: No

#### **JobStatus**

The current status of the entities detection job. If the status is FAILED, the Message field shows the reason for the failure.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **LanguageCode**

The language code of the input documents.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: No

#### **Message**

A description of the status of a job.

Type: String

Required: No

#### **OutputDataConfig**

The output data configuration that you supplied when you created the entities detection job.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: No

**SubmitTime**

The time that the entities detection job was submitted for processing.

Type: Timestamp

Required: No

**VolumeKmsKeyId**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

**VpcConfig**

Configuration parameters for a private Virtual Private Cloud (VPC) containing the resources you are using for your entity detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## Entity

Provides information about an entity.

### Contents

#### **BeginOffset**

A character offset in the input text that shows where the entity begins (the first character is at position 0). The offset returns the position of each UTF-8 code point in the string. A *code point* is the abstract character from a particular graphical representation. For example, a multi-byte UTF-8 character maps to a single code point.

Type: Integer

Required: No

#### **EndOffset**

A character offset in the input text that shows where the entity ends. The offset returns the position of each UTF-8 code point in the string. A *code point* is the abstract character from a particular graphical representation. For example, a multi-byte UTF-8 character maps to a single code point.

Type: Integer

Required: No

#### **Score**

The level of confidence that Amazon Comprehend has in the accuracy of the detection.

Type: Float

Required: No

#### **Text**

The text of the entity.

Type: String

Length Constraints: Minimum length of 1.

Required: No

#### **Type**

The entity's type.

Type: String

Valid Values: PERSON | LOCATION | ORGANIZATION | COMMERCIAL\_ITEM | EVENT | DATE | QUANTITY | TITLE | OTHER

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## EntityLabel

Specifies one of the label or labels that categorize the personally identifiable information (PII) entity being analyzed.

### Contents

#### Name

The name of the label.

Type: String

Valid Values: BANK\_ACCOUNT\_NUMBER | BANK\_ROUTING | CREDIT\_DEBIT\_NUMBER | CREDIT\_DEBIT\_CVV | CREDIT\_DEBIT\_EXPIRY | PIN | EMAIL | ADDRESS | NAME | PHONE | SSN | DATE\_TIME | PASSPORT\_NUMBER | DRIVER\_ID | URL | AGE | USERNAME | PASSWORD | AWS\_ACCESS\_KEY | AWS\_SECRET\_KEY | IP\_ADDRESS | MAC\_ADDRESS

Required: Yes

#### Score

The level of confidence that Amazon Comprehend has in the accuracy of the detection.

Type: Float

Required: Yes

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerAnnotations

Describes the annotations associated with a entity recognizer.

## Contents

### S3Uri

Specifies the Amazon S3 location where the annotations for an entity recognizer are located. The URI must be in the same region as the API endpoint that you are calling.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-z0-9][\.\-\\_a-z0-9]{1,61}[a-z0-9](/.\*)?

Required: Yes

### TestS3Uri

This specifies the Amazon S3 location where the test annotations for an entity recognizer are located. The URI must be in the same AWS Region as the API endpoint that you are calling.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-z0-9][\.\-\\_a-z0-9]{1,61}[a-z0-9](/.\*)?

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerDocuments

Describes the training documents submitted with an entity recognizer.

## Contents

### **InputFormat**

Specifies how the text in an input file should be processed. This is optional, and the default is ONE\_DOC\_PER\_LINE. ONE\_DOC\_PER\_FILE - Each file is considered a separate document. Use this option when you are processing large documents, such as newspaper articles or scientific papers. ONE\_DOC\_PER\_LINE - Each line in a file is considered a separate document. Use this option when you are processing many short documents, such as text messages.

Type: String

Valid Values: ONE\_DOC\_PER\_FILE | ONE\_DOC\_PER\_LINE

Required: No

### **S3Uri**

Specifies the Amazon S3 location where the training documents for an entity recognizer are located. The URL must be in the same region as the API endpoint that you are calling.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-z0-9][\.-a-z0-9]{1,61}[a-z0-9](/.\*)?

Required: Yes

### **TestS3Uri**

Specifies the Amazon S3 location where the test documents for an entity recognizer are located. The URI must be in the same AWS Region as the API endpoint that you are calling.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-z0-9][\.-a-z0-9]{1,61}[a-z0-9](/.\*)?

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## EntityRecognizerEntityList

Describes the entity recognizer submitted with an entity recognizer.

### Contents

#### S3Uri

Specifies the Amazon S3 location where the entity list is located. The URI must be in the same region as the API endpoint that you are calling.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-zA-Z0-9][\.\-\w]{1,61}[a-zA-Z0-9](/.\*)?

Required: Yes

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerEvaluationMetrics

Detailed information about the accuracy of an entity recognizer.

## Contents

### F1Score

A measure of how accurate the recognizer results are for the test data. It is derived from the Precision and Recall values. The F1Score is the harmonic average of the two scores. The highest score is 1, and the worst score is 0.

Type: Double

Required: No

### Precision

A measure of the usefulness of the recognizer results in the test data. High precision means that the recognizer returned substantially more relevant results than irrelevant ones.

Type: Double

Required: No

### Recall

A measure of how complete the recognizer results are for the test data. High recall means that the recognizer returned most of the relevant results.

Type: Double

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerFilter

Provides information for filtering a list of entity recognizers. You can only specify one filtering parameter in a request. For more information, see the [ListEntityRecognizers \(p. 326\)](#) operation./>

## Contents

### RecognizerName

The name that you assigned the entity recognizer.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

### Status

The status of an entity recognizer.

Type: String

Valid Values: SUBMITTED | TRAINING | DELETING | STOP\_REQUESTED | STOPPED | IN\_ERROR | TRAINED

Required: No

### SubmitTimeAfter

Filters the list of entities based on the time that the list was submitted for processing. Returns only jobs submitted after the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

### SubmitTimeBefore

Filters the list of entities based on the time that the list was submitted for processing. Returns only jobs submitted before the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerInputDataConfig

Specifies the format and location of the input data.

## Contents

### Annotations

The S3 location of the CSV file that annotates your training documents.

Type: [EntityRecognizerAnnotations \(p. 455\)](#) object

Required: No

### AugmentedManifests

A list of augmented manifest files that provide training data for your custom model. An augmented manifest file is a labeled dataset that is produced by Amazon SageMaker Ground Truth.

This parameter is required if you set `DataFormat` to `AUGMENTED_MANIFEST`.

Type: Array of [AugmentedManifestsListItem \(p. 411\)](#) objects

Required: No

### DataFormat

The format of your training data:

- `COMPREHEND_CSV`: A CSV file that supplements your training documents. The CSV file contains information about the custom entities that your trained model will detect. The required format of the file depends on whether you are providing annotations or an entity list.

If you use this value, you must provide your CSV file by using either the `Annotations` or `EntityList` parameters. You must provide your training documents by using the `Documents` parameter.

- `AUGMENTED_MANIFEST`: A labeled dataset that is produced by Amazon SageMaker Ground Truth. This file is in JSON lines format. Each line is a complete JSON object that contains a training document and its labels. Each label annotates a named entity in the training document.

If you use this value, you must provide the `AugmentedManifests` parameter in your request.

If you don't specify a value, Amazon Comprehend uses `COMPREHEND_CSV` as the default.

Type: String

Valid Values: `COMPREHEND_CSV` | `AUGMENTED_MANIFEST`

Required: No

### Documents

The S3 location of the folder that contains the training documents for your custom entity recognizer.

This parameter is required if you set `DataFormat` to `COMPREHEND_CSV`.

Type: [EntityRecognizerDocuments \(p. 456\)](#) object

Required: No

### EntityList

The S3 location of the CSV file that has the entity list for your custom entity recognizer.

Type: [EntityRecognizerEntityList \(p. 457\)](#) object

Required: No

#### EntityTypes

The entity types in the labeled training data that Amazon Comprehend uses to train the custom entity recognizer. Any entity types that you don't specify are ignored.

A maximum of 25 entity types can be used at one time to train an entity recognizer. Entity types must not contain the following invalid characters: \n (line break), \\n (escaped line break), \r (carriage return), \\r (escaped carriage return), \t (tab), \\t (escaped tab), space, and , (comma).

Type: Array of [EntityTypeListItems \(p. 470\)](#) objects

Required: Yes

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerMetadata

Detailed information about an entity recognizer.

## Contents

### EntityTypes

Entity types from the metadata of an entity recognizer.

Type: Array of [EntityRecognizerMetadataEntityTypesListItem \(p. 463\)](#) objects

Required: No

### EvaluationMetrics

Detailed information about the accuracy of an entity recognizer.

Type: [EntityRecognizerEvaluationMetrics \(p. 458\)](#) object

Required: No

### NumberOfTestDocuments

The number of documents in the input data that were used to test the entity recognizer. Typically this is 10 to 20 percent of the input documents.

Type: Integer

Required: No

### NumberOfTrainedDocuments

The number of documents in the input data that were used to train the entity recognizer. Typically this is 80 to 90 percent of the input documents.

Type: Integer

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerMetadataEntityTypesListItem

Individual item from the list of entity types in the metadata of an entity recognizer.

## Contents

### EvaluationMetrics

Detailed information about the accuracy of the entity recognizer for a specific item on the list of entity types.

Type: [EntityTypeEvaluationMetrics \(p. 469\)](#) object

Required: No

### NumberOfTrainMentions

Indicates the number of times the given entity type was seen in the training data.

Type: Integer

Required: No

### Type

Type of entity from the list of entity types in the metadata of an entity recognizer.

Type: String

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerProperties

Describes information about an entity recognizer.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) of the AWS Identity and Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:aws(-[^\:]*)?:iam::[0-9]{12}:role/.+`

Required: No

### EndTime

The time that the recognizer creation completed.

Type: Timestamp

Required: No

### EntityRecognizerArn

The Amazon Resource Name (ARN) that identifies the entity recognizer.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^\:]*)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:entity-recognizer/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*)?`

Required: No

### InputDataConfig

The input data properties of an entity recognizer.

Type: [EntityRecognizerInputDataConfig \(p. 460\)](#) object

Required: No

### LanguageCode

The language of the input documents. All documents must be in the same language. Only English ("en") is currently supported.

Type: String

Valid Values: `en | es | fr | it | de | pt`

Required: No

### Message

A description of the status of the recognizer.

Type: String

Required: No

**ModelKmsKeyId**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt trained custom models. The ModelKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

**RecognizerMetadata**

Provides information about an entity recognizer.

Type: [EntityRecognizerMetadata \(p. 462\)](#) object

Required: No

**Status**

Provides the status of the entity recognizer.

Type: String

Valid Values: SUBMITTED | TRAINING | DELETING | STOP\_REQUESTED | STOPPED | IN\_ERROR | TRAINED

Required: No

**SubmitTime**

The time that the recognizer was submitted for processing.

Type: Timestamp

Required: No

**TrainingEndTime**

The time that training of the entity recognizer was completed.

Type: Timestamp

Required: No

**TrainingStartTime**

The time that training of the entity recognizer started.

Type: Timestamp

Required: No

**VersionName**

The version name you assigned to the entity recognizer.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[ a-zA-Z0-9 ](-\*[ a-zA-Z0-9 ])\*\$

Required: No

#### VolumeKmsKeyId

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### VpcConfig

Configuration parameters for a private Virtual Private Cloud (VPC) containing the resources you are using for your custom entity recognizer. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityRecognizerSummary

Describes the information about an entity recognizer and its versions.

## Contents

### **LatestVersionCreatedAt**

The time that the latest entity recognizer version was submitted for processing.

Type: Timestamp

Required: No

### **LatestVersionName**

The version name you assigned to the latest entity recognizer version.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

### **LatestVersionStatus**

Provides the status of the latest entity recognizer version.

Type: String

Valid Values: SUBMITTED | TRAINING | DELETING | STOP\_REQUESTED | STOPPED | IN\_ERROR | TRAINED

Required: No

### **NumberOfVersions**

The number of versions you created.

Type: Integer

Required: No

### **RecognizerName**

The name that you assigned the entity recognizer.

Type: String

Length Constraints: Maximum length of 63.

Pattern: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*\$

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EntityTypesEvaluationMetrics

Detailed information about the accuracy of an entity recognizer for a specific entity type.

## Contents

### F1Score

A measure of how accurate the recognizer results are for a specific entity type in the test data. It is derived from the Precision and Recall values. The F1Score is the harmonic average of the two scores. The highest score is 1, and the worst score is 0.

Type: Double

Required: No

### Precision

A measure of the usefulness of the recognizer results for a specific entity type in the test data. High precision means that the recognizer returned substantially more relevant results than irrelevant ones.

Type: Double

Required: No

### Recall

A measure of how complete the recognizer results are for a specific entity type in the test data. High recall means that the recognizer returned most of the relevant results.

Type: Double

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## EntityTypesListItem

An entity type within a labeled training dataset that Amazon Comprehend uses to train a custom entity recognizer.

### Contents

#### Type

An entity type within a labeled training dataset that Amazon Comprehend uses to train a custom entity recognizer.

Entity types must not contain the following invalid characters: \n (line break), \\n (escaped line break), \\r (carriage return), \\\r (escaped carriage return), \\t (tab), \\\t (escaped tab), space, and , (comma).

Type: String

Length Constraints: Maximum length of 64.

Pattern: ^(?:(!\\n+|\\t+|\\r+|[\r\t\n,])+)\$

Required: Yes

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## EventsDetectionJobFilter

Provides information for filtering a list of event detection jobs.

### Contents

#### **JobName**

Filters on the name of the events detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

Required: No

#### **JobStatus**

Filters the list of jobs based on job status. Returns only jobs with the specified status.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **SubmitTimeAfter**

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted after the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

#### **SubmitTimeBefore**

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted before the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# EventsDetectionJobProperties

Provides information about an events detection job.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that grants Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^\:]+)\:iam\::[0-9]{12}\:role/.+

Required: No

### EndTime

The time that the events detection job completed.

Type: Timestamp

Required: No

### InputDataConfig

The input data configuration that you supplied when you created the events detection job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: No

### JobArn

The Amazon Resource Name (ARN) of the events detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

arn:<partition>:comprehend:<region>:<account-id>:events-detection-job/<job-id>

The following is an example job ARN:

arn:aws:comprehend:us-west-2:111122223333:events-detection-job/1234abcd12ab34cd56ef1234567890ab

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^\:]+)\:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}\:[a-zA-Z0-9-]{1,64}\/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(./version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

### JobId

The identifier assigned to the events detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

Required: No

#### **JobName**

The name you assigned the events detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

Required: No

#### **JobStatus**

The current status of the events detection job.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **LanguageCode**

The language code of the input documents.

Type: String

Valid Values: en

Required: No

#### **Message**

A description of the status of the events detection job.

Type: String

Required: No

#### **OutputDataConfig**

The output data configuration that you supplied when you created the events detection job.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: No

#### **SubmitTime**

The time that the events detection job was submitted for processing.

Type: Timestamp

Required: No

#### **TargetEventTypes**

The types of events that are detected by the job.

Type: Array of strings

Array Members: Minimum number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 40.

Pattern: [A-Z\_]\*

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# InputDataConfig

The input properties for an inference job.

## Contents

### DocumentReaderConfig

The document reader config field applies only for InputDataConfig of StartEntitiesDetectionJob.

Use DocumentReaderConfig to provide specifications about how you want your inference documents read. Currently it applies for PDF documents in StartEntitiesDetectionJob custom inference.

Type: [DocumentReaderConfig \(p. 438\)](#) object

Required: No

### InputFormat

Specifies how the text in an input file should be processed:

- ONE\_DOC\_PER\_FILE - Each file is considered a separate document. Use this option when you are processing large documents, such as newspaper articles or scientific papers.
- ONE\_DOC\_PER\_LINE - Each line in a file is considered a separate document. Use this option when you are processing many short documents, such as text messages.

Type: String

Valid Values: ONE\_DOC\_PER\_FILE | ONE\_DOC\_PER\_LINE

Required: No

### S3Uri

The Amazon S3 URI for the input data. The URI must be in same region as the API endpoint that you are calling. The URI can point to a single input file or it can provide the prefix for a collection of data files.

For example, if you use the URI `s3://bucketName/prefix`, if the prefix is a single file, Amazon Comprehend uses that file as input. If more than one file begins with the prefix, Amazon Comprehend uses all of them as input.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: `s3://[a-z0-9][\.\-\_a-z0-9]{1,61}[a-z0-9](/.*)?`

Required: Yes

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)



## KeyPhrase

Describes a key noun phrase.

### Contents

#### **BeginOffset**

A character offset in the input text that shows where the key phrase begins (the first character is at position 0). The offset returns the position of each UTF-8 code point in the string. A *code point* is the abstract character from a particular graphical representation. For example, a multi-byte UTF-8 character maps to a single code point.

Type: Integer

Required: No

#### **EndOffset**

A character offset in the input text where the key phrase ends. The offset returns the position of each UTF-8 code point in the string. A *code point* is the abstract character from a particular graphical representation. For example, a multi-byte UTF-8 character maps to a single code point.

Type: Integer

Required: No

#### **Score**

The level of confidence that Amazon Comprehend has in the accuracy of the detection.

Type: Float

Required: No

#### **Text**

The text of a key noun phrase.

Type: String

Length Constraints: Minimum length of 1.

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## KeyPhrasesDetectionJobFilter

Provides information for filtering a list of dominant language detection jobs. For more information, see the [ListKeyPhrasesDetectionJobs \(p. 335\)](#) operation.

### Contents

#### JobName

Filters on the name of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\*\$

Required: No

#### JobStatus

Filters the list of jobs based on job status. Returns only jobs with the specified status.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### SubmitTimeAfter

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted after the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

#### SubmitTimeBefore

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted before the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# KeyPhrasesDetectionJobProperties

Provides information about a key phrases detection job.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) that gives Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: No

### EndTime

The time that the key phrases detection job completed.

Type: Timestamp

Required: No

### InputDataConfig

The input data configuration that you supplied when you created the key phrases detection job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: No

### JobArn

The Amazon Resource Name (ARN) of the key phrases detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

arn:<partition>:comprehend:<region>:<account-id>:key-phrases-detection-job/<job-id>

The following is an example job ARN:

arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-job/1234abcd12ab34cd56ef1234567890ab

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

### JobId

The identifier assigned to the key phrases detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\* )\$

Required: No

#### **JobName**

The name that you assigned the key phrases detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\* )\$

Required: No

#### **JobStatus**

The current status of the key phrases detection job. If the status is FAILED, the Message field shows the reason for the failure.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **LanguageCode**

The language code of the input documents.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: No

#### **Message**

A description of the status of a job.

Type: String

Required: No

#### **OutputDataConfig**

The output data configuration that you supplied when you created the key phrases detection job.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: No

#### **SubmitTime**

The time that the key phrases detection job was submitted for processing.

Type: Timestamp

Required: No

#### **VolumeKmsKeyId**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:11112223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### VpcConfig

Configuration parameters for a private Virtual Private Cloud (VPC) containing the resources you are using for your key phrases detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# OutputDataConfig

Provides configuration parameters for the output of topic detection jobs.

## Contents

### KmsKeyId

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt the output results from an analysis job. The KmsKeyId can be one of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
- KMS Key Alias: "alias/ExampleAlias"
- ARN of a KMS Key Alias: "arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

### S3Uri

When you use the OutputDataConfig object with asynchronous operations, you specify the Amazon S3 location where you want to write the output data. The URI must be in the same region as the API endpoint that you are calling. The location is used as the prefix for the actual location of the output file.

When the topic detection job is finished, the service creates an output file in a directory specific to the job. The S3Uri field contains the location of the output file, called `output.tar.gz`. It is a compressed archive that contains the ouput of the operation.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-z0-9][\.\-\\_a-z0-9]{1,61}[a-z0-9](/.\*)?

Required: Yes

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## PartOfSpeechTag

Identifies the part of speech represented by the token and gives the confidence that Amazon Comprehend has that the part of speech was correctly identified. For more information about the parts of speech that Amazon Comprehend can identify, see [Analyze Syntax \(p. 112\)](#).

### Contents

#### Score

The confidence that Amazon Comprehend has that the part of speech was correctly identified.

Type: Float

Required: No

#### Tag

Identifies the part of speech that the token represents.

Type: String

Valid Values: ADJ | ADP | ADV | AUX | CONJ | CCONJ | DET | INTJ | NOUN | NUM | O | PART | PRON | PROPN | PUNCT | SCONJ | SYM | VERB

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## PiiEntitiesDetectionJobFilter

Provides information for filtering a list of PII entity detection jobs.

### Contents

#### **JobName**

Filters on the name of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\\-%@]\*\$

Required: No

#### **JobStatus**

Filters the list of jobs based on job status. Returns only jobs with the specified status.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **SubmitTimeAfter**

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted after the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

#### **SubmitTimeBefore**

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted before the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# PiiEntitiesDetectionJobProperties

Provides information about a PII entities detection job.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) that gives Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:aws(-[^\:]+)?::iam::[0-9]{12}:role/.+`

Required: No

### EndTime

The time that the PII entities detection job completed.

Type: Timestamp

Required: No

### InputDataConfig

The input properties for a PII entities detection job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: No

### JobArn

The Amazon Resource Name (ARN) of the PII entities detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

```
arn:<partition>:comprehend:<region>:<account-id>:pii-entities-detection-job/<job-id>
```

The following is an example job ARN:

```
arn:aws:comprehend:us-west-2:111122223333:pii-entities-detection-job/1234abcd12ab34cd56ef1234567890ab
```

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^\:]+)?::comprehend:[a-zA-Z0-9-]*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*?)?`

Required: No

### JobId

The identifier assigned to the PII entities detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-+\\-%@]\*\$

Required: No

#### **JobName**

The name that you assigned the PII entities detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:/-+\\-%@]\*\$

Required: No

#### **JobStatus**

The current status of the PII entities detection job. If the status is FAILED, the Message field shows the reason for the failure.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **LanguageCode**

The language code of the input documents

Type: String

Valid Values: en

Required: No

#### **Message**

A description of the status of a job.

Type: String

Required: No

#### **Mode**

Specifies whether the output provides the locations (offsets) of PII entities or a file in which PII entities are redacted.

Type: String

Valid Values: ONLY\_REDACTION | ONLY\_OFFSETS

Required: No

#### **OutputDataConfig**

The output data configuration that you supplied when you created the PII entities detection job.

Type: [PiiOutputDataConfig \(p. 490\)](#) object

Required: No

### **RedactionConfig**

Provides configuration parameters for PII entity redaction.

This parameter is required if you set the `Mode` parameter to `ONLY_REDACTION`. In that case, you must provide a `RedactionConfig` definition that includes the `PiiEntityTypes` parameter.

Type: [RedactionConfig \(p. 491\)](#) object

Required: No

### **SubmitTime**

The time that the PII entities detection job was submitted for processing.

Type: `Timestamp`

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## PiiEntity

Provides information about a PII entity.

### Contents

#### BeginOffset

A character offset in the input text that shows where the PII entity begins (the first character is at position 0). The offset returns the position of each UTF-8 code point in the string. A *code point* is the abstract character from a particular graphical representation. For example, a multi-byte UTF-8 character maps to a single code point.

Type: Integer

Required: No

#### EndOffset

A character offset in the input text that shows where the PII entity ends. The offset returns the position of each UTF-8 code point in the string. A *code point* is the abstract character from a particular graphical representation. For example, a multi-byte UTF-8 character maps to a single code point.

Type: Integer

Required: No

#### Score

The level of confidence that Amazon Comprehend has in the accuracy of the detection.

Type: Float

Required: No

#### Type

The entity's type.

Type: String

Valid Values: BANK\_ACCOUNT\_NUMBER | BANK\_ROUTING | CREDIT\_DEBIT\_NUMBER | CREDIT\_DEBIT\_CVV | CREDIT\_DEBIT\_EXPIRY | PIN | EMAIL | ADDRESS | NAME | PHONE | SSN | DATE\_TIME | PASSPORT\_NUMBER | DRIVER\_ID | URL | AGE | USERNAME | PASSWORD | AWS\_ACCESS\_KEY | AWS\_SECRET\_KEY | IP\_ADDRESS | MAC\_ADDRESS | LICENSE\_PLATE | VEHICLE\_IDENTIFICATION\_NUMBER | UK\_NATIONAL\_INSURANCE\_NUMBER | CA\_SOCIAL\_INSURANCE\_NUMBER | US\_INDIVIDUAL\_TAX\_IDENTIFICATION\_NUMBER | UK\_UNIQUE\_TAXPAYER\_REFERENCE\_NUMBER | IN\_PERMANENT\_ACCOUNT\_NUMBER | IN\_NREGA | INTERNATIONAL\_BANK\_ACCOUNT\_NUMBER | SWIFT\_CODE | UK\_NATIONAL\_HEALTH\_SERVICE\_NUMBER | CA\_HEALTH\_NUMBER | IN\_AADHAAR | UK\_VOTER\_NUMBER | IN\_VOTER\_NUMBER | ALL

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# PiiOutputDataConfig

Provides configuration parameters for the output of PII entity detection jobs.

## Contents

### KmsKeyId

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt the output results from an analysis job.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

### S3Uri

When you use the `PiiOutputDataConfig` object with asynchronous operations, you specify the Amazon S3 location where you want to write the output data.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: s3://[a-z0-9][\.\-\\_a-z0-9]{1,61}[a-z0-9](/.\*)?

Required: Yes

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## RedactionConfig

Provides configuration parameters for PII entity redaction.

### Contents

#### MaskCharacter

A character that replaces each character in the redacted PII entity.

Type: String

Length Constraints: Fixed length of 1.

Pattern: [ !@#\$%&\* ]

Required: No

#### MaskMode

Specifies whether the PII entity is redacted with the mask character or the entity type.

Type: String

Valid Values: MASK | REPLACE\_WITH\_PII\_ENTITY\_TYPE

Required: No

#### PiiEntityTypes

An array of the types of PII entities that Amazon Comprehend detects in the input text for your request.

Type: Array of strings

Valid Values: BANK\_ACCOUNT\_NUMBER | BANK\_ROUTING | CREDIT\_DEBIT\_NUMBER | CREDIT\_DEBIT\_CVV | CREDIT\_DEBIT\_EXPIRY | PIN | EMAIL | ADDRESS | NAME | PHONE | SSN | DATE\_TIME | PASSPORT\_NUMBER | DRIVER\_ID | URL | AGE | USERNAME | PASSWORD | AWS\_ACCESS\_KEY | AWS\_SECRET\_KEY | IP\_ADDRESS | MAC\_ADDRESS | LICENSE\_PLATE | VEHICLE\_IDENTIFICATION\_NUMBER | UK\_NATIONAL\_INSURANCE\_NUMBER | CA\_SOCIAL\_INSURANCE\_NUMBER | US\_INDIVIDUAL\_TAX\_IDENTIFICATION\_NUMBER | UK\_UNIQUE\_TAXPAYER\_REFERENCE\_NUMBER | IN\_PERMANENT\_ACCOUNT\_NUMBER | IN\_NREGA | INTERNATIONAL\_BANK\_ACCOUNT\_NUMBER | SWIFT\_CODE | UK\_NATIONAL\_HEALTH\_SERVICE\_NUMBER | CA\_HEALTH\_NUMBER | IN\_AADHAAR | UK\_VOTER\_NUMBER | IN\_VOTER\_NUMBER | ALL

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## SentimentDetectionJobFilter

Provides information for filtering a list of dominant language detection jobs. For more information, see the [ListSentimentDetectionJobs \(p. 341\)](#) operation.

### Contents

#### JobName

Filters on the name of the job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+%-@]\*\$

Required: No

#### JobStatus

Filters the list of jobs based on job status. Returns only jobs with the specified status.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### SubmitTimeAfter

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted after the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

#### SubmitTimeBefore

Filters the list of jobs based on the time that the job was submitted for processing. Returns only jobs submitted before the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# SentimentDetectionJobProperties

Provides information about a sentiment detection job.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) that gives Amazon Comprehend read access to your input data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: arn:aws(-[^:]+)?:iam::[0-9]{12}:role/.+

Required: No

### EndTime

The time that the sentiment detection job ended.

Type: Timestamp

Required: No

### InputDataConfig

The input data configuration that you supplied when you created the sentiment detection job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: No

### JobArn

The Amazon Resource Name (ARN) of the sentiment detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

arn:<partition>:comprehend:<region>:<account-id>:sentiment-detection-job/<job-id>

The following is an example job ARN:

arn:aws:comprehend:us-west-2:111122223333:sentiment-detection-job/1234abcd12ab34cd56ef1234567890ab

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:aws(-[^:]+)?:comprehend:[a-zA-Z0-9-]\*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*(/version/[a-zA-Z0-9](-\*[a-zA-Z0-9])\*)?

Required: No

### JobId

The identifier assigned to the sentiment detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\* )\$

Required: No

#### **JobName**

The name that you assigned to the sentiment detection job

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\* )\$

Required: No

#### **JobStatus**

The current status of the sentiment detection job. If the status is FAILED, the Messages field shows the reason for the failure.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **LanguageCode**

The language code of the input documents.

Type: String

Valid Values: en | es | fr | de | it | pt | ar | hi | ja | ko | zh | zh-TW

Required: No

#### **Message**

A description of the status of a job.

Type: String

Required: No

#### **OutputDataConfig**

The output data configuration that you supplied when you created the sentiment detection job.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: No

#### **SubmitTime**

The time that the sentiment detection job was submitted for processing.

Type: Timestamp

Required: No

#### **VolumeKmsKeyId**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### VpcConfig

Configuration parameters for a private Virtual Private Cloud (VPC) containing the resources you are using for your sentiment detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## SentimentScore

Describes the level of confidence that Amazon Comprehend has in the accuracy of its detection of sentiments.

### Contents

#### Mixed

The level of confidence that Amazon Comprehend has in the accuracy of its detection of the `MIXED` sentiment.

Type: Float

Required: No

#### Negative

The level of confidence that Amazon Comprehend has in the accuracy of its detection of the `NEGATIVE` sentiment.

Type: Float

Required: No

#### Neutral

The level of confidence that Amazon Comprehend has in the accuracy of its detection of the `NEUTRAL` sentiment.

Type: Float

Required: No

#### Positive

The level of confidence that Amazon Comprehend has in the accuracy of its detection of the `POSITIVE` sentiment.

Type: Float

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## SyntaxToken

Represents a work in the input text that was recognized and assigned a part of speech. There is one syntax token record for each word in the source text.

### Contents

#### **BeginOffset**

The zero-based offset from the beginning of the source text to the first character in the word.

Type: Integer

Required: No

#### **EndOffset**

The zero-based offset from the beginning of the source text to the last character in the word.

Type: Integer

Required: No

#### **PartOfSpeech**

Provides the part of speech label and the confidence level that Amazon Comprehend has that the part of speech was correctly identified. For more information, see [Analyze Syntax \(p. 112\)](#).

Type: [PartOfSpeechTag \(p. 483\)](#) object

Required: No

#### **Text**

The word that was recognized in the source text.

Type: String

Length Constraints: Minimum length of 1.

Required: No

#### **TokenId**

A unique identifier for a token.

Type: Integer

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## Tag

A key-value pair that adds as a metadata to a resource used by Amazon Comprehend. For example, a tag with the key-value pair 'Department':'Sales' might be added to a resource to indicate its use by a particular department.

### Contents

#### Key

The initial part of a key-value pair that forms a tag associated with a given resource. For instance, if you want to show which resources are used by which departments, you might use "Department" as the key portion of the pair, with multiple possible values such as "sales," "legal," and "administration."

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Required: Yes

#### Value

The second part of a key-value pair that forms a tag associated with a given resource. For instance, if you want to show which resources are used by which departments, you might use "Department" as the initial (key) portion of the pair, with a value of "sales" to indicate the sales department.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Required: No

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# TopicsDetectionJobFilter

Provides information for filtering topic detection jobs. For more information, see [ListTopicsDetectionJobs \(p. 346\)](#).

## Contents

### JobName

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/+\-%@]\*\$

Required: No

### JobStatus

Filters the list of topic detection jobs based on job status. Returns only jobs with the specified status.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

### SubmitTimeAfter

Filters the list of jobs based on the time that the job was submitted for processing. Only returns jobs submitted after the specified time. Jobs are returned in ascending order, oldest to newest.

Type: Timestamp

Required: No

### SubmitTimeBefore

Filters the list of jobs based on the time that the job was submitted for processing. Only returns jobs submitted before the specified time. Jobs are returned in descending order, newest to oldest.

Type: Timestamp

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

# TopicsDetectionJobProperties

Provides information about a topic detection job.

## Contents

### DataAccessRoleArn

The Amazon Resource Name (ARN) of the AWS Identity and Management (IAM) role that grants Amazon Comprehend read access to your job data.

Type: String

Length Constraints: Minimum length of 20. Maximum length of 2048.

Pattern: `arn:aws(-[^\:]*)?:iam::[0-9]{12}:role/.+`

Required: No

### EndTime

The time that the topic detection job was completed.

Type: Timestamp

Required: No

### InputDataConfig

The input data configuration supplied when you created the topic detection job.

Type: [InputDataConfig \(p. 475\)](#) object

Required: No

### JobArn

The Amazon Resource Name (ARN) of the topics detection job. It is a unique, fully qualified identifier for the job. It includes the AWS account, Region, and the job ID. The format of the ARN is as follows:

`arn:<partition>:comprehend:<region>:<account-id>:topics-detection-job/<job-id>`

The following is an example job ARN:

`arn:aws:comprehend:us-west-2:111122223333:topics-detection-job/1234abcd12ab34cd56ef1234567890ab`

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:aws(-[^\:]*)?:comprehend:[a-zA-Z0-9-]*:[0-9]{12}:[a-zA-Z0-9-]{1,64}/[a-zA-Z0-9](-*[a-zA-Z0-9])*(/version/[a-zA-Z0-9](-*[a-zA-Z0-9])*?)?`

Required: No

### JobId

The identifier assigned to the topic detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\*\$

Required: No

#### **JobName**

The name of the topic detection job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: ^([\p{L}\p{Z}\p{N}\_.:=/=+\-%@]\*\$

Required: No

#### **JobStatus**

The current status of the topic detection job. If the status is Failed, the reason for the failure is shown in the Message field.

Type: String

Valid Values: SUBMITTED | IN\_PROGRESS | COMPLETED | FAILED | STOP\_REQUESTED | STOPPED

Required: No

#### **Message**

A description for the status of a job.

Type: String

Required: No

#### **NumberOfTopics**

The number of topics to detect supplied when you created the topic detection job. The default is 10.

Type: Integer

Required: No

#### **OutputDataConfig**

The output data configuration supplied when you created the topic detection job.

Type: [OutputDataConfig \(p. 482\)](#) object

Required: No

#### **SubmitTime**

The time that the topic detection job was submitted for processing.

Type: Timestamp

Required: No

#### **VolumeKmsKeyId**

ID for the AWS Key Management Service (KMS) key that Amazon Comprehend uses to encrypt data on the storage volume attached to the ML compute instance(s) that process the analysis job. The VolumeKmsKeyId can be either of the following formats:

- KMS Key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- Amazon Resource Name (ARN) of a KMS Key: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

Type: String

Length Constraints: Maximum length of 2048.

Pattern: .\*

Required: No

#### VpcConfig

Configuration parameters for a private Virtual Private Cloud (VPC) containing the resources you are using for your topic detection job. For more information, see [Amazon VPC](#).

Type: [VpcConfig \(p. 503\)](#) object

Required: No

## See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## VpcConfig

Configuration parameters for an optional private Virtual Private Cloud (VPC) containing the resources you are using for the job. For more information, see [Amazon VPC](#).

### Contents

#### SecurityGroupIds

The ID number for a security group on an instance of your private VPC. Security groups on your VPC function serve as a virtual firewall to control inbound and outbound traffic and provides security for the resources that you'll be accessing on the VPC. This ID number is preceded by "sg-", for instance: "sg-03b388029b0a285ea". For more information, see [Security Groups for your VPC](#).

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: [-0-9a-zA-Z]+

Required: Yes

#### Subnets

The ID for each subnet being used in your private VPC. This subnet is a subset of the a range of IPv4 addresses used by the VPC and is specific to a given availability zone in the VPC's region. This ID number is preceded by "subnet-", for instance: "subnet-04ccf456919e69055". For more information, see [VPCs and Subnets](#).

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 16 items.

Length Constraints: Minimum length of 1. Maximum length of 32.

Pattern: [-0-9a-zA-Z]+

Required: Yes

### See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

## Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

**AccessDeniedException**

You do not have sufficient access to perform this action.

HTTP Status Code: 400

**IncompleteSignature**

The request signature does not conform to AWS standards.

HTTP Status Code: 400

**InternalFailure**

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

**InvalidAction**

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

**InvalidClientTokenId**

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

**InvalidParameterCombination**

Parameters that must not be used together were used together.

HTTP Status Code: 400

**InvalidParameterValue**

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

**InvalidQueryParameter**

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

**MalformedQueryString**

The query string contains a syntax error.

HTTP Status Code: 404

**MissingAction**

The request is missing an action or a required parameter.

HTTP Status Code: 400

**MissingAuthenticationToken**

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

**MissingParameter**

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

**NotAuthorized**

You do not have permission to perform this action.

HTTP Status Code: 400

**OptInRequired**

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

**RequestExpired**

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

**ServiceUnavailable**

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

**ThrottlingException**

The request was denied due to request throttling.

HTTP Status Code: 400

**ValidationException**

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

## Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see [Signature Version 4 Signing Process](#) in the *Amazon Web Services General Reference*.

**Action**

The action to be performed.

Type: string

Required: Yes

**Version**

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

**X-Amz-Algorithm**

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: AWS4-HMAC-SHA256

Required: Conditional

**X-Amz-Credential**

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4\_request"). The value is expressed in the following format: *access\_key/YYYYMMDD/region/service/aws4\_request*.

For more information, see [Task 2: Create a String to Sign for Signature Version 4](#) in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

**X-Amz-Date**

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: 20120325T120000Z.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see [Handling Dates in Signature Version 4](#) in the *Amazon Web Services General Reference*.

Type: string

Required: Conditional

**X-Amz-Security-Token**

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS Security Token Service, go to [AWS Services That Work with IAM](#) in the *IAM User Guide*.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

Required: Conditional

**X-Amz-Signature**

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

**X-Amz-SignedHeaders**

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see [Task 1: Create a Canonical Request For Signature Version 4](#) in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

# Document History for Amazon Comprehend

The following table describes the documentation for this release of Amazon Comprehend.

| update-history-change | update-history-description                                                                                                                                                                                                                                                 | update-history-date |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| New feature           | You can now use AWS Trusted Advisor to view recommendations that can help you optimize the cost and security of your Amazon Comprehend endpoints. For more information, see <a href="#">Using Trusted Advisor with Amazon Comprehend</a> .                                 | September 29, 2021  |
| New feature           | Amazon Comprehend has launched a suite of features for Comprehend Custom which enable continuous model improvements by giving you the ability to create new model versions, continuously test on specific test sets, and perform live migration to new model endpoints.    | September 21, 2021  |
| New feature           | Amazon Comprehend now allows you to analyze PDF and Word documents for custom entity recognition. With PDF and Word formats, you can extract information from documents containing headers, lists and tables.                                                              | September 14, 2021  |
| New feature           | Amazon Comprehend has launched a new endpoints overview feature which provides you a global view of your endpoints. From the endpoints overview page, you can view all of your endpoints in one place to understand your endpoint usage versus your actual resource usage. | August 24, 2021     |
| New feature           | Amazon Comprehend Medical now allows you to establish a private connection with your Virtual Private Cloud (VPC) by creating an interface VPC                                                                                                                              | June 13, 2021       |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                             |                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
|                    | endpoint. For more information, see <a href="#">VPC endpoints(PrivateLink)</a> .                                                                                                                                                                                                                                                                                                                            |                   |
| Language expansion | Amazon Comprehend has added four additional languages for the dominant language feature: Hausa (ha), Lao (lo), Maltese (mt), and Oromo (om). For more information, see <a href="#">Supported Languages in Amazon Comprehend</a> .                                                                                                                                                                           | May 10, 2021      |
| New feature        | With Amazon Comprehend, you can now encrypt custom models using a customer managed key (CMK). For more information, see <a href="#">KMS Encryption in Amazon Comprehend</a> .                                                                                                                                                                                                                               | March 31, 2021    |
| New feature        | You can now use Amazon S3 Object Lambda Access Points to configure how documents that contain personally identifiable information (PII) are retrieved from your Amazon S3 bucket. You can control access of documents that contain PII and redact PII from documents. For more information, see <a href="#">Using Amazon S3 Object Lambda Access Points for Personally Identifiable Information (PII)</a> . | March 18, 2021    |
| New feature        | You can now label a document with personally identifiable information (PII). Amazon Comprehend can analyze your document for the presence of PII and return the labels of identified PII entity types such as name, address, bank account number, or phone number. For more information, see <a href="#">Label Document with PII</a> .                                                                      | March 11, 2021    |
| New feature        | With Amazon Comprehend, you can now detect events in a set of documents. When you create an asynchronous events detection job, Amazon Comprehend can detect supported types of financial events. For more information, see <a href="#">Detect Events</a> .                                                                                                                                                  | November 24, 2020 |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                  |                    |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| New feature  | Amazon Comprehend now allows you to use auto scaling for custom entity recognizer endpoints. With auto scaling, you can automatically set endpoint provisioning to fit your capacity needs. For more information, see <a href="#">Auto Scaling with Endpoints</a> .                                                                                                                                                              | September 28, 2020 |
| New feature  | To train custom classifiers or entity recognizers, you can now provide augmented manifest files, which are labeled datasets that are produced by Amazon SageMaker Ground Truth. For more information about these files, and for examples, see <a href="#">Multi-Class Mode, Multi-Label Mode</a> , and <a href="#">Annotations</a> .                                                                                             | September 22, 2020 |
| New tutorial | Amazon Comprehend now has a tutorial that walks you through a multi-service workflow of analyzing customer reviews and visualizing the analysis results. For more information, see <a href="#">Tutorial: Analyzing Insights from Reviews</a> .                                                                                                                                                                                   | September 17, 2020 |
| New feature  | With Amazon Comprehend, you can now detect entities in your text that contain personally identifiable information (PII), such as addresses, bank account numbers, or phone numbers. Amazon Comprehend can provide the location of each PII entity in your text, or it can provide a copy of your text in which the PII is redacted. For more information, see <a href="#">Detect Personally Identifiable Information (PII)</a> . | September 17, 2020 |
| New feature  | Previously, you could only train a model on up to 12 custom entities. Now Amazon Comprehend allows you to train a model on up to 25 custom entities at a time. For more information, see <a href="#">Custom Entity Recognition</a> .                                                                                                                                                                                             | August 12, 2020    |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                   |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| Language expansion | Amazon Comprehend has added five additional languages for the custom entity recognition feature: German (de), Spanish (es), French (fr), Italian (it), and Portuguese (pt). For more information, see <a href="#">Supported Languages in Amazon Comprehend</a> .                                                                                                                                                                                                                                                             | August 12, 2020   |
| New feature        | Amazon Comprehend now allows you to establish a private connection with your Virtual Private Cloud (VPC) by creating an interface VPC endpoint. For more information, see <a href="#">VPC endpoints (AWS PrivateLink)</a> .                                                                                                                                                                                                                                                                                                  | August 11, 2020   |
| New feature        | With Amazon Comprehend, you can now quickly detect custom entities in individual text documents by running real-time analysis. For more information, see <a href="#">Detecting Custom Entities in Real Time with Amazon Comprehend</a> .                                                                                                                                                                                                                                                                                     | July 9, 2020      |
| New feature added  | Amazon Comprehend now provides support for a second mode in asynchronous Custom Classification for documents that provides greater flexibility when applying custom classes to documents. While multi-class mode associates only a single class with each document, the new multi-label mode can associate more than one. For example, a movie can be classified as both science fiction and action at the same time. For more information, see <a href="#">Multi-Class and Multi-Label Modes in Custom Classification</a> . | December 19, 2019 |
| New feature added  | Amazon Comprehend now provides support for real-time Custom Classification for documents with unstructured text. Customers can use real-time custom classification to understand, label and route information based on their own business rules, synchronously. For more information, see <a href="#">Real-time Analysis with Custom Classification</a> .                                                                                                                                                                    | November 25, 2019 |

|                     |                                                                                                                                                                                                                                                                                                                                                                                                 |                  |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| New languages added | Amazon Comprehend has added six additional languages for several of its features: Arabic (ar), Hindi (hi), Japanese (ja), Korean (ko), simplified Chinese (zh), and traditional Chinese (zh-TW). These new languages are supported only for Determine Sentiment, Detect Key Phrases, and non-custom Detect Entities operations. For more information, see <a href="#">Supported Languages</a> . | November 6, 2019 |
| New feature         | Previously, you could only train a model on a single custom entity. As a result, you could only search for that one entity with an entity recognition operation. Amazon Comprehend has changed this and you can now train a model on up to 12 custom entities at a time. For more information, see <a href="#">Custom Entity Recognition</a>                                                    | July 9, 2019     |
| New feature         | Amazon Comprehend now provides a multi-class confusion matrix for added ability to analyze metrics when training a Custom Classifier. This is currently supported using the APIs only. For more information, see <a href="#">Tagging Resources in Amazon Comprehend</a>                                                                                                                         | April 5, 2019    |
| New feature         | Amazon Comprehend provides tags for Custom Classifiers and Custom Entity Recognizers, which can be used as metadata that enables you to organize, filter, and control access to your resources with a finer level of control than ever. For more information, see <a href="#">Tagging Resources in Amazon Comprehend</a>                                                                        | April 3, 2019    |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                |                   |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| New feature        | Amazon S3 already enables you to encrypt your input documents, and Amazon Comprehend extends this even farther. By using your own KMS key, you can not only encrypt the output results of your job, but also the data on the storage volume attached to the compute instance that processes the analysis job. The result is end-to-end security. For more information, see <a href="#">KMS Encryption in Amazon Comprehend</a> | March 28, 2019    |
| New feature        | Custom entity recognition extends the capability of Amazon Comprehend by enabling you to identify new entity types not supported as one of the preset generic entity types. This means you can analyze documents and extract entities like product codes or business-specific entities that fit your particular needs. For more information, see <a href="#">Custom Entity Recognition</a>                                     | November 16, 2018 |
| New feature        | You can use Amazon Comprehend to build your own models for custom classification, assigning a document to a class or a category. For more information, see <a href="#">Document Classification</a> .                                                                                                                                                                                                                           | November 15, 2018 |
| Region expansion   | Amazon Comprehend is now available in Europe (Frankfurt) (eu-central-1).                                                                                                                                                                                                                                                                                                                                                       | October 10, 2018  |
| Language expansion | In addition to English and Spanish Amazon Comprehend can now also examine documents in French, German, Italian, and Portuguese. For more information, see <a href="#">Supported Languages in Amazon Comprehend</a> .                                                                                                                                                                                                           | October 10, 2018  |
| Region expansion   | Amazon Comprehend is now available in Asia Pacific (Sydney) (ap-southeast-2).                                                                                                                                                                                                                                                                                                                                                  | August 15, 2018   |

|                    |                                                                                                                                                                                                   |                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| New feature        | Amazon Comprehend now parses documents to discover the syntax of a document and the part of speech for each word. For more information, see <a href="#">Syntax</a> .                              | July 17, 2018     |
| New feature        | Amazon Comprehend now supports asynchronous batch processing for language, key phrase, entity, and sentiment detection. For more information, see <a href="#">Asynchronous Batch Processing</a> . | June 27, 2018     |
| New guide (p. 508) | This is the first release of the <i>Amazon Comprehend Developer Guide</i> .                                                                                                                       | November 29, 2017 |