Amazon Kendra

Developer Guide

Amazon Kendra: Developer Guide

Table of Contents

Amazon Kendra is in preview release. This documentation is subject to change.

What Is Amazon Kendra?

Amazon Kendra is an enterprise search service, powered by machine learning, that enables your users to intuitively search unstructured data using natural language. It returns specific answers to questions, giving users an experience that's close to interacting with a human expert. It is highly available and scalable, tightly integrated with other AWS services, and offers enterprise-grade security.

Types of Queries

Amazon Kendra users can express queries in a variety of formats.

- **Factoid questions** — who, what, when, or where questions such as *Who is Amazon's CEO?* or *What is the height of the Space Needle*. These require fact-based answers that can be returned in the form of a single word or phrase. The precise answer, however, must be explicitly stated in the ingested text content.
- **Descriptive questions** — Questions where the answer could be a sentence, passage, or an entire document. For example, *How do I connect my Echo Plus to my network?* or *How do I obtain tax benefits for lower income families?*.
- **Keyword searches** — For questions where the intent and scope isn't clear, Amazon Kendra uses its deep learning models to return relevant documents. For example, *vacation policy* or *health benefits*.

Benefits of Amazon Kendra

The benefits of Amazon Kendra are:

- **Accuracy** — Unlike traditional keyword search where results are based on basic keyword matching and ranking, Amazon Kendra goes deeper to understand the content, the user context, and the question. Amazon Kendra connects the dots across your data and goes beyond traditional search to return the most relevant word, snippet, or document for your query. Amazon Kendra uses machine learning to improve search results over time.
- **Simplicity** — Amazon Kendra provides a console and API for managing the documents that you want to search. A simple search API is provided for you to integrate into your client applications, such as websites or mobile applications.
- **Connectivity** — Amazon Kendra can connect to third-party data sources to provide search across documents managed in different environments.

Pricing for Amazon Kendra

You can start a 30-day trial of Amazon Kendra Enterprise that includes one provisioned index at no cost. After your trial expires, you are charged for all provisioned Amazon Kendra indexes, even if they don't contain documents for querying and no queries are executed. After the trial expires, there are additional charges for scanning and syncing documents using the Amazon Kendra connectors.

For a complete list of charges and prices, see Amazon Kendra pricing.

Are You a First-Time Amazon Kendra User?

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

1. How it Works (p. 3) – This section introduces various Amazon Kendra components that you work with to create a search solution.
2. Getting Started (p. 10) – In this section you set up your account and test the Amazon Kendra search API.
3. Creating an Index (p. 41) – This section provides information about using Amazon Kendra to create a search index and to add data sources to sync your documents.
4. Adding Documents Directly to an Index (p. 42) – This section provides information about adding documents directly to an Amazon Kendra index.
5. Searching Indexes (p. 61) – This section provides information about using the Amazon Kendra search API to search an index.
6. Deploying Amazon Kendra (p. 84) – This section provides information about using a sample application to deploy Amazon Kendra to your website.

How it Works

Amazon Kendra is a service that provides an interface for indexing and searching documents that you provide. Amazon Kendra creates an index of your documents that can be updated over time. It has a search API that you can use from a variety of client applications such as websites or mobile applications. Amazon Kendra supports indexes built from a variety of document types such as plain text, HTML files, Microsoft Word and PowerPoint files, and PDF files.

The architecture of Amazon Kendra is as follows.

- An index provides a search API for client queries. The index is created from source documents.
- A source repository that contains the documents to index.
- A data source syncs the documents in your source repositories to a Amazon Kendra index. A data source can be automatically synchronized with the Amazon Kendra index so that new, updated, and deleted files in the source repository are updated in the index.
- A document addition API allows documents to be added directly to the index.

Amazon Kendra provides a console and API that you can use to manage indexes and data sources. You can create indexes as well as update and delete them. Deleting an index removes all data sources and all of your document information from Amazon Kendra.

This section provides information about the key components of Amazon Kendra. It provides links to the procedures that describe how to create an index and then use it for searches.

Topics

Index

An index is the Amazon Kendra component that provides search results for documents and frequently asked questions that it has indexed. Documents are indexed in three ways.

- The documents are added to an index from your store using a data source.
- The documents are added directly to the index using the Amazon Kendra API.
- FAQ questions and answers are added to the index from an Amazon S3 bucket.

A single index can contain documents indexed from a data source, documents added directly to the index, and FAQs. You can create indexes with the Amazon Kendra console, the AWS CLI, or an AWS SDK. For information about the types of documents that can be indexed, see Types of Documents (p. 5).

For information about using a data source with an index, see Data Source (p. 6).

Documents are added directly to an index by calling the BatchPutDocument (p. 92) operation. The documents are supplied as plain text, as a binary blob, or using a path to a document stored in an Amazon S3 bucket. For an example, see Adding Files from an Amazon S3 Bucket (p. 44).

Index Fields

An index contains fields that Amazon Kendra uses to search your documents. The fields make up the schema of the index. You create a mapping between the fields and the attributes of your documents. After you create the mapping, you can use the information in the field for searching, for display, and to create facets of the search result.

Amazon Kendra defines six reserved fields. You can map your document attributes to these fields. The reserved field names are:

- _category – The category of the document.
- _created_at – The date and time that the document was created.
- _file_type – The file type of the document (html, pdf, etc.).
- _last_updated_at – The date and time that the document was last updated.
- _version – The version of the document.
- _view_count – The number of times that the document has been viewed.

You can also create custom fields that you can use like the reserved fields for search, display, and to create facets. You create a custom field using the console or by using the UpdateIndex (p. 149) operation.

A custom field can be one of four types:

- Date
- Number
- String
- String list

After you create the custom field, you map the field to document attributes. For documents added to the index with BatchPutDocument (p. 92), you map the attributes with the API. For documents indexed from an Amazon S3 data source, you map the attributes using a metadata file that contains a JSON structure that describes the document attributes. For documents indexed with a database or SharePoint Online data source, you configure attribute mapping using the console or the data source configuration. For more information, see Document Attributes (p. 6).

Searching Indexes

After you create an index, you can use it for search queries. For more information, see Searching Indexes (p. 61).

Documents

Amazon Kendra can index multiple types of documents. You can also associate attributes with documents to provide information such as the source URI and the author of a document.

Topics

- Document Attributes (p. 6)

Types of Documents

An index can include the following types of documents.

- Structured text
  - Frequently asked questions and answers
- Unstructured text
  - HTML
  - Microsoft PowerPoint document
  - Microsoft Word document
  - plain text
  - PDF

You can add documents directly to an index by calling the BatchPutDocument (p. 92) operation. You can also add documents from a data source. For information about adding files to a data source, see Adding Documents from a Data Source (p. 45). For an example that shows adding Microsoft Word files directly to an index from an Amazon S3 bucket, see Adding Files from an Amazon S3 Bucket (p. 44).

A single index can contain multiple types of documents.

HTML

HTML format files. You add an HTML file to an index the same way that you add a plain text file.

Plain Text

You can add plain text files to an index using the `BatchPutDocument` operation or from a data source. For an example of adding a plain text document directly to an index, see Adding Documents with the API (p. 43).

Microsoft Word Document

Microsoft Word format files can be added to an index as binary data, from an Amazon S3 bucket, or from an Amazon Kendra data source.

Microsoft PowerPoint Document

Microsoft PowerPoint format files can be added to an index as binary data, from an Amazon S3 bucket, or from an Amazon Kendra data source.

Portable Document Format (PDF)

PDF format files can be added to an index either as binary data, from an Amazon S3 bucket, or from an Amazon Kendra data source.

Frequently Asked Questions and Answers

Frequently asked question and answer format documents are used to answer questions such as *How tall is the Space Needle?* You can specify multiple questions that return the same answer. You specify the questions and answers in a comma-separated values (CSV) file stored in an Amazon S3 bucket.

For an example, see Adding Questions and Answers (p. 44).

Document Attributes

A document has attributes associated with it. You can also add your own custom attributes. Custom attributes are attributes that you can specify for your own needs. For example, if your index searches tax documents, you might specify a custom attribute for the type of tax document (W-2, 1099, and so on).

Before you can use a document attribute in a query, it must be mapped to a database field. For more information, see Index Fields (p. 4).

You can use document attributes to filter responses and to make faceted search suggestions. For example, you can filter a response to only return a specific version of a document, or you can filter searches to only return 1099 tax documents that match the search term. For more information, see Filtering Queries (p. 63).

You can also use document attributes to manually tune the query response. For example, you can choose to increase the importance of the title field to increase the weight that Amazon Kendra assigns to the field when determining which documents to return in the response. For more information, see Manually Tuning an Index (p. 74).

Before you can add an attribute, you must create an index field to map the attribute to. You create index fields using the console or by using the UpdateIndex (p. 149) operation.

If you are adding a document directly to an index, you specify the attributes in the Document (p. 174) input parameter to the the section called "BatchPutDocument" (p. 92) operation. You specify the custom attribute values in a DocumentAttribute (p. 176) object array. If you are using a data source, the method that you use to add the document attributes depends on the data source. For more information, see Creating Custom Document Attributes (p. 51).

Data Source

A data source is a location, such as an Amazon S3 bucket, where the documents for indexing are kept. Data sources can be automatically synchronized with an Amazon Kendra index so that new, updated, or deleted documents in the source repositories are included in searches. The supported data sources are:

- Amazon S3 bucket
- Amazon RDS for MySQL, Amazon RDS for PostgreSQL
- Microsoft SharePoint Online

The supported document formats are plain text, Microsoft Word, Microsoft PowerPoint, HTML, and PDF. For more information, see Types of Documents (p. 5).

> **Note**
> A data source isn't required to create a working index. You can add documents directly to an index. For more information, see Adding Documents Directly to an Index (p. 42).

To index documents using a data source.

1. Create an index (p. 41).
2. Create a data source (p. 45).

For a walkthrough with the Amazon Kendra console or with the AWS CLI, see Getting Started (p. 10).

Queries

You query your Amazon Kendra index to search for and return the documents relevant to the query text. You can use natural language in your query text. The response from Amazon Kendra contains the documents from your index that are most relevant to the query.

Amazon Kendra uses all of the information that you provide about your documents to determine if a document is relevant to the query, not just the contents of the document. For example, if your index contains information about when documents were last updated, Amazon Kendra may assign a higher relevance to documents that were updated more recently.

A query can also contain instructions on how to filter the response so that only documents that pass the filter are returned. For example, if you created an index field called *department*, you can filter the response so that only documents with the department field set to *legal* are returned. For more information, see Filtering Queries (p. 63).

The results of a query are also influenced by tuning the relevance of individual fields in the index. Tuning changes the importance of a field on the results. For example, if you raise the importance of documents with the file type *pdf*, PDF files are more likely to be included in the response. For more information, see Manually Tuning an Index (p. 74).

For more information on using a query to search your index, see Searching Indexes (p. 61).

Setting Up Amazon Kendra

Before using Amazon Kendra, you must have an Amazon Web Services (AWS) account. After you have an AWS account, you can access Amazon Kendra through the Amazon Kendra console, the AWS Command Line Interface (AWS CLI), or the AWS SDKs.

This guide includes examples for AWS CLI and Python.

Topics

Sign Up for AWS

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

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To sign up for AWS

1. Open https://aws.amazon.com, and then choose **Create an AWS Account**.
2. Follow the on-screen instructions to complete the account creation. Note your 12-digit AWS account number. Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.
3. Create an AWS Identity and Access Management (IAM) admin user. See Creating Your First IAM User and Group in the *AWS Identity and Access Management User Guide* for instructions.

Regions and Endpoints

An endpoint is a URL that is the entry point for a web service. Each endpoint is associated with a specific AWS region. If you use a combination of the Amazon Kendra console, the AWS CLI, and the Amazon Kendra SDKs, pay attention to their default regions as all Amazon Kendra components of a given campaign (index, query, etc.) must be created in the same region. For the regions and endpoints supported by Amazon Kendra, see Regions and Endpoints.

Setting Up the AWS CLI

The AWS Command Line Interface (AWS CLI) is a unified developer tool for managing AWS services, including Amazon Kendra. We recommend that you install it.

1. To install the AWS CLI, follow the instructions in Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

2. To configure the AWS CLI and set up a profile to call the AWS CLI, follow the instructions in Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.
3. To confirm that the AWS CLI profile is configured properly, run the following command:

```
aws configure --profile default
```

   If your profile has been configured correctly, you will see output similar to the following:

```
AWS Access Key ID [****************52FQ]:
AWS Secret Access Key [****************xgyZ]:
Default region name [us-west-2]:
Default output format [json]:
```

4. To verify that the AWS CLI is configured for use with Amazon Kendra, run the following commands:

```
aws kendra help
```

   If the AWS CLI is configured correctly, you will see a list of the supported AWS CLI commands for Amazon Kendra, Amazon Kendra runtime, and Amazon Kendra events.

Setting Up the AWS SDKs

Download and install the AWS SDKs that you want to use. This guide provides examples for Python. For information about other AWS SDKs, see Tools for Amazon Web Services.

Getting Started

This section shows how to create a data source and add documents, create an index, get search results, and get code samples to help you integrate Kendra into your application. Instructions are provided for the AWS console, the AWS CLI and a Python program using the AWS SDK for Python (Boto 3).

Topics

Prerequisites

The following steps are prerequisites for the getting started exercises. The steps show you how to set up your account, create an IAM role that gives Amazon Kendra to make calls on your behalf, and upload documents for indexing into an Amazon S3 bucket.

1. Create an AWS account and an AWS Identity and Access Management user, as specified in Sign Up for AWS (p. 8).
2. Create an S3 bucket in the same region that you are using Amazon Kendra. For instructions, see Creating and Configuring an S3 Bucket in the *Amazon Simple Storage Service Console User Guide*.
3. Upload your documents to your S3 bucket.

   For instructions, see Uploading, Downloading, and Managing Objects in the *Amazon Simple Storage Service Console User Guide*.

If you are using the console to get started, do Getting Started with the Console (p. 13). If you are using the AWS CLI or the SDK, you need to create IAM roles and polices for Kendra to use to access resources.

To create IAM roles and policies for Kendra

1. Create an IAM policy and role that enables Kendra to access your Amazon CloudWatch Logs.

   a. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
   b. From the left menu, choose **Policies** and then choose **Create policy**.
   c. Choose **JSON** and then replace the default policy with the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "AWS/Kendra"
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:region:account ID:log-group:/aws/kendra/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account ID:log-group:/aws/kendra/*:log-
stream:*"
            ]
        }
    ]
}
```

d. Choose **Review policy**.

e. Name the policy "KendraPolicyForGettingStartedIndex" and then choose **Create policy**.

f. From the left menu, choose **Roles** and then choose **Create role**.

g. Choose **Another AWS account** and then type your account ID in **Account ID**. Choose **Next: Permissions**.

h. Choose the policy that you created above and then choose **Next: Tags**

i. Don't add any tags. Choose **Next: Review**.

j. Name the role "KendraRoleForGettingStartedIndex" and then choose **Create role**.

k. Find the role that you just created. Choose the role name to open the summary. Choose **Trust relationships** and then choose **Edit trust relationship**.

l. Replace the existing trust relationship with the following:

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

m. Choose **Update trust policy**.

2. Create an IAM policy and role that enables Kendra to access and index your Amazon S3 bucket.

a. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

b. From the left menu, choose **Policies** and then choose **Create policy**.

c. Choose **JSON** and then replace the default policy with the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket name/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket name"
            ],
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kendra:BatchPutDocument",
                "kendra:BatchDeleteDocument"
            ],
            "Resource": "arn:aws:kendra:region:account ID:index/*"
        }
    ]
}
```

d. Choose **Review policy**.

e. Name the policy "KendraPolicyForGettingStartedDataSource" and then choose **Create policy**.

f. From the left menu, choose **Roles** and then choose **Create role**.

g. Choose **Another AWS account** and then type your account ID in **Account ID**. Choose **Next: Permissions**.

h. Choose the policy that you created above and then choose **Next: Tags**

i. Don't add any tags. Choose **Next: Review**.

j. Name the role "KendraRoleForGettingStartedDataSource" and then choose **Create role**.

k. Find the role that you just created. Choose the role name to open the summary. Choose **Trust relationships** and then choose **Edit trust relationship**.

l. Replace the existing trust relationship with the following:

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

```
    }
  ]
}
```

m. Choose **Update trust policy**.

3. Do Getting Started with the AWS CLI (p. 14) or Getting Started with the AWS SDK for Python (Boto 3) (p. 15).

Getting Started with the Console

The following procedures show how to create and test an Amazon Kendra index by using the AWS console. In the procedures you create an index and a data source for an S3 bucket. Finally, you test your index by making a search request.

Step 1: To create an index (Console)

1. Choose **Create index** to start creating a new index.
2. In **Specify index details**, give your index a name and a description.
3. In **IAM role** choose **Create a new role** and then give the role a name. The IAM role will have the prefix "AmazonKendra-".
4. Choose **Create**.
5. Wait for your index to be created. Kendra provisions the hardware for your index. This operation can take some time.

Step 2: To add a data source to an index (Console)

1. From **Getting started** choose **Add data sources**.
2. In **Select data source type** choose **Amazon S3**.
3. In **Name data source** give your data source a name and a description and then choose **Next**.
4. In **Configure S3 connector**, enter the name of the S3 bucket that contains the files that you want to index. Leave the metadata and access control fields blank. In the **IAM role** field, choose **Create a new role** and then give the role a name. The IAM role will have the prefix "AmazonKendra-".
5. In **Set sync run schedule** choose how often you want your the index to synchronize the data source.
6. Choose **Next**. Review the configuration of your data source. If everything looks correct, choose **Create**.
7. Wait for your data source to be created.

Step 3: Synchronize your index with your data source (Console)

1. After your data source is created, choose **Sync now** to start synchronizing your data base with your index.
2. Wait for the synchronization to complete. The time that it takes depends on the number of documents that you are indexing.
3. Since this is the first time that you have synchronized this data source, the number of new or updated documents in the sync run history should be the same as the number of documents in your repository. If you have trouble indexing your data source, see Troubleshooting (p. 58).

Step 4: To search an index (Console)

1. In the navigation pane, choose **Search console**

2. Enter a search term that's appropriate for your index. The **top results** and **top document** results are shown.

Step 5: To deploy a search page to your Web site

1. From the left menu, choose **Search console**.
2. Enter a query in the search box hit Enter.
3. From the right menu, choose the code symbol (**</>**).
4. Hover over one of the three sections on the page to see a description of the section.
5. Choose a link to read more about deploying a search box on your Web site.

Getting Started with the AWS CLI

The following procedure shows how to create an Amazon Kendra index using the AWS CLI. The procedure creates a data source, index, and runs a query on the index.

To create an Amazon Kendra index (CLI)

1. Do the Prerequisites (p. 10).
2. Enter the following command to create an index

```
aws kendra create-index \
 --name cli-getting-started-index \
 --description "Index for CLI getting started guide." \
 --role-arn arn:aws:iam::account id:role/KendraRoleForGettingStartedIndex
```

3. Wait for Amazon Kendra to create the index. Check the progress using the following command. When the status field is `ACTIVE`, go on to the next step.

```
aws kendra describe-index \
 --id index id
```

4. At the command prompt, enter the following command to create a data source.

```
aws kendra create-data-source \
 --index-id index id \
 --name data source name \
 --role-arn arn:aws:iam::account id:role/KendraRoleForGettingStartedDataSource
 --type S3 \
 --configuration '{"S3Configuration":{"BucketName":"S3 bucket name"}}'
```

5. It will take Amazon Kendra a while to create the data source. Enter the following command to check the progress. When the status is `ACTIVE`, go on to the next step.

```
aws kendra describe-data-source \
 --id data source ID \
 --index-id index ID
```

6. Enter the following command to synchronize the data source.

```
aws kendra start-data-source-sync-job \
 --id data source ID \
 --index-id index ID
```

7. Kendra will index your data source. The amount of time that it takes depends on the number of documents. You can check the status of the sync job using the following command. When the status is `ACTIVE`, go on to the next step.

```
aws kendra describe-data-source \
 --id data source ID \
 --index-id index ID
```

8. Enter the following command to make a query.

```
aws kendra query \
 --index-id index ID \
 --query-text "search term"
```

The results of the search are displayed in JSON format.

Getting Started with the AWS SDK for Python (Boto 3)

The following program is an example of using Amazon Kendra in a Python program. The program performs the following actions:

1. Creates a new index using the CreateIndex (p. 102) operation.
2. Waits for index creation to complete. It uses the DescribeIndex (p. 118) operation to monitor the status of the index.
3. Once the index is active, it creates a data source using the CreateDataSource (p. 95) operation.
4. Waits for data source creation to complete. It uses the DescribeDataSource (p. 109) operation to monitor the status of the data source.
5. When the data source is active, it synchronizes the index with the contents of the data source using the StartDataSourceSyncJob (p. 139) operation.

```
import boto3
from botocore.exceptions import ClientError
import pprint
import time

kendra = boto3.client("kendra")

print("Create an index")

description = "Getting started index"
index_name = "python-getting-started-index"
index_role_arn = "arn:aws:iam::${accountId}:role/KendraRoleForGettingStartedIndex"

try:
    index_response = kendra.create_index(
        Description = description,
        Name = index_name,
        RoleArn = index_role_arn
    )

    pprint.pprint(index_response)

    index_id = index_response["Id"]
```

```
    print("Wait for Kendra to create the index.")

    while True:
        # Get index description
        index_description = kendra.describe_index(
            Id = index_id
        )
        # When status is not CREATING quit.
        status = index_description["Status"]
        print("    Creating index. Status: "+status)
        time.sleep(60)
        if status != "CREATING":
            break

    print("Create an S3 data source")

    data_source_name = "python-getting-started-data-source"
    data_source_description = "Getting started data source."
    s3_bucket_name = "${bucketName}"
    data_source_type = "S3"
    data_source_role_arn = "arn:aws:iam::${accountId}:role/
KendraRoleForGettingStartedDataSource"

    configuration = {"S3Configuration":
        {
            "BucketName": s3_bucket_name
        }
    }

    data_source_response=kendra.create_data_source(
        Configuration = configuration,
        Name = data_source_name,
        Description = description,
        RoleArn = data_source_role_arn,
        Type = data_source_type,

        IndexId = index_id
    )

    pprint.pprint(data_source_response)

    data_source_id = data_source_response["Id"]

    print("Wait for Kendra to create the data source.")

    while True:
        data_source_description = kendra.describe_data_source(
            Id = data_source_id,
            IndexId = index_id
        )
        # When status is not CREATING quit.
        status = data_source_description["Status"]
        print("    Creating data source. Status: "+status)
        time.sleep(60)
        if status != "CREATING":
            break

    print("Synchronize the data source.")

    sync_response = kendra.start_data_source_sync_job(
        Id = data_source_id,
        IndexId = index_id
    )

    pprint.pprint(sync_response)
```

```
    print("Wait for the data source to sync with the index.")

    while True:
        data_source_description = kendra.describe_data_source(
            Id = data_source_id,
            IndexId = index_id
        )
        # When status is not SYNCING quit.
        status = data_source_description["Status"]
        print("    Syncing data source. Status: "+status)
        time.sleep(60)
        if status != "SYNCING":
            break

except  ClientError as e:
        print("%s" % e)

print("Program ends.")
```

Security in Amazon Kendra

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 Kendra, 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 Kendra. The following topics show you how to configure Amazon Kendra 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 Kendra resources.

Topics

Data Protection in Amazon Kendra

Amazon Kendra conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

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), so that 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.

- 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.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon Kendra or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon Kendra or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

Encryption at Rest

Amazon Kendra encrypts your data at rest with your choice of an encryption key. You can choose one of the following:

- An AWS owned customer master key (CMK). If you don't specify an encryption key your data is encrypted with this key by default.
- An AWS managed CMK in your account. This key is created, managed, and used on your behalf by Amazon Kendra. The key name is `aws/kendra`.
- A customer managed CMK. You can provide the ARN of an encryption key that you created in your account. When you use a customer managed CMK, you must give the key a key policy that enables Amazon Kendra to use the key. Select a symmetric customer managed CMK, Amazon Kendra does not support asymmetric CMKs. For more information, see Key Management (p. 19).

Encryption in Transit

Amazon Kendra uses the HTTPS protocol to communicate with your client application. It uses HTTPS and AWS signatures to communicate with other services on your application's behalf.

Key Management

Amazon Kendra encrypts the contents of your index using one of three types of keys. You can choose one of the following:

- An AWS owned customer master key (CMK). This is the default.
- An AWS managed CMK. This key is created in your account and is managed and used on your behalf by Amazon Kendra.
- A customer managed CMK. You can create the key when you are creating an Amazon Kendra index or data source, or you can create the key using the AWS KMS console. Select a symmetric customer managed CMK, Amazon Kendra does not support asymmetric CMKs. For more information, see Using Symmetric and Asymmetric Keys in the *AWS Key Management Service Developer Guide*.

When you create a key using the AWS KMS console, you must give the key the following policy that enables Amazon Kendra to use the key. If you create a key with the Amazon Kendra console, the policy is applied to the key for you. For more information, see Using Key Policies in AWS KMS in the *AWS Key Management Service Developer Guide*.

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "kendra.amazonaws.com"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
```

```
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey",
        "kms:CreateGrant",
        "kms:RetireGrant"
    ],
    "Resource": "*"
}
```

Identity and Access Management for Amazon Kendra

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Kendra resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in Amazon Kendra.

Service user – If you use the Amazon Kendra service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon Kendra features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon Kendra, see Troubleshooting Amazon Kendra Identity and Access (p. 28).

Service administrator – If you're in charge of Amazon Kendra resources at your company, you probably have full access to Amazon Kendra. It's your job to determine which Amazon Kendra features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon Kendra, see How Amazon Kendra Works with IAM (p. 24).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon Kendra. To view example Amazon Kendra identity-based policies that you can use in IAM, see Amazon Kendra Identity-Based Policy Examples (p. 26).

Authenticating With Identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see The IAM Console and Sign-in Page in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *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*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

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 Users and Groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing Access Keys for IAM Users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to Create an IAM User (Instead of a Role) in the *IAM User Guide*.

IAM Roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM Roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.

- **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*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. 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*.

To learn whether to use IAM roles, see When to Create an IAM Role (Instead of a User) in the *IAM User Guide*.

Managing Access Using Policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON Policies in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which

resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM Policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing Between Managed Policies and Inline Policies in the *IAM User Guide*.

Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket. Service administrators can use these policies to define what actions a specified principal (account member, user, or role) can perform on that resource and under what conditions. Resource-based policies are inline policies. There are no managed resource-based policies.

Access Control Lists (ACLs)

Access control lists (ACLs) are a type of policy that controls which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format. Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*.

Other Policy Types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions Boundaries for IAM Entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs Work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session Policies in the *IAM User Guide*.

Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy Evaluation Logic in the *IAM User Guide*.

How Amazon Kendra Works with IAM

Before you use IAM to manage access to Amazon Kendra, you should understand what IAM features are available to use with Amazon Kendra. To get a high-level view of how Amazon Kendra and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

Topics

Amazon Kendra Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Amazon Kendra supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon Kendra use the following prefix before the action: `kendra:`. For example, to grant someone permission to list Amazon Kendra indexes with the ListIndices (p. 131) API operation, you include the `kendra:ListIndices` action in their policy. Policy statements must include either an `Action` or `NotAction` element. Amazon Kendra defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "kendra:action1",
      "kendra:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "kendra:Describe*"
```

To see a list of Amazon Kendra actions, see Actions Defined by Amazon Kendra in the *IAM User Guide*.

Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

The Amazon Kendra index resource has the following ARN:

```
arn:${Partition}:kendra:${Region}:${Account}:index/${IndexId}
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, to specify an index in your statement, use the GUID of the index in the following ARN:

```
"Resource": "arn:aws:kendra:${Region}:${Account}:index/${GUID}"
```

To specify all indexes that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:${Region}:${Account}:index/*"
```

Some Amazon Kendra actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

```
"Resource": "*"
```

To see a list of Amazon Kendra resource types and their ARNs, see Resources Defined by Amazon Kendra in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon Kendra.

Condition Keys

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can build conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM Policy Elements: Variables and Tags in the *IAM User Guide*.

Amazon Kendra does not provide any service-specific condition keys, but it does support using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

Examples

To view examples of Amazon Kendra identity-based policies, see Amazon Kendra Identity-Based Policy Examples (p. 26).

Amazon Kendra Resource-Based Policies

Amazon Kendra does not support resource-based policies.

Access Control Lists (ACLs)

Amazon Kendra does not support access control lists (ACLs).

Authorization Based on Amazon Kendra Tags

Amazon Kendra does not support tagging resources or controlling access based on tags.

Amazon Kendra IAM Roles

An IAM role is an entity within your AWS account that has specific permissions.

Using Temporary Credentials with Amazon Kendra

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

Amazon Kendra supports using temporary credentials.

Service Roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

Amazon Kendra supports service roles.

Choosing an IAM Role in Amazon Kendra

When you create an index, call the `BatchPutDocument` operation, create a data source or create an FAQ, you must provide an access role Amazon Resource Name (ARN) that Amazon Kendra uses to access the required resources on your behalf. If you have previously created a role, then the Amazon Kendra console provides you with a list of roles to choose from. It's important to choose a role that allows access to the resources that you require. For more information, see IAM Access Roles for Amazon Kendra (p. 32).

Amazon Kendra Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify Amazon Kendra resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

Topics

Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon Kendra resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using Amazon Kendra quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the *IAM User Guide*.

- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

AWS Managed (Predefined) Polices for Amazon Kendra

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These policies are called AWS managed policies. AWS managed policies make it easier for you to assign permissions to users, groups, and roles than if you had to write the policies yourself. For more information, see Adding Permissions to a User in the *IAM User Guide*.

The following AWS managed policies, which you can attach to groups and roles in your account, are specific to Amazon Kendra:

- **AmazonKendraReadOnly** — Grants read-only access to Amazon Kendra resources.
- **AmazonKendraFullAccess** — Grants full access to create, read, update, delete, and run all Amazon Kendra resources.

For the console, your role must also have `iam:CreateRole`, `iam:CreatePolicy`, `iam:AttachRolePolicy`, and `s3:ListBucket` permissions.

> **Note**
> You can review these permissions by signing in to the IAM console and searching for specific policies.

You can also create your own custom policies to allow permissions for Amazon Kendra API actions. You can attach these custom policies to the IAM roles or groups that require those permissions. For examples of IAM policies for Amazon Kendra, see Amazon Kendra Identity-Based Policy Examples (p. 26).

Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "ViewOwnUserInfo",
               "Effect": "Allow",
               "Action": [
                   "iam:GetUserPolicy",
                   "iam:ListGroupsForUser",
                   "iam:ListAttachedUserPolicies",
                   "iam:ListUserPolicies",
                   "iam:GetUser"
               ],
```

```
            "Resource": [
                "arn:aws:iam::*:user/${aws:username}"
            ]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Accessing One Amazon Kendra Index

In this example, you want to grant an IAM user in your AWS account access to query an index.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QueryIndex",
            "Effect": "Allow",
            "Action": [
                "kendra:Query"
            ],
            "Resource": "arn:aws:kendra:${Region}:${Account}:index/${Index ID}"
        }
    ]
}
```

Troubleshooting Amazon Kendra Identity and Access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Kendra and IAM.

Topics

I Am Not Authorized to Perform an Action in Amazon Kendra

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about an index but does not have `kendra:DescribeIndex` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 kendra:DescribeIndex on resource: index ARN
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `index` resource using the `kendra:DescribeIndex` action.

I Am Not Authorized to Perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to Amazon Kendra.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Kendra. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I Want to View My Access Keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing Access Keys in the *IAM User Guide*.

I'm an Administrator and Want to Allow Others to Access Amazon Kendra

To allow others to access Amazon Kendra, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Amazon Kendra.

To get started right away, see Creating Your First IAM Delegated User and Group in the *IAM User Guide*.

I Want to Allow People Outside of My AWS Account to Access My Amazon Kendra Resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Kendra supports these features, see How Amazon Kendra Works with IAM (p. 24).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing Access to an IAM User in Another AWS Account That You Own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing Access to AWS Accounts Owned by Third Parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing Access to Externally Authenticated Users (Identity Federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.

Logging and Monitoring in Amazon Kendra

Monitoring is an important part of maintaining the reliability, availability, and performance of your Amazon Kendra applications. To monitor Amazon Kendra API calls, you can use AWS CloudTrail. To monitor the status of your jobs, use Amazon CloudWatch Logs.

- **Amazon CloudWatch Alarms** — Using CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions when a metric is in a particular state. Rather the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring Amazon Kendra with Amazon CloudWatch (p. 77).
- **AWS CloudTrail Logs** — CloudTrail provides a record of actions taken by a user, role, or an AWS service in Amazon Kendra. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Kendra, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging Amazon Kendra API Calls with AWS CloudTrail Logs (p. 76).

Compliance Validation for Amazon Kendra

Third-party auditors assess the security and compliance of Amazon Kendra as part of multiple AWS compliance programs. Amazon Kendra is not in scope of any AWS compliance programs.

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.

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 Kedra 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.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config 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.

Resilience in Amazon Kendra

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 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.

In addition to the AWS global infrastructure, Amazon Kendra offers several features to help support your data resiliency and backup needs.

Infrastructure Security in Amazon Kendra

As a managed service, Amazon Kendra is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Amazon Kendra through the network. 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 IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

IAM Access Roles for Amazon Kendra

When you create an index, data source, or an FAQ, Amazon Kendra needs access to the AWS resources required to create the Amazon Kendra resource. You must create a AWS Identity and Access Management (IAM) policy before you create the Amazon Kendra resource. When you call the operation, you provide the Amazon Resource Name (ARN) of the role with the policy attached. For example, if you are calling the BatchPutDocument (p. 92) operation to add documents from an Amazon S3 bucket, you provide Amazon Kendra with a role with a policy that has access to the bucket.

The Amazon Kendra console enables you to create a new IAM role or to choose an IAM existing role to use. The console displays roles that have the string "kendra" or "Kendra" in the role name.

The following topics provide details for the required policies. If you create IAM roles using the Amazon Kendra console these policies are created for you.

Topics

IAM Roles for Indexes

When you create an index, you must provide an IAM role with permission to write to an Amazon CloudWatch Logs. You must also provide a trust policy that allows Amazon Kendra to assume the role. The following are the policies that must be provided.

A role policy to enable Amazon Kendra to access a CloudWatch log.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "Kendra"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "logs:DescribeLogGroups",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:region:account ID:log-group:/aws/kendra/*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:region:account ID:log-group:/aws/kendra/*:log-
stream:*"
        }
    ]
}
```

A trust policy to enable Amazon Kendra to assume a role.

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

IAM Roles for the BatchPutDocument Operation

When you use the BatchPutDocument (p. 92) operation to index documents in an Amazon S3 bucket, you must provide Amazon Kendra with an IAM role with access to the bucket. You must also provide a trust policy that enables Amazon Kendra to assume the role. If the documents in the bucket are encrypted, you must provide permission to use the AWS KMS customer master key (CMK) to decrypt the documents.

A required role policy to enable Amazon Kendra to access an Amazon S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket name/*"
            ]
        }
    ]
}
```

A required trust policy to enable Amazon Kendra to assume a role.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowKendraToAssumeAttachedRole",
        "Effect": "Allow",
        "Principal": {
```

```
            "Service": "kendra.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
}
```

An optional role policy to enable Amazon Kendra to use an AWS KMS customer master key (CMK) to decrypt documents in an Amazon S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region:account ID:key/key ID"
            ]
        }
    ]
}
```

IAM Roles for Data Sources

When you use the CreateDataSource (p. 95) operation, you must give Amazon Kendra an IAM role that has permission to access the database resources. The specific permissions required depend on the data source.

Topics

- IAM Roles for Amazon S3 Data Sources (p. 34)
- IAM Roles for Database Data Sources (p. 35)
- IAM Roles for SharePoint Online Data Sources (p. 37)

IAM Roles for Amazon S3 Data Sources

When you use an Amazon S3 bucket as a data source, you supply a role that has permission to access the bucket, and to use the `BatchPutDocument` and `BatchDeleteDocument` operations. If the documents in the Amazon S3 bucket are encrypted, you must provide permission to use the AWS KMS customer master key (CMK) to decrypt the documents.

A required role policy to enable Amazon Kendra to use an Amazon S3 bucket as a data source.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket name/*"
            ],
            "Effect": "Allow"
```

```
        },
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket name"
            ],
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kendra:BatchPutDocument",
                "kendra:BatchDeleteDocument"
            ],
            "Resource": [
                "arn:aws:kendra:region:account ID:index/index ID"
            ]
        }
    ]
}
```

An optional role policy to enable Amazon Kendra to use an AWS KMS customer master key (CMK) to decrypt documents in an Amazon S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region:account ID:key/key ID"
            ]
        }
    ]
}
```

IAM Roles for Database Data Sources

When you use a Microsoft database as a data source, you provide Amazon Kendra with a role that has the permissions necessary for connecting to the site. These include:

- Permission to access the AWS Secrets Manager secret that contains the user name and password for the database site.
- Permission to use the AWS KMS customer master key (CMK) to decrypt the user name and password secret stored by Secrets Manager.
- Permission to use the `BatchPutDocument` and `BatchDeleteDocument` operations to update the index.
- Permission to access the Amazon S3 bucket that contains the SSL certificate used to communicate with the database site.

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

```
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:region:account ID:secret:secret ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region:account ID:key/key ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kendra:BatchPutDocument",
                "kendra:BatchDeleteDocument"
            ],
            "Resource": [
                "arn:aws:kendra:region:account ID:index/index ID"
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "kendra.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket name/*"
            ]
        }
    ]
}
```

There are two optional policies that you might use with a database data source.

If you have encrypted the Amazon S3 bucket that contains the SSL certificate used to communicate with the database, provide a policy to give Amazon Kendra access to the key.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region:account ID:key/key ID"
            ]
        }
    ]
}
```

If you are using a VPC, provide a policy that gives Amazon Kendra access to the required resources.

```
{
    "Version": "2012-10-17",
    "Statement": [
  {
    "Effect": "Allow",
    "Action":[
      "ec2:CreateNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DeleteNetworkInterface"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterfacePermission"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "ec2:AuthorizedService": "kendra.amazonaws.com"
      },
      "ArnEquals": {
        "ec2:Subnet": [
          "arn:aws:ec2:region:account ID:subnet/subnet IDs"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeSubnets"
    ],
    "Resource": "*"
  }    ]
}
```

IAM Roles for SharePoint Online Data Sources

For a Microsoft SharePoint Online data source, you provide a role with the following policies.

- Permission to access the AWS Secrets Manager secret that contains the user name and password for the SharePoint site.
- Permission to use the AWS KMS customer master key (CMK) to decrypt the user name and password secret stored by Secrets Manager.
- Permission to use the `BatchPutDocument` and `BatchDeleteDocument` operations to update the index.
- Permission to access the Amazon S3 bucket that contains the SSL certificate used to communicate with the SharePoint site.

You must also attach a trust policy that enables Amazon Kendra to assume the role.

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

```
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:region:account ID:secret:secret ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region:account ID:key/key ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kendra:BatchPutDocument",
                "kendra:BatchDeleteDocument"
            ],
            "Resource": [
                "arn:aws:kendra:region:account ID:index/index ID"
            ],
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "kendra.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket name/*"
            ]
        }
    ]
}
```

You must apply the following trust policy to the role.

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

If you have encrypted the Amazon S3 bucket that contains the SSL certificate used to communicate with the Sharepoint site, provide a policy to give Amazon Kendra access to the key.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region:account ID:key/key ID"
            ]
        }
    ]
}
```

IAM Roles for Frequently Asked Questions

When you use the CreateFaq (p. 99) operation to load questions and answers into an index, you must provide Amazon Kendra with an IAM role with access to the Amazon S3 bucket that contains the source files. If the source files are encrypted, you must provide permission to use the AWS KMS customer master key (CMK) to decrypt the files.

A required role policy to enable Amazon Kendra to access an Amazon S3 bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket name/*"
            ]
        }
    ]
}
```

A required trust policy to enable Amazon Kendra to assume a role.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowKendraToAssumeAttachedRole",
        "Effect": "Allow",
        "Principal": {
            "Service": "kendra.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
}
```

An optional role policy to enable Amazon Kendra to use an AWS KMS customer master key (CMK) to decrypt files in an Amazon S3 bucket.

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

```
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:region:account ID:key/key ID"
            ],
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "kendra.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Creating an Index

You can create an index using the console, the AWS Command Line Interface, or by calling the CreateIndex (p. 102) API operation. The following procedures show how to create an index. Once you have created your index, you can add documents directly to your index or you can add them from a data source.

To create an index, you need to provide the Amazon Resource Name (ARN) of an IAM role that has permissions to any Amazon S3 bucket that you use and to perform actions on your behalf.

To create an index (Console)

1. Sign into the AWS Management Console and open the Amazon Kendra console at https://console.aws.amazon.com/kendra/.
2. Choose **Create index** to start creating a new index.
3. In **Specify index details**, give your index a name and a description.
4. In **IAM role** provide an IAM role. You can either choose from roles in your account that contain the word "kendra" or you can type the name of another role. For more information about the permissions that the role requires, see IAM Roles for Indexes (p. 32).
5. Choose **Create**.
6. Creating an index can take some time. Check the list of indexes to watch the progress of creating your index. When the status of the index is `ACTIVE` your index is ready to use.

To create an index (CLI)

1. Use the following command to create an index. The role-arn should be the Amazon Resource Name (ARN) of a role that can execute Amazon Kendra actions. For more information, see IAM Access Roles for Amazon Kendra (p. 32).

```
aws kendra create-index \
 --index-name index name \
 --description "index description" \
 --role-arn arn:aws:iam::account ID:role/role name
```

2. Creating an index can take some time. Use the following command to check the state of your index. When the status of the index is `ACTIVE` your index is ready to use.

```
aws kendra describe-index \
 --index-id index ID
```

To create an index (SDK)

1. You need to provide values for the following variables:
   - `description` – A description of the index that you are creating.
   - `index_name` – The name of the index that you are creating.
   - `role_arn` – The Amazon Resource Name (ARN) of a role that can execute Amazon Kendra actions. For more information, see IAM Access Roles for Amazon Kendra (p. 32).
2. Use the following Python code:

```
import boto3
```

```
from botocore.exceptions import ClientError
import pprint
import time

kendra = boto3.client("kendra")

print("Create an index")

description = "index description"
index_name = "index-name"
role_arn = "arn:aws:iam::${account id}:role/${role name}"

try:
    index_response = kendra.create_index(
        Description = description,
        Name = index_name,
        RoleArn = role_arn
    )

    pprint.pprint(index_response)

    index_id = index_response["IndexId"]

    print("Wait for Kendra to create the index.")

    while True:
        # Get index description
        index_description = kendra.describe_index(
            Id = index_id
        )
        # If status is not CREATING quit
        status = index_description["Status"]
        print("    Creating index. Status: "+status)
        if status != "CREATING":
            break
        time.sleep(60)

except  ClientError as e:
        print("%s" % e)

print("Program ends.")
```

Once you have created your index, you add documents to the index. You can either add them directly to the index or you can create a data source that automatically updates your index on a regular schedule.

Topics

Adding Documents Directly to an Index

You can add documents directly to an index by calling the BatchPutDocument (p. 92) operation. You can add the following types of documents.

- Plain text

- Question and Answer (FAQ)
- HTML
- PDF
- Microsoft PowerPoint
- Microsoft Word

Documents can be added from an Amazon S3 bucket or supplied as binary data. The following examples show how to add documents directly to an index.

Topics

Adding Documents with the API

The following example adds text to an index by calling the BatchPutDocument (p. 92) operation.

You can use the `BatchPutDocument` operation to add documents in the following formats:

- DOC
- HTML
- PDF
- Plain text
- PPT

Files added to the index must be in a UTF-8 encoded byte stream. The following example adds UTF-8 encoded text to the index.

```
import boto3

kendra = boto3.client('kendra')

index_id = '${indexID}'

title = 'Information about Amazon.com'
text = 'Amazon.com is an online retailer.'

document = {
    "Id": "1",
    "Blob": text,
    "ContentType": "PLAIN_TEXT",
    "Title": title
}

documents = [
    document
]

result = kendra.batch_put_document(
    IndexId = index_id,
    Documents = documents
)

print(result)
```

Adding Files from an Amazon S3 Bucket

You can add documents directly to your index from an Amazon S3 bucket. You can add up to 10 documents in the same call. When you use an S3 bucket, you must provide an IAM role with permission to access the bucket containing your documents. You specify the role in the `RoleArn` parameter.

Using BatchPutDocument (p. 92) operation to add documents from an Amazon S3 bucket is a one-time operation. To keep an index synchronized with the contents of a bucket create an S3 data source. For more information, see Using an Amazon S3 Data Source (p. 46).

The following example adds two Microsoft Word documents to the index using the `BatchPutDocument` operation.

```
import boto3

kendra = boto3.client('kendra')

index_id = '${indexId}'
role_arn = 'arn:aws:iam::${acccountID}:policy/${roleName}'

polly_s3_file_data = {
    'Bucket': '${bucketName}',
    'Key': 'What is Amazon Polly.docx'
}

polly_document = {
    'S3Path': polly_s3_file_data,
    'Title': 'What is Amazon Polly',
    'Id': 'polly_doc_1'
}

rekognition_s3_file_data = {
    'Bucket': '${bucketName}',
    'Key': 'What is Amazon Rekognition.docx'
}

rekognition_document = {
    'S3Path': rekognition_s3_file_data,
    'Title': 'What is Amazon Rekognition',
    'Id': 'rekognition_doc_1'
}

documents = [
    polly_document,
    rekognition_document
]

result = kendra.batch_put_document(
    Documents = documents,
    IndexId = index_id,
    RoleArn = role_arn
)

print(result)
```

Adding Questions and Answers

You add questions and answers (FAQs) to your index using the CreateFaq (p. 99) operation. The data for the FAQ comes from a comma-separated values (csv) document stored in an Amazon S3 bucket. The file contains a list of questions, answers, and optionally the URI of a document that contains more information. You can provide one or more questions for each answer.

The following is a csv file that provides answers to questions about the height of buildings in Seattle.

```
What is the height of the Space Needle?, 605 feet, https://www.spaceneedle.com/
How tall is the Space Needle?, 605 feet, https://www.spaceneedle.com/
What is the height of the Smith Tower?, 484 feet, https://www.smithtower.com
How tall is the Smith Tower, 484 feet, https://www.smithtower.com/
```

Once you store your FAQ input file in an Amazon S3 bucket, you use the console or the `CreateFaq` operation to put the questions and answers into your index. You must provide an IAM role that has access to the bucket containing your source files. You specify the role in the console or in the `RoleArn` parameter. The following is a Python program that adds an FAQ file to an index.

```
import boto3


kendra = boto3.client('kendra')

index_id = '${indexId}'
role_arn = 'arn:aws:iam::${accountId}:role/${roleName}'

faq_path = {
    'Bucket': '${bucketName}',
    'Key': 'SeattleBuildings.csv'
}

response = kendra.create_faq(
    S3Path =  faq_path,
    Name = 'SeattleBuildings',
    IndexId = index_id,
    RoleArn = role_arn
)

print(response)
```

Adding Documents from a Data Source

When you create a data source you give Amazon Kendra the location of documents that it should index. Unlike adding documents directly to an index, you can periodically scan the data source to update the index.

For example, say that you have a repository of tax instruction stored in an Amazon S3 bucket. Existing documents are changed and new documents are added to the repository from time to time. If you add the repository to Amazon Kendra as a data source, you can keep your index up to date by periodically updating your index.

You can update the index manually using console or the StartDataSourceSyncJob (p. 139) operation, or you can set up a schedule to update the index.

An index can have more than one data source. Each data source can have its own update schedule. For example, you might update the index of your working documents daily, or even hourly, while updating your archived documents manually whenever the archive changes.

Setting an Update Schedule

Configure your data source to periodically update with the console or by using the `Schedule` parameter when you create or update a data source. The content of the parameter is a string that holds either

a `cron`-format schedule string or an empty string to indicate that the index should be updated on demand. For the format of a cron expression, see Schedule Expressions for Rules in the *Amazon CloudWatch Events User Guide*. Amazon Kendra only supports cron expressions, it does not support rate expressions.

- Using an Amazon S3 Data Source (p. 46)
- Using a Microsoft SharePoint Data Source (p. 51)
- Using a Database Data Source (p. 49)

Using an Amazon S3 Data Source

Use an S3 data source when you document repository is an Amazon S3 bucket.

You must create an index before you create a data source. You provide the index identifier as a parameter to the CreateDataSource (p. 95) operation.

Amazon Kendra must have permission to access the Amazon S3 bucket that contains your documents. You provide the Amazon Resource Name (ARN) of a role that has access when you create the data source using the `RoleARN` parameter.

The following examples demonstrate creating an S3 data source. The examples assume that you have already created an index and an IAM role with permission to read the data from the index. For more information about the IAM role, see IAM Roles for Amazon S3 Data Sources (p. 34) For more information about creating an index, see Creating an Index (p. 41).

CLI

```
aws kendra create-data-source \
 --index-id index ID \
 --name example-data-source \
 --type S3 \
 --configuration '{"S3n-Configuration":{"BucketName":"bucket name"}}'
 --role-arn 'arn:aws:iam::account id:role:/role name
```

Python

The following snippet of Python code creates an S3 data source. For the complete example, see Getting Started with the AWS SDK for Python (Boto 3) (p. 15).

```
    print("Create an S3 data source")

    name = "getting-started-data-source"
    description = "Getting started data source."
    s3_bucket_name = "${bucketName}"
    type = "S3"
    role_arn = "arn:aws:iam::${accountID}:role/${roleName}"

    configuration = {"S3DataSourceConfiguration":
        {
            "BucketName": s3_bucket_name
        }
    }

    data_source_response=kendra.create_data_source(
        Configuration = configuration,
        Name = name,
        Description = description,
        RoleArn = role_arn,
```

```
        Type = type,

        IndexId = index_id
    )
```

It can take some time to create your data source. You can monitor the progress by using the DescribeDataSource (p. 109) operation. When the data source status is `ACTIVE` the data source is ready to use.

The following examples demonstrate getting the status of a data source.

CLI

```
aws kendra describe-data-source \
 --index-id index ID \
 --id data source ID
```

Python

The following snippet of Python code gets information about an S3 data source. For the complete example, see Getting Started with the AWS SDK for Python (Boto 3) (p. 15).

```
    print("    Wait for Kendra to create the data source.")

    while True:
        data_source_description = kendra.describe_data_source(
            Id = "data source ID",
            IndexId = "index ID"
        )
        status = data_source_description["Status"]
        print("Creating data source. Status: "+status)
        time.sleep(60)
        if status != "CREATING":
            break
```

This data source doesn't have a schedule, so it will not run automatically. To index the data source you call the StartDataSourceSyncJob (p. 139) operation to synchronize the index with the data source.

The following examples demonstrate synchronizing a data source.

CLI

```
aws kendra start-data-source-sync-job \
 --index-id index ID \
 --id data source ID
```

Python

The following snippet of Python code synchronizes an S3 data source. For the complete example, see Getting Started with the AWS SDK for Python (Boto 3) (p. 15).

```
    print("Synchronize the data source.")

    sync_response = kendra.start_data_source_sync_job(
        Id = "data source IC",
        IndexId = "index ID"
    )
```

S3 Document Metadata

You can add metadata, additional information about a document, to documents in an Amazon S3 bucket using a metadata file. Each metadata file is associated with an indexed document.

Your metadata files must be stored in the same bucket as your indexed files. You can specify a location within the bucket for your metadata files using the console or the `S3Prefix` field of the `DocumentsMetadataConfiguration` parameter when you create an S3 data source. If you don't specify an S3 prefix, your metadata files must be stored in the same location as your indexed documents.

If you specify an S3 prefix for your metadata files, they live in a directory structure parallel to your indexed documents.

The following examples show how the indexed document location maps to the metadata file location. Note that the document's S3 key is appended to the metadata's S3 prefix and then suffixed with `.metatdata.json` to form the metadata file's S3 path.

```
Bucket name:
     s3://bucketName
Document path:
     documents
Metadata path:
     none
File mapping
     s3://bucketName/documents/file.txt ->
        s3://bucketName/documents/file.txt.metadata.json
```

```
Bucket name:
     s3://bucketName
Document path:
     documents/legal
Metadata path:
     metadata
File mapping
     s3://bucketName/documents/legal/file.txt ->
        s3://bucketName/metadata/documents/legal/file.txt.metadata.json
```

Your document metadata is defined in a JSON file. The file must be a UTF-8 text file without a BOM marker. The file name of the JSON file should be *`document`*`.`*`extension`*`.metadata.json`, where "document" is the name of the document that the metadata applies to and "extension" is the file extension for the document.

The content of the JSON file follows this template. All of the attributes are optional.

```
{
    "DocumentId": "document ID",
    "Attributes": {
        "_category": "document category",
        "_created_at": "ISO 8601 encoded string",
        "_last_updated_at": "ISO 8601 encoded string",
        "_version": "file version",
        "_view_count": number of times document has been viewed,
        "custom attribute key": "custom attribute value",
        additional custom attributes
    },
    "AccessControlList": [
         "Name": "user name",
         "Type": "GROUP | USER",
         "Access": "ALLOW | DENY"
```

```
    ],
    "Title": "document title",
    "ContentType": "HTML | MS_WORD | PDF | PLAIN_TEXT | PPT"
    }
}
```

The `_created_at` and `_last_updated_at` metadata fields are ISO 8601 encoded dates. For example, "2019-09-24T01:04:41Z".

You can add additional information to the `Attributes` field about a document that you use to filter queries or to group query responses. For more information, see Creating Custom Document Attributes (p. 51).

The `AccessControlList` field enables you to filter the response from a query so that only certain users and groups have access to documents. For more information, see Filtering on User Context (p. 65).

Using a Database Data Source

You can index documents stored in a database using a database data source. After you provided connection information for the database Amazon Kendra will connect and index documents. Amazon Kendra supports the following databases:

- Amazon Aurora MySQL
- Amazon Aurora PostgreSQL
- Amazon RDS for MySQL
- Amazon RDS for PostreSQL

Before you create a database data source, you need to create an index and create custom fields in the index for the data from the database. For more information, see Creating an Index (p. 41) and Mapping Data Source Fields (p. 52).

To use a database data source, you need to identify the following:

- Connection information such as credentials for the database stored in AWS Secrets Manager, the host name, port, and name of the data table that contains the document data.
- Column information such as the names of the columns in the data table that contain the document data and document ID, one to five columns to detect if a document has changed, and optional data table columns that map to custom index fields. You can map any of the Amazon Kendra reserved field names to a table column.
- Optionally, VPC information to connect to the database server. For more information about using a VPC, see Configuring Amazon Kendra to use a VPC (p. 54). If you are using a database data source with a VPC, the subnets provided in the VPC configuration must be in one of the following availability zone IDs:
  - US West (Oregon) – usw2-az1, usw2-az2, usw2-az3
  - US East (N. Virginia) – use1-az1, use1-az2, use1-az4
  - EU (Ireland) – euw1-az1, uew1-az2, euw1-az3

Database configuration provides the information required to connect to your database server. The host and port tell Amazon Kendra where to find the database server on the internet, the database name and table name tell Amazon Kendra where to find the document data on the database server.

To enable Amazon Kendra to access your documents, you must specify a user that has read access to the table that contains the documents. Amazon Kendra requires credentials for the user to access the database. You provide these credentials using AWS Secrets Manager. Once you have created the secret, you provide the Amazon Resource Name (ARN) of the secret to Amazon Kendra. The secret must contain

the user name and password that Amazon Kendra uses to access the database in a JSON structure. The secret may contain additional information, but Amazon Kendra uses only the user name and password. The following is the minimum JSON structure that must be in the secret:

```
{
    "username": "user name",
    "password": "password"
}
```

The secret can contain more information, however, Amazon Kendra ignores other fields. For more information, see What Is AWS Secrets Manager in the *AWS Secrets Manager User Guide*.

The following example shows a database configuration.

```
"DatabaseConfiguration": {
    "ConnectionConfiguration": {
        "DatabaseHost": "host.subdomain.domain.tld",
        "DatabaseName": "DocumentDatabase",
        "DatabasePort": 3306,
        "SecretArn": "arn:aws:secretmanager:region:account ID:secret/secret name",
        "TableName": "DocumentTable"
    }
}
```

Note
The `DatabaseHost` field should be the RDS instance endpoint for the database. Don't use the cluster endpoint.

You add document table information to an index by mapping table columns to index fields. There are two types of information that you add. The first is one to five columns that Amazon Kendra uses to determine if a document has changed since the last time that an index update was run. For example, if you have columns in your table named `LastUpdateDate` and `LastUpdateTime` you can tell Amazon Kendra to use them to determine if a document was updated.

The second type of information about columns is to map some or all of the columns in your table to index fields. For example, you can map a column that contains the document abstract to an index field. If you mark the field searchable, Amazon Kendra will use the contents of the field when determining if a document matches the query. For more information about the attributes that you can assign a custom field, see Mapping Data Source Fields (p. 52).

After you map the columns you can also use the index fields as custom attributes to filter the results of a query. For more information, see Filtering Queries (p. 63).

The following example shows a simple column configuration for a database data source.

```
"ColumnConfiguration": {
    "ChangeDetectingColumns": [
        "LastUpdateDate",
        "LastUpdateTime"
    ],
    "DocumentDataColumnName": "TextColumn",
    "DocumentIdColumnName": "IdentifierColumn",
    "DocoumentTitleColumnName": "TitleColumn",
    "FieldMappings": [
        {
            "DataSourceFieldName": "AbstractColumn",
            "IndexFieldName": "Abstract"
        }
    ]
}
```

Using a Microsoft SharePoint Data Source

You can use your Microsoft SharePoint Online site as a data source for Amazon Kendra. When you use Amazon Kendra to index your site, you choose which SharePoint URLs to include in the index, and you specify inclusion and exclusion patterns for the documents stored on those URLs.

Amazon Kendra requires credentials to access the SharePoint site. The SharePoint user must have administrative permission to the SharePoint sites that you want to index. If you are using the console to create your data source, you can enter the credentials there or you can choose an existing AWS Secrets Manager secret. If you are using the API, you must provide the Amazon Resource Name (ARN) of the secret.

The secret must contain the user name and password that Amazon Kendra uses to access the SharePoint site in a JSON structure. The following is the minimum JSON structure that must be in the secret:

```
{
    "username": "user name",
    "password": "password"
}
```

The data source IAM role must have permission to access the secret. For more information, see IAM Roles for Database Data Sources (p. 35) and IAM Roles for SharePoint Online Data Sources (p. 37).

The secret can contain more information, however, Amazon Kendra ignores other fields. For more information, see What Is AWS Secrets Manager in the *AWS Secrets Manager User Guide*.

You must create an index before you create the SharePoint data source. For more information, see Creating an Index (p. 41). You provide the ID of the index when you create the data source.

You specify connection and other information in the console or using an instance of the SharePointConfiguration (p. 201) data type. You must provide the following information:

- The credentials required to log in to the SharePoint site.
- The URLs of the SharePoint site, SharePoint site collection, or Sharepoint list to index.
- The ARN of an IAM role that has permission to run Amazon Kendra commands. For the required permissions, see IAM Roles for SharePoint Online Data Sources (p. 37).

You can optionally provide the following information:

- Whether Amazon Kendra should index the contents of attachments to SharePoint list items.
- Field mappings that map attributes from your SharePoint site to Amazon Kendra index fields. For more information, see Mapping Data Source Fields (p. 52).

Creating Custom Document Attributes

When the source of your data is an S3 bucket, you can apply custom attributes to your documents. For example, you could create a custom attribute called "Department" with values "HR", "Sales", and "Manufacturing". You can apply these attributes to your documents so that you can limit the response to documents in the "HR" department, for example.

For Microsoft SharePoint and database data sources, you use field mapping for the same purpose. For more information, see Mapping Data Source Fields (p. 52).

Amazon Kendra has six reserved attributes that you can use. The attributes are:

- `_category` (String)
- `_created_at` (ISO 8601 encoded string)
- `_last_updated_at` (ISO 8601 encoded string)
- `_source_uri` (String)
- `_version` (String)
- `_view_count` (Long)

After you have created a custom attribute, you can use the attribute when you call the `Query` operation. You can use it for faceted search, use it to filter the response, and choose whether or not the attribute should be returned in the response. For more information, see Filtering Queries (p. 63).

Adding Custom Attributes with the BatchPutDocument Operation

When you use the BatchPutDocument (p. 92) operation to add a document to your index, you specify custom attributes as part of the `Attributes` structure. You can add multiple attributes when you call the operation. The following example is a `Attributes` structure that adds "Department" and "_category" attributes to a document.

```
"Attributes":
    {
        "Department": "HR",
        "_category": "Vacation policy"
    }
```

Adding Custom Attributes to an S3 Data Source

When you use an Amazon S3 bucket as a data source for your index, you add metadata to the documents with companion metadata files. You place the metadata JSON files in a directory structure that is parallel to your documents. For more information, see S3 Document Metadata (p. 48).

You specify custom attributes in the `Attributes` JSON structure. You can add a list of attributes. For example, the following example is an `Attributes` structure that defines three custom attributes and one reserved attribute.

```
"Attributes": {
        "brand": "Amazon Basics",
        "price": 1595,
        "_category": "sports",
        "subcategories": ["outdoors", "electronics"]
    }
```

Mapping Data Source Fields

When the source of you data is a Microsoft SharePoint site or a database you can map SharePoint attributes or database columns to fields in your index. For example if you have a SharePoint attribute or a database column that contains department information for a document, you can map it to an index field called "Department" so that you can use the field in queries.

If you are storing your documents in an Amazon S3 bucket, or if you are using an Amazon S3 data source, you use custom attributes for the same purpose. For more information, see Creating Custom Document Attributes (p. 51).

Mapping your data source attributes or columns to an index field is a three step process:

1. Create an index. For more information, see Creating an Index (p. 41).
2. Update the index to add custom fields.
3. Create a data source that maps attributes or columns to the index fields.

To update the index to add custom fields, you use the console or the UpdateIndex (p. 149) operation.

When you are using the console, you can choose to map a database column to one of the seven reserved field names, or you can choose to create a new index field that maps to the column. If the name of the database column matches the name of a reserved field, the field and column are automatically mapped.

With the API, you add custom and reserved fields using the `DocumentMetadataConfigurationUpdates` parameter.

The following JSON example is a `DocumentMetadataConfigurationUpdates` structure that adds a field called "Department" to the index.

```
"DocumentmetadataConfigurationUpdates": [
   {
       "Name": "Department",
       "Type": "STRING_VALUE"
   }
]
```

When you create the field you have the option of setting how the field should be used in searches. You can choose from the following:

- **Displayable** – determines whether the field is returned in the query response. The default is `true`.
- **Facetable** – indicates that the field can be used to create facets. The default is `false`.
- **Searchable** – Determines whether the field is used in the search. The default is `true` for string fields and `false` for number and date fields.

The following JSON example is a `DocumentMetadataConfigurationUpdates` structure that adds a field called "Department" to the index and marks it as facetable.

```
"DocumentMetadataConfigurationUpdates": [
   {
       "Name": "Department",
       "Type": "STRING_VALUE",
       "Search": {
           "Facetable": "true"
       }
   }
]
```

Amazon Kendra has six reserved fields that you can map to database columns or SharePoint attributes. The fields are:

- `_category` (String)
- `_created_at` (ISO 8601 encoded string)
- `_file_type` (String)
- `_last_updated_at` (ISO 8601 encoded string)
- `_source_uri` (String)
- `_view_count` (Long)

Once you have created the index fields, you can map the fields to database columns or SharePoint attributes.

To map fields to a database data source, see Using a Database Data Source (p. 49).

To map fields to a SharePoint data source, see Using a Microsoft SharePoint Data Source (p. 51).

Configuring Amazon Kendra to use a VPC

Amazon Kendra connects to your Amazon virtual private cloud (VPC) to index information stored in databases running in your private cloud. When you create the database data source you provide security group and subnet identifiers for the subnet that contains your database. Amazon Kendra uses this information to create an elastic network interface that it uses to securely communicate with your database.

If your database isn't running on an Amazon VPC, you can connect your database to your Amazon VPC using a Virtual Private Network (VPN). You get a default VPC when you create your Amazon account. For information on setting up a VPN, see the AWS Virtual Private Network Documentation.

To use a VPC, you must tell Amazon Kendra the identifier of the subnet that the database belongs to and the identifiers of any security groups that Amazon Kendra must use to access the subnet. For example, if you are using the default port for a MySQL database, the security groups must enable Amazon Kendra to access port 3306 on the host that runs the database.

The identifiers for subnets and security groups are configured in the Amazon VPC control panel. To see the identifiers, open the Amazon VPC console as follows:

To view subnet identifiers

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. From the left menu, choose **Subnets**.
3. From the subnet list, choose the subnet that contains your database server.
4. In the description tab, the identifier is in the **Subnet ID** field.

To view security group identifiers

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. From the left menu, choose **Security Groups**.
3. From the security group list, choose the group that you want the identifier for.
4. In the description tab, the identifier is in the **Group ID** field.

You can provide more than one subnet if Amazon Kendra must route the connection between two or more subnets. For example, if the subnet that contains your database server is out of IP addresses, Amazon Kendra can connect to a subnet with free IP addresses and route the connection to the first subnet. If you list multiple subnets, the subnets must be able to communicate with each other. Each subnet should be associated with a route table that provides outbound internet access using a network address translator (NAT) device.

You can also provide multiple security groups. The combined effect of the security groups should allow Amazon Kendra to access the database server that you have specified in the connection configuration for the data source.

Connecting to a Database in a VPC

The following example shows how to connect a database data source to a MySQL database running in a VPC. The example assumes that you are starting with your default VPC and that you need to create a MySQL database. If you already have a VPC, make sure that it is configured as shown. If you have a MySQL database, you can use that instead of creating a new one.

Topics

Step 1: Configure a VPC

First, configure your VPC so that you have a private subnet and a security group that enables Amazon Kendra to access a MySQL database running in the subnet.

To configure a VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. From the left menu, choose **Route tables** then choose **Create route table**.
3. For the **Name tag** field, enter **`Private subnet route table`**. In the **VPC field**, choose your VPC, and then choose **Create**. Choose **Close** to return to the list of route tables.
4. From the left menu, choose **NAT Gateways** then choose **Create NAT Gateway**.
5. In the **Subnet** field, choose the subnet that should be the public subnet and note the subnet ID.
6. If you don't have an available Elastic IP, choose **Create New EIP** and then choose **Create a NAT Gateway** and then choose **Close**.
7. From the left menu, choose **Route Tables**.
8. From the route table list, choose **Private subnet route table** that you created in step 3. From the **Actions** menu, choose **Edit Routes**.
9. Choose **Add route**. Add the destination 0.0.0.0/0 to allow all outgoing traffic to the Internet. In the **Target**, choose **NAT Gateway** and then choose the gateway that you created in step 4. Finally, choose **Save routes** and then choose **Close**.
10. From **Actions**, choose **Edit subnet associations.**
11. Select the subnets that you want to be private. Do not select the subnet with the NAT gateway that you noted above.

Step 2: Configure Security

Next, configure security groups for your database.

To create security groups

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. From the description of your VPC, note the IPv4 CIDR.
3. From the left menu, choose **Security Groups** and then choose **Create security group**.
4. In **Security group name** type **`MySQLSecurityGroup`**. Provide a description, then choose your VPC from the drop down list. Choose **Create** and then choose **Close**.

5. Choose the **Inbound** tab.
6. Choose **Edit rules** and then choose **Add Rule**
7. For a MySQL database, type 3306 for the **Port Range**. For the **Source**, type the CIDR of your VPC. Choose **Save rules** and then choose **Close**.

The security group allows anyone within the VPC to connect to the database, and it allows outbound connections to the Internet.

Step 3: Create a Database

Now create a database to hold your documents. If you already have a database, you can use that instead.

To create a MySQL database

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. From the left menu, choose **Subnet groups** and then choose **Create DB Subnet Group**.
3. Name the group and choose your VPC.
4. Add your VPC's private subnets. Your private subnets are the ones that are not connected to your NAT. Choose **Create**.
5. From the left menu, choose **Databases** and then choose **Create database.**
6. Use the following parameters to create the database. Leave all of the other parameters at their defaults.
   - **Engine options** – MySQL
   - **Templates** – Free tier
   - **Credential Settings** – Enter and confirm a master password
   - Under **Connectivity**, choose **Additional connectivity configuration**
     - **Subnet group** – Choose the subnet group that you created in step 4
     - **VPC security group** – Choose the group that you created in the VPC (**`MySQLSecurityGroup`**).
   - Under **Additional configuration**, set the **Initial database name** to **`content`**.
7. Choose **Create database**.
8. From the list of databases, choose your new database. Make a note of the database endpoint.
9. After you create your database, you must create a table to hold your documents. Creating a table is outside the scope of these instructions. When you create your table, note the following:
   - Database name – **`content`**
   - Table name – **`documents`**
   - Columns – **`ID`**, **`Title`**, **`Body`**, and **`LastUpdate`**. You can include additional columns if you want.

Step 4: Create a MySQL Data Source

Now that you have configured your VPC and created your database, you can create a data source for the database.

To create a MySQL data source

1. Sign in to the AWS Management Console and open the Amazon Kendra console at https://console.aws.amazon.com/kendra/.
2. From the left menu, choose **Indexes** and then choose your index.
3. Choose **Add data sources** and then choose **Amazon RDS**.

4. Type a name and description for the data source and then choose **Next**.
5. Choose **MySQL**.
6. Under **Connection access** enter the following information:
   - **Endpoint** – The endpoint of the database that you created earlier.
   - **Port** – The port number for the database. For MySQL, the default is 3306.
   - **Type of authentication** – Choose **New**.
   - **New secret container name** – type a name for the Secrets Manager container for the database credentials.
   - **Username** – Type the name of a user with administrative access to the database.
   - **Password** – Type the password for the user and then choose **Save authentication**.
   - **Database name** – type **`content`**.
   - **Table name** – type **`documents`**.
   - **IAM role** – Choose **Create a new role** and then type a name for the role.
7. In **Column configuration** provide the following:
   - **Document ID column name** – type **`ID`**.
   - **Document title column name** – type **`Title`**.
   - **Document data column name** – type **`Body`**.
8. In **Column change detection** provide the following:
   - **Change detecting columns** – type **`LastUpdate`**.
9. In **Configure VPC & security group** provide the following:
   - In **Virtual Private Cloud (VPC)** choose your VPC.
   - In **Subnets** choose the private subnets that you created in your VPC.
   - In **VPC security groups** select the security group that you created in your VPC for MySQL databases (**`MySQLSecurityGroup`**).
10. In **Set sync run schedule** choose **Run on demand** and then choose **Next**.
11. In **Data source field mapping**, choose **Next**.
12. Review the configuration of your data source to make sure that it is correct. When you're satisfied that everything is correct, choose **Create**.

Troubleshooting

When you synchronize your Amazon Kendra index with a data source, you may run into issues that prevent the documents from being indexed. Indexing is a two step process. First, there is the data source level process of crawling the data source to find the new and updated documents to index, and to find any documents to remove from the index. Second, there is the document level process where each document is accessed and indexed.

An error can occur in either of these steps. Data source level errors are reported in the console in the **Sync run history** section of the data source details page. The status of the synchronization job can be **Succeeded**, **Incomplete**, or **Failed**. You can also see the number of documents indexed and deleted during the job. If the status is **Failed**, a message is shown in the **Details** column.

Document level errors are reported in Amazon CloudWatch Logs. You can see the errors using the CloudWatch console.

My Synchronization Job Failed

A synchronization job typically fails when there is a configuration error in the index or the data source. The error message in the Details column of the data source gives information about what went wrong. The problem is usually that the index or the data source does not have the proper IAM permissions. The error message describes the permissions that are missing. Here are some of the error messages that you can receive:

```
Failed to create log group for job. Please make sure that the IAM role provided
has sufficient permissions.
```

If your index role does not have permission to use CloudWatch, the data source will not be able to create a CloudWatch log. If you get this error you need to add CloudWatch permissions to the index role.

```
Failed to access S3 file prefix (bucket name) while trying to crawl your
metadata files. Please make sure the IAM Role (role ARN) provided has
sufficient permissions.
```

When you are using an Amazon S3 data source, Amazon Kendra must have permission to access the bucket that contains the documents. You need to add permission for Amazon Kendra to read the bucket to the data source IAM role.

```
The provided IAM Role (role ARN) could not be assumed. Please make sure Amazon
Kendra is a trusted entity that is allowed to assume the role.
```

Amazon Kendra needs permission to assume the index and data source IAM roles. You need to add a trust policy to the roles with permission for the `sts:AssumeRole` action.

For the IAM polices that Amazon Kendra needs to index a data source, see IAM Access Roles for Amazon Kendra (p. 32).

My Synchronization Job is Incomplete

Jobs are generally incomplete when they have completed the data source level process but have had some error during the document level process. When a job is incomplete, some of the documents may have been successfully indexed. For an Amazon S3 data source, an incomplete job is typically caused by:.

- The metadata for one or more documents was invalid.
- When there are documents to submit for indexing, at least one document was not submitted.
- When there are documents to submit for deleting from the index, at least one document was not submitted.

To troubleshoot an incomplete synchronization job, look first to your CloudWatch logs.

1. From the details column, choose **View details in CloudWatch**.
2. Review the error messages to see what caused the document to fail.

My Synchronization Succeeded But There Are No Indexed Documents

Occasionally you will have a index synchronization job run that is marked as **Succeeded** but there are no new or updated documents indexed when you expect there to be. Here are some reasons:

- Check CloudWatch `DocumentsSubmittedForIndexingFailed` metric to see if there were any documents that failed to synchronize. Check your CloudWatch logs for details.
- For an Amazon S3 data source, you may have given Amazon Kendra the wrong bucket name or prefix. Make sure that the bucket that Amazon Kendra is using is the one that contains the documents to index.
- If you are re-indexing a document that failed to be indexed in an earlier job, Amazon Kendra won't index it unless you make a change to the document or its associated metadata file.

General Troubleshooting

Amazon Kendra uses CloudWatch metrics and logs to provide you with insight into synchronizing your data sources. You can use the metrics and logs to determine what went wrong with a synchronization run and to determine what you need to do to fix it.

For general troubleshooting, start with your CloudWatch metrics.

- Check the `DocumentsCrawled` metric to see how many documents your data source checked. For an Amazon S3 bucket, if the number is less than you expect, check to be sure that your data source is pointing to the right bucket.
- Check the `DocumentsSkippedNoChange` metric to see how many documents were skipped because they haven't changed since the last synchronization. If the number does not match what you expect, check to make sure that your repository was updated correctly.
- Check the `DocumentsSkippedInvalidMetadata` metric to see how many documents had invalid metadata. Check your CloudWatch logs to see the specific errors that occurred.
- Check the `DocumentsSubmittedForIndexingFailed` metric to see the number of documents that were sent from the data source to the index but failed to be indexed for some reason. For example, if you use a metadata attribute in an Amazon S3 data source that hasn't been defined as a custom index field the document will not be indexed. Check your CloudWatch logs to see the specific errors that occurred.
- Check the `DocumentsSubmittedForDeletionFailed` metric to see how many documents that the data source attempted to remove from the index failed to be deleted from the index. Check your CloudWatch logs to see the specific errors that occurred.

You can look at the CloudWatch logs for a particular synchronization run to get details of the errors that occurred during the run. For more information about CloudWatch logs with Amazon Kendra, see Monitoring Amazon Kendra with Amazon CloudWatch Logs (p. 81).

Searching Indexes

To search an Amazon Kendra index you use the the section called "Query" (p. 133) operation. This section shows you how to make a query, perform filters, and interpret the response that you get from a `Query` operation. This section also shows how to submit feedback about the quality of a search result.

Topics

Querying an Index

When you search your index, Amazon Kendra uses all of the information that you provided about your documents to determine the documents most relevant to the search terms entered. Some of the items that Amazon Kendra considers are:

- The text of the document.
- The title of the document.
- Custom text fields that you have marked searchable.
- The date field that you have indicated should be used to determine the "freshness" of a document.

Once a set of relevant documents has been selected from the index, Amazon Kendra filters the response based on any attribute filters that you have requested for the search. For example, if you have a custom attribute called "department," you can filter the response so that it only contains documents from a department called "legal." For information about creating custom attributes, see Creating Custom Document Attributes (p. 51).

After finding the relevant documents and then filtering based on the attributes that you set, Amazon Kendra returns the results. The results are sorted by the relevance that Amazon Kendra determined for each doc. The results are paginated so that you can show a page at a time to your user.

The following Python example shows how to search an index by using the the section called "Query" (p. 133) operation. The example determines the type of the search result (answer, document, question/answer) and displays the answer text.

For information about the query responses, see Query Responses (p. 68).

Note
You can use this code to filter document attributes. The topic Filtering Queries (p. 63) contains examples that you can use to replace the following code.

```
response=kendra.query(
        QueryText = query,
        IndexId = index)
```

Prerequisites

To run this example, you need to:

- Set up permissions. For more information, see IAM Access Roles for Amazon Kendra (p. 32).
- Set up the AWS CLI. For more information, see Setting Up the AWS CLI (p. 8).
- Create a data source and index. For more information, see Getting Started with the Console (p. 13).

Searching an Index (Console)

You can use the Amazon Kendra search console to test your index. You can make queries and see the results.

To search an index with the console

1. Sign in to the AWS Management Console and open the Amazon Kendra console at http://console.aws.amazon.com/kendra/.
2. From the left menu, choose **Indexes**.
3. Choose your index.
4. From the left menu, choose **Search console**.
5. Enter a query in the text box and then press enter or choose the magnifying glass icon.
6. Amazon Kendra returns the results of the search.

Searching an Index (Python)

To search an index with Python

- Use the following code. Change the value of `query` to your search query and `index_id` to the index identifier of the index that you want to search.

```
import boto3
import pprint

kendra = boto3.client('kendra')

query='${searchString}'
index_id='${indexID}'

response=kendra.query(
        QueryText = query,
        IndexId = index_id)

print ('\nSearch results for query: ' + query + '\n')

for query_result in response['ResultItems']:

    print('-------------------')
    print('Type: ' + str(query_result['Type']))

    if query_result['Type']=='ANSWER':
        answer_text = query_result['DocumentExcerpt']['Text']
        print(answer_text)

    if query_result['Type']=='DOCUMENT':
        if 'DocumentTitle' in query_result:
```

```
            document_title = query_result['DocumentTitle']['Text']
            print('Title: ' + document_title)
        document_text = query_result['DocumentExcerpt']['Text']
        print(document_text)

    print ('------------------\n\n')
```

Filtering Queries

You can provide input parameters to filter the response from a call to the section called "Query" (p. 133). You can create faceted search suggestions, you can use boolean logic to filter out documents that don't match a specified criteria, or you can filter out specific document attributes from the response.

Note
The example code in this section can be used with the example code in Querying an Index (p. 61).

Facets

Facets are scoped views into a set of search results. For example, if your index provides search results for cities across the world, you can use facets to offer your users search results filtered to a specific city. Or, you can create facets for search results that are written by a specific author. The following example shows how to get facet information for the `_category` custom attribute.

```
response=kendra.query(
        QueryText = query,
        IndexId = index,
        Facets = [
            {
                "DocumentAttributeKey" : "_category"
            }
        ]
        )
```

Facet information is returned in the `FacetResults` response array. You use the contents to display faceted search suggestions in your application. For example, if the document attribute `Category` contains the city that a search could apply to, use that information to display a list of city searches. Users can select a city to get faceted search results scoped to the city. To make the faceted search, call Query (p. 133) and use the selected document attribute to filter the results. For an example, see Using Document Attributes to Filter Queries (p. 64).

The following sample JSON response shows facets scoped to the `_category` document attribute. The response includes the count of documents that include the value of the document attribute.

```
{
    'FacetResults': [
        {
            'DocumentAttributeKey': '_category',
            'DocumentAttributeValueCountPairs': [
                {
                    'Count': 3,
                    'DocumentAttributeValue': {
                        'StringValue': 'Dubai'
                    }
                },
```

```
                {
                    'Count': 3,
                    'DocumentAttributeValue': {
                        'StringValue': 'Seattle'
                    }
                },
                {
                    'Count': 1,
                    'DocumentAttributeValue': {
                        'StringValue': 'Paris'
                    }
                }
            ]
        }
    ]
```

When you use a string list field to create facets, the facet results returned are based on the contents of the string list. For example, if you have a string list field that contains two items, one with the value "corgi","dachshund," and one with the value "husky," you will get a `FacetsResults` structure with three facets.

For more information, see Query Responses (p. 68).

Using Document Attributes to Filter Queries

By default, `Query` returns all search results. To filter responses, you can perform logical operations on the document attributes. For example, if you only want documents for a specific city, you can filter on the `City` and `State` custom document attributes. You use the AttributeFilter (p. 157) input parameter to create a boolean operation on filters that you supply.

The following example shows how to perform a logical AND operation by filtering on a specific city, *Seattle*, and state, *Washington*.

```
response=kendra.query(
        QueryText = query,
        IndexId = index,
        AttributeFilter = {'AndAllFilters':
            [
                {"EqualsTo": {"Key": "City","Value": {"StringValue": "Seattle"}}},
                {"EqualsTo": {"Key": "State","Value": {"StringValue": "Washington"}}}
            ]
            }
        )
```

The following example shows how to perform a logical OR operation for when any of the `Fileformat`, `Author`, or `SourceURI` keys match the specified values.

```
response=kendra.query(
        QueryText = query,
        IndexId = index,
        AttributeFilter = {'OrAllFilters':
            [
                {"EqualsTo": {"Key": "Fileformat","Value": {"StringValue":
 "AUTO_DETECT"}}},
                {"EqualsTo": {"Key": "Author","Value": {"StringValue": "Ana Carolina"}}},
                {"EqualsTo": {"Key": "SourceURI","Value": {"StringValue": "https://
aws.amazonaws.com/234234242342"}}}
            ]
            }
        )
```

For `StringList` fields, use the `ContainsAny` or `ContainsAll` attribute filters to return documents with the specified string. The following example shows how to return all documents that have the values "Seattle" or "Portland" in their `Locations` custom attribute.

```
response=kendra.query(
        QueryText = query,
        IndexId = index,
        AttributeFilter = {
                "ContainsAny": { "Key": "Locations", "Value": { "StringListValue":
 [ "Seattle", "Portland"] }}
            }
        )
```

Filtering a Document's Attributes

By default, all document attributes for a document are returned in the `DocumentAttributes` response field. You can choose to only include specific document attributes by using the `IncludeDocumentAttributes` input parameter. In the following example, only the `SourceURI` and `Author` document attributes are included in the response for a document.

```
response=kendra.query(
        QueryText = query,
        IndexId = index,
        IncludeDocumentAttributes = ["SourceURI", "Author"]
        )
```

Filtering on User Context

Amazon Kendra enables you to filter a query response to remove documents from the result based on a user name and group membership. User context filtering depends on two things:

- Setting attributes on the indexed documents.
- Including user and group information when querying the index.

Use user context to filter documents based on user and group membership. If no user and group information is included in the query, Amazon Kendra returns all documents.

You can add a user or a group to either an allow list or a deny list for the document. If a user or group is added to the deny list, the document is filtered out of any query that contains the user or group.

You specify the user and group information when you query the index using the built-in attributes "_user_id" and "_group_ids". You can set up to 100 group identifiers.

Amazon Kendra does not match users to groups. When you use user-context filtering, you must include all of the groups that a user belongs to when you make the query. For example, if a user belongs to two groups, "HR" and "IT," you must include both groups in the request.

The following example shows a request that filters the query response based on the user ID and groups. The query will return any document that has the user or the "HR" or "IT" groups in the allow list. If the user or either group is in the deny list for a document, the document will not be returned.

```
response = kendra.query(
    QueryText = query,
    IndexId = index,
```

```
AttributeFilter = {
    "OrAllFilters": [
        {
            "EqualsTo": {
                "Key": "_user_id",
                "Value": {
                    "StringValue": "user1"
                }
             }
        },
        {
            "EqualsTo": {
                "Key": "_group_ids",
                "Value": {
                    "StringListValue": ["HR", "IT"]
                }
            }
        }
    ]
}
)
```

Note
User context filtering isn't an authentication or authorization control for your content. It doesn't do user authentication on the user and groups sent to the `Query` operation. It is up to your application to ensure that the user and group information sent to `Query` operation is authenticated and authorized.

There is an implementation of user context filtering for each data source. The following section describes each implementation.

Topics

User Context Filtering for Documents Added Directly to an Index

To specify user context filters for a document that you add directly to an index, you provide the filter in the `AccessControlList` field of the document. You provide three pieces of information:

- The access that the entity should have. You can say `ALLOW` or `DENY`.
- The type of entity. You can say `USER` or `GROUP`.
- The name of the entity.

You can add up to 200 entries in the `AccessControlList` field.

User Context Filtering for Frequently Asked Questions

You can't add user context filtering to frequently asked question (FAQ) index entries.

User-Context Filtering for Database Data Sources

For a database data source, information for user context filtering comes from a column in the source table. You specify the column name in the console or when you create the data source using the `AclConfiguration` field.

A database data source has the following limitations:

- You can only specify an allow list for a database data source. You can't specify a deny list.
- You can only specify groups. You can't specify individual users for the allow list.
- The database column should be string containing a semi-colon delimited lists of groups.

User Context Filtering for Amazon S3 Data Sources

You add user-context filtering to a document in an Amazon S3 data source using a metadata file associated with the document. You add the information to the `AccessControlList` field in the JSON document. For more information about adding metadata to the documents indexed from an Amazon S3 data source, see S3 Document Metadata (p. 48).

You provide three pieces of information:

- The access that the entity should have. You can say `ALLOW` or `DENY`.
- The type of entity. You can say `USER` or `GROUP`.
- The name of the entity.

You can add up to 200 entries in the `AccessControlList` field.

User Context Filtering for Microsoft SharePoint

Amazon Kendra retrieves user and group information from Microsoft SharePoint when it indexes the documents on the site. To filter your documents, provide user and group information when you call the `Query` operation.

To filter using a user name, use the user's email address. For example, johnstiles@example.com.

When you use a SharePoint group for user context filtering, calculate the group ID as follows:

- Get the site name. For example, `https://host.example.com/sites/siteName.`
- Take the MD5 hash of the site name. For example, `430a6b90503eef95c89295c8999c7981`.
- Create the group ID by concatenating the MD5 hash with a vertical bar (|) and the group name. For example, if the group name is "site owners", the group ID would be

  `"430a6b90503eef95c89295c8999c7981|site owners"`

Send the group ID to Amazon Kendra as the _group_ids attribute when you call the Query (p. 133) operation. For example, the AWS CLI command looks like this:

```
aws kendra  query \
                --index-id index ID
                --query-text "query text"
                --attribute-filter '{
                   "EqualsTo":{
                     "Key": "_group_ids",
                     "Value": {"StringValue": "430a6b90503eef95c89295c8999c7981|site
 owners"}
```

```
                }}'
```

Query Responses

A call to Query (p. 133) returns information about the results of a search. The results are in an array of QueryResultItem (p. 191) objects (`ResultItems`). Each `QueryResultItem` includes a summary of the result. Document attributes associated with the query result are included.

Summary Information

The summary information varies depending on the type of result. In each case it includes document text that matches the search term. It also includes highlight information that you can use to highlight the search text in your application's output. For example, if the search term is *what is the height of the Space Needle?*, the summary information includes text location for the words *height* and *space needle*. For information about response types, see Types of Response (p. 69).

Document Attributes

Each result contains document attributes for the document that matches a query. Some of the attributes are predefined, such as `DocumentId`, `DocumentTitle` and `DocumentUri`. Others are custom attributes that you define. You can use document attributes to filter the response from the `Query` operation. For example, you might want only the documents written by a specific author or a specific version of a document. For more information, see Filtering Queries (p. 63). You specify document attributes when you add documents to an index. For more information, see Creating Custom Document Attributes (p. 51).

The following is sample JSON code for a query result. Note the document attributes in `DocumentAttributes` and `AdditionalAttributes`.

```
{
    "QueryId": "query-id",
    "ResultItems": [
        {
            "Id": "result-id",
            "Type": "ANSWER",
            "AdditionalAttributes": [
                {
                    "Key": "AnswerText",
                    "ValueType": "TEXT_WITH_HIGHLIGHTS_VALUE",
                    "Value": {
                        "TextWithHighlightsValue": {
                            "Text": "text",
                            "Highlights": [
                                {
                                    "BeginOffset": 55,
                                    "EndOffset": 90,
                                    "TopAnswer": false
                                }
                            ]
                        }
                    }
                }
            ],
            "DocumentId": "document-id",
            "DocumentTitle": {
                "Text": "title"
            },
            "DocumentExcerpt": {
                "Text": "text",
                "Highlights": [
                    {
```

```
                        "BeginOffset": 0,
                        "EndOffset": 300,
                        "TopAnswer": false
                    }
                ]
            },
            "DocumentURI": "uri",
            "DocumentAttributes": []
        },
        {
            "Id": "result-id",
            "Type": "DOCUMENT",
            "AdditionalAttributes": [],
            "DocumentId": "document-id",
            "DocumentTitle": {
                "Text": "title",
                "Highlights": []
            },
            "DocumentExcerpt": {
                "Text": "text",
                "Highlights": [
                    {
                        "BeginOffset": 74,
                        "EndOffset": 77,
                        "TopAnswer": false
                    }
                ]
            },
            "DocumentURI": "uri",
            "DocumentAttributes": [
                {
                    "Key": "_source_uri",
                    "Value": {
                        "StringValue": "uri"
                    }
                }
            ]
        }
    ],
    "FacetResults": [],
    "TotalNumberOfResults": number
}
```

Types of Response

Amazon Kendra returns three types of query response.

- Answer
- Document
- Question and answer

The type of the response is returned in the `Type` response field of QueryResultItem (p. 191).

Answer

Amazon Kendra detected one or more question answers in the response. A factoid is the response to a who, what, when, or where question such as *What is the height of the space needle?* Amazon Kendra returns text in the index that best matches the query. The text is in the `AnswerText` field and contains highlight information for the search term within the response text.

```
{
    'AnswerText': {
        'Highlights': [
            {
                'BeginOffset': 271,
                'EndOffset': 279,
                'TopAnswer': False
            },
            {
                'BeginOffset': 481,
                'EndOffset': 489,
                'TopAnswer': False
            },
            {
                'BeginOffset': 547,
                'EndOffset': 555,
                'TopAnswer': False
            },
            {
                'BeginOffset': 764,
                'EndOffset': 772,
                'TopAnswer': False
            }
        ],
        'Text': 'Asynchronousoperationscan\n''alsoprocess
\n''documentsthatareinPDF''format.UsingPDFformatfilesallowsyoutoprocess''multi-
page\n''documents.\n''Forinformationabouthow''AmazonTextractrepresents
\n''documentsasBlockobjects,
        ''seeDocumentsandBlockObjects.\n''\n''\n''\n''Forinformationaboutdocument''limits,
        seeLimitsinAmazonTextract.
\n''\n''\n''\n''TheAmazonTextractsynchronous''operationscanprocessdocumentsstoredinanAmazon
\n''S3Bucketoryoucanpass''base64encodedimagebytes.\n''Formoreinformation,

 see''CallingAmazonTextractSynchronousOperations.''Asynchronousoperationsrequireinputdocuments
\n''tobesuppliedinanAmazon''S3Bucket.'
    },
    'Excerpt': {
        'Highlights': [
            {
                'BeginOffset': 0,
                'EndOffset': 300,
                'TopAnswer': False
            }
        ],
        'Text': 'Asynchronousoperationscan\n''alsoprocess
\n''documentsthatareinPDF''format.UsingPDFformatfilesallowsyoutoprocess''multi-page
\n''documents.\n''ForinformationabouthowAmazon''Textractrepresents\n'''
    },
    'Type': 'ANSWER'
}
```

Document

Amazon Kendra returns ranked documents for those that match the search term. The ranking is based on the confidence that Amazon Kendra has in the accuracy of the search result. Information about the matching document is returned in the QueryResultItem (p. 191). It includes the title of the document, The excerpt includes highlight information for search text and the section of matching text in the document. The URI for matching documents is in the `SourceURI` document attribute. The following sample JSON show the document summary for a matching document.

```
{
    'DocumentTitle': {
```

```
        'Highlights': [
            {
                'BeginOffset': 7,
                'EndOffset': 15,
                'TopAnswer': False
            },
            {
                'BeginOffset': 97,
                'EndOffset': 105,
                'TopAnswer': False
            }
        ],
        'Text': 'AmazonTextractAPIPermissions: Actions,
        \n''Permissions,
        andResourcesReference-''AmazonTextract'
    },
    'Excerpt': {
        'Highlights': [
            {
                'BeginOffset': 68,
                'EndOffset': 76,
                'TopAnswer': False
            },
            {
                'BeginOffset': 121,
                'EndOffset': 129,
                'TopAnswer': False
            }
        ],
        'Text': '...LoggingandMonitoring\tMonitoring
\n''\tCloudWatchMetricsforAmazonTextract
\n''\tLoggingAmazonTextractAPICallswithAWSCloudTrail\n''\tAPIReference\tActions
\tAnalyzeDocument\n''\tDetectDocumentText\n''\tGetDocumentAnalysis...'
    },
    'Type': 'DOCUMENT'
}
```

Question and Answer

A question and answer response is returned when Amazon Kendra matches a question with one of the frequently asked questions in your index. The response includes the matching question and answer in the QueryResultItem (p. 191) field. It also includes highlight information for query terms detected in query string. The following JSON shows a question and answer response. Note that the response includes the question text

```
{
    'AnswerText': {
        'Highlights': [

        ],
        'Text': '605feet'
    },
    'Excerpt': {
        'Highlights': [
            {
                'BeginOffset': 0,
                'EndOffset': 8,
                'TopAnswer': False
            }
        ],
        'Text': '605feet'
    },
    'Type': 'QUESTION_ANSWER',
```

```
    'QuestionText': {
        'Highlights': [
            {
                'BeginOffset': 12,
                'EndOffset': 18,
                'TopAnswer': False
            },
            {
                'BeginOffset': 26,
                'EndOffset': 31,
                'TopAnswer': False
            },
            {
                'BeginOffset': 32,
                'EndOffset': 38,
                'TopAnswer': False
            }
        ],
        'Text': 'whatistheheightoftheSpaceNeedle?'
    }
}
```

For information about adding question and answer text to an index, see Adding Questions and Answers (p. 44)

Submitting Feedback

You can provide feedback about query results returned from an Amazon Kendra index. Amazon Kendra uses this information to improve the underlying search model and provide better search results over time. To submit feedback, you use the SubmitFeedback (p. 143) operation. To identify the query, you supply the `IndexID` of the index that the query applies to, and the `QueryId` returned in the response from the Query (p. 133) operation. Two types of feedback are accepted.

- *Clicks* - Information about which query results were chosen by the end user. The feedback includes the result ID (ResultId) and the Unix timestamp of the date and time that the search result was chosen. Your application will need to collect click information from the activities of your end-users.
- *Relevance* - Information about the relevance of a search result. This is typically information provided by the end-user. The feedback contains the result ID and an relevance indicator (`RELEVANT` or `NOT_RELEVANT`). Relevance information is determined by your end user. To submit relevance feedback, your application needs to provide a feedback mechanism that allows the end-user to choose the appropriate relevance for a query result.

The following example shows how to submit click and relevance feedback. You can submit multiple sets of feedback through the `ClickFeedbackItems` and `RelevanceFeedbackItems` arrays. This example submits a single click and a single relevance feedback item. The current timestamp is used for the timing of the feedback submittal.

To submit feedback for a search

1. Use the following code. Change:

    a. `index_id` to the name of the index that the query applies to.
    b. `query_id` to the query that you want to provide feedback on.
    c. `result_id` to the ID of the query result that you want to provide feedback on. The result ID is returned in the query response.
    d. `relevance_value` to either `RELEVANT` (the query result is relevant) or `NOT_RELEVANT` (the query result is not relevant).

```
import boto3
import time

kendra = boto3.client('kendra',
region_name='${region}',
endpoint_url='${endpoint}')

index_id = '${indexID}'
query_id = '${queryID}'
result_id = '${resultID}'
feedback_item = {'ClickTime': int(time.time()),
    'ResultId':result_id}


relevance_value = 'RELEVANT'
relevance_item = {'RelevanceValue': relevance_value,
    'ResultId':result_id
    }

response=kendra.submit_feedback(
    QueryId = query_id,
    IndexId = index_id,
    ClickFeedbackItems = [feedback_item],
    RelevanceFeedbackItems = [relevance_item]
)


print ('Submitted feedback for query: ' + query_id)
```

2. Run the code. A message is displayed after the feedback has been submitted.

Manually Tuning an Index

Amazon Kendra queries produce search results ranked by their relevance. The searchable fields in the index all contribute to this ranking.

You can modify the effect of field or attribute on the relevance of a document through *relevance tuning*. When you use relevance tuning, a result is given a boost in the response when the query includes terms that match the field or attribute. You also specify how much of a boost the document receives when there is a match. Relevance tuning doesn't cause Amazon Kendra to include a document in the query response, it is only one of the factors that Amazon Kendra uses to determine the relevance of a document.

You can boost certain fields in your index to assign more importance to specific responses. Amazon Kendra enables you to tune for specific data sources or document freshness. For example when searching for "When is re:Invent?" you boost the relevance of document freshness so that the latest date is the suggested answer. Or, in an index of research reports, you could boost a more reputable data source.

Amazon Kendra also supports boosting documents based on votes or view counts which is common in forums and other support knowledge bases. You can combine boosts to, for example, boost documents that are not only viewed more but that are also more recent, like trending news or updates.

You set the amount of boost that a document receives using the `Importance` parameter. The larger the number in this parameter, the more the field or attribute will boost the relevance of the document. When you are tuning your index, you should increase the value of the `Importance` parameter in small increments until you get the effect that you want. Query your index and compare the results to previous queries to determine if you are getting improved search results.

You can specify date, number, or string fields and attributes to tune an index. Each type of field or attribute has specific criteria for when it boosts a result.

- **Date fields and attributes** – There are three specific criteria for date fields, `Duration`, `Freshness` and rank order.

  `Duration` sets the time period that the boost applies to. For example, if you set the time period to 86400 seconds (1 day) the boost begins to drop off after one day. The higher the importance, the faster the boost effect drops off.

  `Freshness` indicates that the field or attribute should be used to determine how "fresh" a document is. For example, if document 1 was created on November 14, and document 2 was created on November 5, document 1 is "fresher" than document 2. The fresher the document, the more this boost is applied. You can only have one `Freshness` field in your index.

  Rank order applies the boost in either ascending or descending order. When you use `ASCENDING`, later dates have precedence. If you specify `DESCENDING`, earlier dates have precedence.

- **Number fields and attributes** – For number fields or attributes you can specify the rank order that Amazon Kendra should use when determining the relevance of the field or attribute. If you specify `ASCENDING` larger numbers are given precedence. If you specify `DESCENDING`, lower numbers have precedence.

- **String fields and attributes** – For string fields or attributes you can create subcategories of a field to give a different boost to different values in the field or attribute. For example, if your are boosting a field or attribute called "Department," you can give a different boost to documents from "HR" to those from "Legal".

You tune the relevance of a field or attribute using the UpdateIndex (p. 149) operation. For example, the following example marks the "_last_updated_at" field as the freshness field for a document.

```
"DocumentMetadataConfigurationUpdates" : [
    "Name": "_last_updated_at",
    "Type": "DATE_VALUE",
    "Relevance": {
        "Freshness": TRUE,
        "Importance": 2
    }
]
```

The following example applies a different importance to values in the "department" field.

```
"DocumentMetadataConfigurationUpdates" : [
    "Name": "department",
    "Type": "STRING_VALUE",
    "Relevance": {
        "Importance": 2,
        "ValueImportanceMap": {
            "HR": 3,
            "Legal": 1
        }
    }
]
```

Monitoring and Logging for Amazon Kendra

Topics

Logging Amazon Kendra API Calls with AWS CloudTrail Logs

Amazon Kendra is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Kendra. CloudTrail captures a subset of API calls from Amazon Kendra as events, including calls from the Amazon Kendra console and from code calls to the Amazon Kendra APIs. If you create a trail, you can enable continuous deliver of CloudTrail events to and Amazon S3 bucket, including events for Amazon Kendra. 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 Kendra, 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 Kendra Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon Kendra, that activity is recorded in a CloudTrail event along with other AWS service events in the CloudTrail **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 Kendra, create a trail. A *trail* is a configuration that enables CloudTrail to deliver events as log files to a specified 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 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:

- 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

CloudTrail logs all Amazon Kendra actions, which are documented in the API Reference (p. 89). For example, calls to the `CreateIndex`, `CreateDataSource`, and `Query` operations generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. For more information, see the CloudTrail userIdentity Element.

Example: Amazon Kendra Log File Entries

A *trail* is a configuration that enables delivery of events as log files to a specified S3 bucket. 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.

Calls to the `Query` operation creates the following entry.

```
       {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "AssumedRole | FederatedUser | IAMUser | Root | SAMLUser |
WebIdentityUser",
                "principalId": "principal ID",
                "arn": "ARN",
                "accountId": "account ID",
                "accessKeyId": "access key ID",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "principal Id",
                        "arn": "ARN",
                        "accountId": "account ID",
                        "userName": "user name"
                    },
                    "webIdFederationData": {

                    },
                    "attributes": {
                        "mfaAuthenticated": "false",
                        "creationDate": "timestamp"
                    }
                }
            },
            "eventTime": "timestamp",
            "eventSource": "kendra.amazonaws.com",
            "eventName": "Query",
            "awsRegion": "region",
            "sourceIPAddress": "source IP address",
            "userAgent": "user agent",
            "requestParameters": {
                "indexId": "index ID"
            },
            "responseElements": null,
            "requestID": "request ID",
            "eventID": "event ID",
            "eventType": "AwsApiCall",
            "recipientAccountId": "account ID"
       },
```

Monitoring Amazon Kendra with Amazon CloudWatch

To track the health of your indexes, use Amazon CloudWatch. With CloudWatch, you can get metrics for document synchronization for your index. You can also set up CloudWatch alarms to be notified when

one or more metrics exceeds a threshold that you define. For example, you can monitor the number of documents submitted to be indexed or the number of documents that failed to be indexed.

You must have the appropriate CloudWatch permissions to monitor Amazon Kendra with CloudWatch. For more information, see Authentication and Access Control for Amazon CloudWatch in the *Amazon CloudWatch User Guide*.

Viewing Amazon Kendra Metrics

View Amazon Kendra metrics using the CloudWatch console.

To view metrics (CloudWatch console)

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose **Metrics**, choose **All Metrics** and then choose **Kendra**.
3. Choose the dimension, choose a metric name, then choose **Add to graph**.
4. Choose a value for the date range. The metric count for the selected date range is displayed in the graph.

Creating an Alarm

A CloudWatch alarm watches a single metric over a specified time period and performs one or more actions: sending a notification to an Amazon Simple Notification Service (Amazon SNS) top or Auto Scaling policy. The actions or actions are based on the value of the metric relative to a given threshold over a number of time periods that you specify. CloudWatch can also send you an Amazon SNS message when the alarm changes state.

CloudWatch alarms invoke actions only when the state changes and has persisted for the period that you specify.

To set an alarm

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose **Alarms** and then choose **Create Alarm**.
3. Choose **Kendra metrics** and then choose a metric.
4. For **Time Range**, choose a time range to monitor, and then choose **Next**.
5. Enter a **Name** and **Description**.
6. For **Whenever**, choose **>=**, and type a maximum value.
7. If you want CloudWatch to send an email when the alarm state is reached, in the **Actions** section, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to**, choose a mailing list or choose **New list** and create a new mailing list
8. Preview the alarm in the **Alarm Preview** section. If you are satisfied with the alarm, choose **Create Alarm**.

CloudWatch Metrics for Index Synchronization Jobs

The following table describes the Amazon Kendra metrics for data source synchronization jobs.

Metric	Description
DocumentsCrawled	The number of documents that the synchronization job scanned or discovered during the run. Dimensions: • IndexId • DataSourceId Unit: Count
DocumentsSubmittedForIndexing	The number of documents that the synchronization job submitted to the index. Dimensions: • IndexId • DataSourceId Unit: Count
DocumentsSubmittedForIndexingFailed	The number of documents that failed indexing. Check the contents of the CloudWatch log for the synchronization job for details. Dimensions: • IndexId • DataSourceId Unit: Count
DocumentsSubmittedForDeletion	The number of documents that the synchronization job asked to be removed from the index. Dimensions: • IndexId • DataSourceId Unit: Count
DocumentsSubmittedForDeletionFailed	The number of documents that failed to be deleted. Check the contents of the CloudWatch log for the synchronization job for details. Dimensions: • IndexId • DataSourceId

Metric	Description
	Unit: Count

Metrics for Amazon S3 Data Sources

The following table describes the Amazon Kendra metrics for data source synchronization jobs.

Metric	Description
`DocumentsSkippedNoChange`	The number of documents that were examined and found not to have changed so they weren't submitted for indexing. Dimensions: • IndexId • DataSourceId Unit: Count
`DocumentsSkippedInvalidMetadata`	The number of documents that were skipped because there was a problem with the associated metadata file. Check the contents of the CloudWatch log for the synchronization run for details. Dimensions: • IndexId • DataSourceId Unit: Count
`MetadataFilesCrawled`	The number of metadata files the were found. Dimensions: • IndexId • DataSourceId Unit: Count

Metrics for Indexed Documents

The following table describes the Amazon Kendra metrics for indexed documents. For documents that are indexed using the BatchPutDocument (p. 92) operation, only the `IndexId` dimension is supported.

Metric	Description
`DocumentsIndexed`	The number of documents indexed.

Metric	Description
	Dimensions: • IndexId • DataSourceId Unit: Count
`DocumentsFailedToIndex`	The number of documents that could not be indexed. Check the contents of the CloudWatch log for details. Dimensions: • IndexId • DataSourceId Unit: Count

Monitoring Amazon Kendra with Amazon CloudWatch Logs

Amazon Kendra uses Amazon CloudWatch Logs to give you insight into the operation of your data sources. Amazon Kendra logs process details for the documents that as they are indexed. It logs errors from your data source that occur while your documents are being indexed. You use CloudWatch Logs to monitor, store and access the log files.

CloudWatch Logs stores log events in a log stream that is part of a log group. Amazon Kendra uses these features as follows:

- Log groups – Amazon Kendra stores all of your log streams in a single log group for each index. Amazon Kendra creates the log group when the index is created. The log group identifier always begins with "aws/kendra/".
- Log stream – creates a new data source log stream in the log group for each index synchronization job that you run. It also creates a new document log stream when a stream reaches approximately 500 entries.
- Log entries – Amazon Kendra creates a log entry in the log stream as it indexes documents. Each entry provides information about processing the document or any errors that are encountered.

For more information about using CloudWatch Logs, see What Is Amazon Cloud Watch Logs in the *Amazon Cloud Watch Logs User Guide*.

Amazon Kendra creates two types of log streams:

- Data Source Log Streams (p. 82)
- Document Log Streams (p. 83)

Data Source Log Streams

Data source log streams publish entries about your index synchronization jobs. Each synchronization job creates a new log stream that it uses to publish entries. The log stream name is:

```
data source id/YYYY-MM-DD-HH/data source sync job ID
```

A new log stream is created for each synchronization job run.

There are three types of log messages published to a data source log stream:

- A log message for a document that failed to be sent for indexing. The following is an example of this message for a document in an S3 data source:

```
{
    "DocumentId": "document ID",
    "S3Path": "s3://bucket/prefix/object",
    "Message": "Failed to ingest document via BatchPutDocument.",
    "ErrorCode": "InvalidRequest",
    "ErrorMessage": "No document metadata configuration found for document attribute key
 city."
}
```

- A log message for a document that failed to be sent for deletion. The following is an example of this message:

```
{
    "DocumentId": "document ID",
    "Message": "Failed to delete document via BatchDeleteDocument.",
    "ErrorCode": "InvalidRequest",
    "ErrorMessage": "Document can't be deleted because it doesn't exist."
}
```

- A log message when an invalid metadata file for a document in an Amazon S3 bucket is found. The following is an example of this message.

```
{
    "Message": "Found invalid metadata
 file bucket/prefix/filename.extension.metadata.json."
}
```

- For SharePoint and database connectors, Amazon Kendra only writes messages to the log stream if a document can't be indexed. The following is an example of the error message that Amazon Kendra logs.

```
{
    "DocumentID": "document ID",
    "IndexID": "index ID",
    "SourceURI": "",
    "CrawlStatus": "FAILED",
    "ErrorCode": "403",
    "ErrorMessage": "Access Denied",
    "DataSourceErrorCode": "403"
}
```

Document Log Streams

Amazon Kendra logs information about processing documents while the are being indexed. It logs a set of messages for documents stored in an Amazon S3 data source. It logs errors only for documents stored in a Microsoft SharePoint or a database data source.

If the documents were added to the index using the BatchPutDocument (p. 92) operation, the log stream is named as follows:

```
YYYY-MM-DD-HH/UUID
```

If the documents were added to the index using a datasource, the log stream is named as follows:

```
dataSourceId/YYYY-MM-DD-HH/UUID
```

Each log stream contains up to 500 messages.

If indexing a document fails, this message is output to the log stream:

```
{
    "DocumentId": "document ID",
    "IndexName": "index name",
    "IndexId": "index ID"
    "SourceURI": "source URI"
    "IndexingStatus": "DocumentFailedToIndex",
    "ErrorCode": "400 | 500",
    "ErrorMessage": "message"
}
```

Deploying Amazon Kendra

When it comes time to deploy Amazon Kendra search to your Web site, we provide source code that you can use to get a head start on your application. The source code is provided free of charge under a modified MIT license so that you can use it as is or change it for your own needs.

The example application is modeled on the search page of the Amazon Kendra console. It has all of the same features for searching and displaying search results. You can use the whole example application, or you can choose just one of the features for your own use.

To see the three components of the search page, choose the code icon (**</>**) from the right menu. Hover your mouse over each section to see a brief description of the component and to get the URL of the component source.

Topics

Overview

You add the example code to an existing React application to enable search. The search files and components are structured as follows:

- Main search page – this is the main page that contains all of the components. This is where you will integrate your application with the Amazon Kendra API.
- Search bar – this is the component where a user enters a search term and that calls the search function.
- Results – this is the component that displays the results from Amazon Kendra. It has three components: Suggested answers, FAQ results, and recommended documents.
- Pagination – this is the component that paginates the response from Amazon Kendra.

Prerequisites

Before you begin you need the following:

- An existing React Web application.
- A development environment with the following libraries:
  - React version 16.8.10
  - Bootstrap version 4.3.1
  - Typescript version 3.2.1
  - @types/lodash

- The AWS SDK for JavaScript library. For information about installing the AWS SDK, see AWS SDK for JavaScript on GitHub.

Setting Up The Example

To install the example application

1. Download the source files from the repository. https://kendrasamples.s3.amazonaws.com/kendrasamples.zip
2. Create a new Web page for search, or you can choose an existing page to add the search bar to.
3. Add the Search.tsx component to the page.
4. Build the change into your package.

Main Search Page

The main search page contains all of the example search components. It includes the search bar component for output, the results components to display the response from the Query (p. 133) operation, and a pagination component that enables you to page through the response.

- File path: `search/Search.tsx`

Search Component

The search component provides a text box to enter query text. The `onSearch` function is a hook that calls the main function in `Search.tsx` to make the Amazon Kendra Query (p. 133) operation call.

- File path: `search/searchBar/SearchBar.tsx`

Results Component

The results component shows the response from the `Query` operation. The results are shown in three separate areas.

- Suggested answers – These are the top results returned by the `Query` operation. It contains up to three suggested answers. In the response, they have the result type `ANSWER`.
- FAQ answers – These are the frequently asked questions results returned by the response. FAQs are added to the index separately. In the response, they have the type `QUESTION_ANSWER`. For more information, see Adding Questions and Answers (p. 44).
- Recommended documents – These are additional documents that Amazon Kendra returns in the response. In the response from the `Query` operation, they have the type `DOCUMENT`.

The results component uses the following source files:

- `search/resultsPanel/ResultPanel.tsx`
- `search/resultsPanel/TopResults.tsx`
- `search/resultsPanel/FAQResults.tsx`
- `search/resultsPanel/DocumentResults.tsx`

The results components share a set of components for features like highlighting, titles, links, etc. The shared components must be present for the result components to work. They are located at the following path:

- `search/resultsPanel/components`

Pagination Component

The pagination control enables you to display the search results from the `Query` operation in multiple pages. It calls the `Query` operation with the `PageSize` and `PageNumber` parameters to get a specific page of results.

- `search/pagination/Pagination.tsx`

Quotas for Amazon Kendra

Supported Regions

For a list of AWS Regions where Amazon Kendra is available, see AWS Regions and Endpoints in the *Amazon Web Services General Reference*.

Quotas

Service quotas, also referred to as limits, are the maximum number of service resources for your AWS account. For more information, see AWS Service Quotas in the *AWS General Reference*.

During the preview release, Amazon Kendra has the following limits:

Description	Default
Maximum number of indexes	1
Maximum number of data sources per index	3
Maximum number of documents per index	100,000
Maximum size of a single document	50 MB
Maximum amount of text extracted from a document	500 KB

You can use Amazon CloudWatch to determine if your index or documents have exceeded these quotas. For more information, see General Troubleshooting (p. 59).

Document History for Amazon Kendra

- **Latest documentation update:** November 21, 2019

The following table describes important changes in each release of Amazon Kendra. For notification about updates to this documentation, you can subscribe to an RSS feed.

update-history-change	update-history-description	update-history-date
Preview release	This is the preview release of the *Amazon Kendra Developer Guide*.	December 3, 2019

API Reference

This section contains the API Reference documentation.

Actions

The following actions are supported:

BatchDeleteDocument

Removes one or more documents from an index. The documents must have been added with the BatchPutDocument (p. 92) operation.

The documents are deleted asynchronously. You can see the progress of the deletion by using AWS CloudWatch. Any error messages releated to the processing of the batch are sent to you CloudWatch log.

Request Syntax

```
{
   "DocumentIdList": [ "string" ],
   "IndexId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

DocumentIdList (p. 90)

One or more identifiers for documents to delete from the index.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

IndexId (p. 90)

The identifier of the index that contains the documents to delete.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Response Syntax

```
{
   "FailedDocuments": [
      {
         "ErrorCode": "string",
         "ErrorMessage": "string",
         "Id": "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.

FailedDocuments (p. 90)

A list of documents that could not be removed from the index. Each entry contains an error message that indicates why the document couldn't be removed from the index.

Type: Array of BatchDeleteDocumentResponseFailedDocument (p. 159) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

BatchPutDocument

Adds one or more documents to an index.

The `BatchPutDocument` operation enables you to ingest inline documents or a set of documents stored in an Amazon S3 bucket. Use this operation to ingest your text and unstructured text into an index, add custom attributes to the documents, and to attach an access control list to the documents added to the index.

The documents are indexed asynchronously. You can see the progress of the batch using AWS CloudWatch. Any error messages related to processing the batch are sent to your AWS CloudWatch log.

Request Syntax

```
{
   "Documents": [
      {
         "AccessControlList": [
            {
               "Access": "string",
               "Name": "string",
               "Type": "string"
            }
         ],
         "Attributes": [
            {
               "Key": "string",
               "Value": {
                  "DateValue": number,
                  "LongValue": number,
                  "StringListValue": [ "string" ],
                  "StringValue": "string"
               }
            }
         ],
         "Blob": blob,
         "ContentType": "string",
         "Id": "string",
         "S3Path": {
            "Bucket": "string",
            "Key": "string"
         },
         "Title": "string"
      }
   ],
   "IndexId": "string",
   "RoleArn": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Documents (p. 92)

One or more documents to add to the index.

Each document is limited to 5 Mb, the total size of the list is limited to 50 Mb.

Type: Array of Document (p. 174) objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

IndexId (p. 92)

The identifier of the index to add the documents to. You need to create the index first using the CreateIndex (p. 102) operation.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

RoleArn (p. 92)

The Amazon Resource Name (ARN) of a role that is allowed to run the `BatchPutDocument` operation. For more information, see IAM Roles for Amazon Kendra.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Required: No

Response Syntax

```
{
   "FailedDocuments": [
      {
         "ErrorCode": "string",
         "ErrorMessage": "string",
         "Id": "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.

FailedDocuments (p. 93)

A list of documents that were not added to the index because the document failed a validation check. Each document contains an error message that indicates why the document couldn't be added to the index.

If there was an error adding a document to an index the error is reported in your AWS CloudWatch log.

Type: Array of BatchPutDocumentResponseFailedDocument (p. 160) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ServiceQuotaExceededException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateDataSource

Creates a data source that you use to with an Amazon Kendra index.

You specify a name, connector type and description for your data source. You can choose between an S3 connector, a SharePoint Online connector, and a database connector.

You also specify configuration information such as document metadata (author, source URI, and so on) and user context information.

`CreateDataSource` is a synchronous operation. The operation returns 200 if the data source was successfully created. Otherwise, an exception is raised.

Request Syntax

```
{
   "Configuration": {
      "DatabaseConfiguration": {
         "AclConfiguration": {
            "AllowedGroupsColumnName": "string"
         },
         "ColumnConfiguration": {
            "ChangeDetectingColumns": [ "string" ],
            "DocumentDataColumnName": "string",
            "DocumentIdColumnName": "string",
            "DocumentTitleColumnName": "string",
            "FieldMappings": [
               {
                  "DataSourceFieldName": "string",
                  "DateFieldFormat": "string",
                  "IndexFieldName": "string"
               }
            ]
         },
         "ConnectionConfiguration": {
            "DatabaseHost": "string",
            "DatabaseName": "string",
            "DatabasePort": number,
            "SecretArn": "string",
            "TableName": "string"
         },
         "DatabaseEngineType": "string",
         "VpcConfiguration": {
            "SecurityGroupIds": [ "string" ],
            "SubnetIds": [ "string" ]
         }
      },
      "S3Configuration": {
         "AccessControlListConfiguration": {
            "KeyPath": "string"
         },
         "BucketName": "string",
         "DocumentsMetadataConfiguration": {
            "S3Prefix": "string"
         },
         "ExclusionPatterns": [ "string" ],
         "InclusionPrefixes": [ "string" ]
      },
      "SharePointConfiguration": {
         "CrawlAttachments": boolean,
         "DocumentTitleFieldName": "string",
         "FieldMappings": [
            {
```

```
                "DataSourceFieldName": "string",
                "DateFieldFormat": "string",
                "IndexFieldName": "string"
             }
          ],
          "SecretArn": "string",
          "SharePointVersion": "string",
          "Urls": [ "string" ],
          "VpcConfiguration": {
             "SecurityGroupIds": [ "string" ],
             "SubnetIds": [ "string" ]
          }
       }
    },
    "Description": "string",
    "IndexId": "string",
    "Name": "string",
    "RoleArn": "string",
    "Schedule": "string",
    "Type": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Configuration (p. 95)

The connector configuration information that is required to access the repository.

Type: DataSourceConfiguration (p. 167) object

Required: Yes

Description (p. 95)

A description for the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

Required: No

IndexId (p. 95)

The identifier of the index that should be associated with this data source.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Name (p. 95)

A unique name for the data source. A data source name can't be changed without deleting and recreating the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

RoleArn (p. 95)

The Amazon Resource Name (ARN) of a role with permission to access the data source. For more information, see IAM Roles for Amazon Kendra.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Required: Yes

Schedule (p. 95)

Sets the frequency that Amazon Kendra will check the documents in your repository and update the index. If you don't set a schedule Amazon Kendra will not periodically update the index. You can call the `StartDataSourceSyncJob` operation to update the index.

Type: String

Required: No

Type (p. 95)

The type of repository that contains the data source.

Type: String

Valid Values: `S3 | SHAREPOINT | DATABASE`

Required: Yes

Response Syntax

```
{
   "Id": "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.

Id (p. 97)

A unique identifier for the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceAlreadyExistException

HTTP Status Code: 400

ResourceNotFoundException

HTTP Status Code: 400

ServiceQuotaExceededException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateFaq

Creates an new set of frequently asked question (FAQ) questions and answers.

Request Syntax

```
{
   "Description": "string",
   "IndexId": "string",
   "Name": "string",
   "RoleArn": "string",
   "S3Path": {
      "Bucket": "string",
      "Key": "string"
   }
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Description (p. 99)

A description of the FAQ.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

Required: No

IndexId (p. 99)

The identifier of the index that contains the FAQ.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Name (p. 99)

The name that should be associated with the FAQ.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

RoleArn (p. 99)

The Amazon Resource Name (ARN) of a role with permission to access the S3 bucket that contains the FAQs. For more information, see IAM Roles for Amazon Kendra.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Required: Yes

S3Path (p. 99)

The S3 location of the FAQ input data.

Type: S3Path (p. 198) object

Required: Yes

Response Syntax

```
{
   "Id": "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.

Id (p. 100)

The unique identifier of the FAQ.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ServiceQuotaExceededException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateIndex

Creates a new Amazon Kendra index. Index creation is an asynchronous operation. To determine if index creation has completed, check the `Status` field returned from a call to DescribeIndex (p. 118). The `Status` field is set to `ACTIVE` when the index is ready to use.

Once the index is active you can index your documents using the BatchPutDocument (p. 92) operation or using one of the supported data sources.

Request Syntax

```
{
   "Description": "string",
   "Name": "string",
   "RoleArn": "string",
   "ServerSideEncryptionConfiguration": {
      "KmsKeyId": "string"
   }
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Description (p. 102)

A description for the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

Required: No

Name (p. 102)

The name for the new index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

RoleArn (p. 102)

An IAM role that gives Amazon Kendra permissions to access your Amazon CloudWatch logs and metrics. This is also the role used when you use the `BatchPutDocument` operation to index documents from an Amazon S3 bucket.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Required: Yes

ServerSideEncryptionConfiguration (p. 102)

The identifier of the AWS KMS customer managed key (CMK) to use to encrypt data indexed by Amazon Kendra. Amazon Kendra doesn't support asymmetric CMKs.

Type: ServerSideEncryptionConfiguration (p. 200) object

Required: No

Response Syntax

```
{
   "Id": "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.

Id (p. 103)

The unique identifier of the index. Use this identifier when you query an index, set up a data source, or index a document.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceAlreadyExistException

HTTP Status Code: 400

ServiceQuotaExceededException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteFaq

Removes an FAQ from an index.

Request Syntax

```
{
   "Id": "string",
   "IndexId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Id (p. 105)

The identifier of the FAQ to remove.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

IndexId (p. 105)

The index to remove the FAQ from.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[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. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteIndex

Deletes an existing Amazon Kendra index. An exception is not thrown if the index is already being deleted. While the index is being deleted, the `Status` field returned by a call to the DescribeIndex (p. 118) operation is set to `DELETING`.

Request Syntax

```
{
   "Id": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Id (p. 107)

The identifier of the index to delete.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[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. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeDataSource

Gets information about a Amazon Kendra data source.

Request Syntax

```
{
   "Id": "string",
   "IndexId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Id (p. 109)

The unique identifier of the data source to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

IndexId (p. 109)

The identifier of the index that contains the data source.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Response Syntax

```
{
   "Configuration": {
      "DatabaseConfiguration": {
         "AclConfiguration": {
            "AllowedGroupsColumnName": "string"
         },
         "ColumnConfiguration": {
            "ChangeDetectingColumns": [ "string" ],
            "DocumentDataColumnName": "string",
            "DocumentIdColumnName": "string",
            "DocumentTitleColumnName": "string",
            "FieldMappings": [
               {
                  "DataSourceFieldName": "string",
                  "DateFieldFormat": "string",
                  "IndexFieldName": "string"
```

```
                }
            ]
         },
         "ConnectionConfiguration": {
            "DatabaseHost": "string",
            "DatabaseName": "string",
            "DatabasePort": number,
            "SecretArn": "string",
            "TableName": "string"
         },
         "DatabaseEngineType": "string",
         "VpcConfiguration": {
            "SecurityGroupIds": [ "string" ],
            "SubnetIds": [ "string" ]
         }
      },
      "S3Configuration": {
         "AccessControlListConfiguration": {
            "KeyPath": "string"
         },
         "BucketName": "string",
         "DocumentsMetadataConfiguration": {
            "S3Prefix": "string"
         },
         "ExclusionPatterns": [ "string" ],
         "InclusionPrefixes": [ "string" ]
      },
      "SharePointConfiguration": {
         "CrawlAttachments": boolean,
         "DocumentTitleFieldName": "string",
         "FieldMappings": [
            {
               "DataSourceFieldName": "string",
               "DateFieldFormat": "string",
               "IndexFieldName": "string"
            }
         ],
         "SecretArn": "string",
         "SharePointVersion": "string",
         "Urls": [ "string" ],
         "VpcConfiguration": {
            "SecurityGroupIds": [ "string" ],
            "SubnetIds": [ "string" ]
         }
      }
   },
   "CreatedAt": number,
   "Description": "string",
   "ErrorMessage": "string",
   "Id": "string",
   "IndexId": "string",
   "Name": "string",
   "RoleArn": "string",
   "Schedule": "string",
   "Status": "string",
   "Type": "string",
   "UpdatedAt": 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.

Configuration (p. 109)

Information that describes where the data source is located and how the data source is configured. The specific information in the description depends on the data source provider.

Type: DataSourceConfiguration (p. 167) object

CreatedAt (p. 109)

The Unix timestamp of when the data source was created.

Type: Timestamp

Description (p. 109)

The description of the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

ErrorMessage (p. 109)

When the `Status` field value is `FAILED`, the `ErrorMessage` field contains a description of the error that caused the data source to fail.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^\P{C}*$`

Id (p. 109)

The identifier of the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

IndexId (p. 109)

The identifier of the index that contains the data source.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Name (p. 109)

The name that you gave the data source when it was created.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

RoleArn (p. 109)

The Amazon Resource Name (ARN) of the role that enables the data source to access its resources.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Schedule (p. 109)

The schedule that Amazon Kendra will update the data source.

Type: String

Status (p. 109)

The current status of the data source. When the status is `ACTIVE` the data source is ready to use. When the status is `FAILED`, the `ErrorMessage` field contains the reason that the data source failed.

Type: String

Valid Values: `CREATING | DELETING | FAILED | UPDATING | ACTIVE`

Type (p. 109)

The type of the data source.

Type: String

Valid Values: `S3 | SHAREPOINT | DATABASE`

UpdatedAt (p. 109)

The Unix timestamp of when the data source was last updated.

Type: Timestamp

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeFaq

Gets information about an FAQ list.

Request Syntax

```
{
   "Id": "string",
   "IndexId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Id (p. 114)

The unique identifier of the FAQ.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

IndexId (p. 114)

The identifier of the index that contains the FAQ.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Response Syntax

```
{
   "CreatedAt": number,
   "Description": "string",
   "ErrorMessage": "string",
   "Id": "string",
   "IndexId": "string",
   "Name": "string",
   "RoleArn": "string",
   "S3Path": {
      "Bucket": "string",
      "Key": "string"
   },
   "Status": "string",
   "UpdatedAt": 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.

CreatedAt (p. 114)

The date and time that the FAQ was created.

Type: Timestamp

Description (p. 114)

The description of the FAQ that you provided when it was created.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

ErrorMessage (p. 114)

If the `Status` field is `FAILED`, the `ErrorMessage` field contains the reason why the FAQ failed.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^\P{C}*$`

Id (p. 114)

The identifier of the FAQ.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

IndexId (p. 114)

The identifier of the index that contains the FAQ.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Name (p. 114)

The name that you gave the FAQ when it was created.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

RoleArn (p. 114)

The Amazon Resource Name (ARN) of the role that provides access to the S3 bucket containing the input files for the FAQ.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

S3Path (p. 114)

Information required to find a specific file in an Amazon S3 bucket.

Type: S3Path (p. 198) object

Status (p. 114)

The status of the FAQ. It is ready to use when the status is `ACTIVE`.

Type: String

Valid Values: `CREATING | UPDATING | ACTIVE | DELETING | FAILED`

UpdatedAt (p. 114)

The date and time that the FAQ was last updated.

Type: Timestamp

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeIndex

Describes an existing Amazon Kendra index

Request Syntax

```
{
   "Id": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Id (p. 118)

The name of the index to describe.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Response Syntax

```
{
   "CreatedAt": number,
   "Description": "string",
   "DocumentMetadataConfigurations": [
      {
         "Name": "string",
         "Relevance": {
            "Duration": "string",
            "Freshness": boolean,
            "Importance": number,
            "RankOrder": "string",
            "ValueImportanceMap": {
               "string" : number
            }
         },
         "Search": {
            "Displayable": boolean,
            "Facetable": boolean,
            "Searchable": boolean
         },
         "Type": "string"
      }
   ],
   "ErrorMessage": "string",
   "Id": "string",
   "IndexStatistics": {
      "FaqStatistics": {
```

```
            "IndexedQuestionAnswersCount": number
        },
        "TextDocumentStatistics": {
            "IndexedTextDocumentsCount": number
        }
    },
    "Name": "string",
    "RoleArn": "string",
    "ServerSideEncryptionConfiguration": {
        "KmsKeyId": "string"
    },
    "Status": "string",
    "UpdatedAt": 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.

CreatedAt (p. 118)

The Unix datetime that the index was created.

Type: Timestamp

Description (p. 118)

The description of the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

DocumentMetadataConfigurations (p. 118)

Configuration settings for any metadata applied to the documents in the index.

Type: Array of DocumentMetadataConfiguration (p. 179) objects

Array Members: Minimum number of 0 items. Maximum number of 500 items.

ErrorMessage (p. 118)

When th e`Status` field value is `FAILED`, the `ErrorMessage` field contains a message that explains why.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^\P{C}*$`

Id (p. 118)

the name of the index.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

IndexStatistics (p. 118)

Provides information about the number of FAQ questions and answers and the number of text documents indexed.

Type: IndexStatistics (p. 189) object

Name (p. 118)

The name of the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

RoleArn (p. 118)

The Amazon Resource Name (ARN) of the IAM role that gives Amazon Kendra permission to write to your Amazon Cloudwatch logs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

ServerSideEncryptionConfiguration (p. 118)

The identifier of the AWS KMS customer master key (CMK) used to encrypt your data. Amazon Kendra doesn't support asymmetric CMKs.

Type: ServerSideEncryptionConfiguration (p. 200) object

Status (p. 118)

The current status of the index. When the value is `ACTIVE`, the index is ready for use. If the `Status` field value is `FAILED`, the `ErrorMessage` field contains a message that explains why.

Type: String

Valid Values: `CREATING | ACTIVE | DELETING | FAILED | SYSTEM_UPDATING`

UpdatedAt (p. 118)

The Unix datetime that the index was last updated.

Type: Timestamp

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListDataSources

Lists the data sources that you have created.

Request Syntax

```
{
   "IndexId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

IndexId (p. 122)

The identifier of the index that contains the data source.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

MaxResults (p. 122)

The maximum number of data sources to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

NextToken (p. 122)

If the previous response was incomplete (because there is more data to retrieve), Amazon Kendra returns a pagination token in the response. You can use this pagination token to retrieve the next set of data sources (`DataSourceSummaryItems`).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 800.

Required: No

Response Syntax

```
{
   "NextToken": "string",
   "SummaryItems": [
      {
```

```
            "CreatedAt": number,
            "Id": "string",
            "Name": "string",
            "Status": "string",
            "Type": "string",
            "UpdatedAt": 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. 122)

If the response is truncated, Amazon Kendra returns this token that you can use in the subsequent request to retrieve the next set of data sources.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 800.

SummaryItems (p. 122)

An array of summary information for one or more data sources.

Type: Array of DataSourceSummary (p. 168) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListDataSourceSyncJobs

Gets statistics about synchronizing Amazon Kendra with a data source.

Request Syntax

```
{
   "Id": "string",
   "IndexId": "string",
   "MaxResults": number,
   "NextToken": "string",
   "StartTimeFilter": {
      "EndTime": number,
      "StartTime": number
   },
   "StatusFilter": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Id (p. 125)

The identifier of the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

IndexId (p. 125)

The identifier of the index that contains the data source.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

MaxResults (p. 125)

The maximum number of synchronization jobs to return in the response. If there are fewer results in the list, this response contains only the actual results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 10.

Required: No

NextToken (p. 125)

If the result of the previous request to `GetDataSourceSyncJobHistory` was truncated, include the `NextToken` to fetch the next set of jobs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 800.

Required: No

StartTimeFilter (p. 125)

When specified, the synchronization jobs returned in the list are limited to jobs between the specified dates.

Type: TimeRange (p. 205) object

Required: No

StatusFilter (p. 125)

When specified, only returns synchronization jobs with the `Status` field equal to the specified status.

Type: String

Valid Values: `FAILED | SUCCEEDED | SYNCING | INCOMPLETE | STOPPING | ABORTED`

Required: No

Response Syntax

```
{
   "History": [
      {
         "DataSourceErrorCode": "string",
         "EndTime": number,
         "ErrorCode": "string",
         "ErrorMessage": "string",
         "ExecutionId": "string",
         "StartTime": number,
         "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.

History (p. 126)

A history of synchronization jobs for the data source.

Type: Array of DataSourceSyncJob (p. 170) objects

NextToken (p. 126)

The `GetDataSourceSyncJobHistory` operation returns a page of vocabularies at a time. The maximum size of the page is set by the `MaxResults` parameter. If there are more jobs in the list than the page size, Amazon Kendra returns the NextPage token. Include the token in the next request to the `GetDataSourceSyncJobHistory` operation to return in the next page of jobs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 800.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListFaqs

Gets a list of FAQ lists associated with an index.

Request Syntax

```
{
   "IndexId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

IndexId (p. 128)

The index that contains the FAQ lists.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

MaxResults (p. 128)

The maximum number of FAQs to return in the response. If there are fewer results in the list, this response contains only the actual results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

NextToken (p. 128)

If the result of the previous request to `ListFaqs` was truncated, include the `NextToken` to fetch the next set of FAQs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 800.

Required: No

Response Syntax

```
{
   "FaqSummaryItems": [
      {
         "CreatedAt": number,
```

```
            "Id": "string",
            "Name": "string",
            "Status": "string",
            "UpdatedAt": 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.

FaqSummaryItems (p. 128)

information about the FAQs associated with the specified index.

Type: Array of FaqSummary (p. 184) objects

NextToken (p. 128)

The `ListFaqs` operation returns a page of FAQs at a time. The maximum size of the page is set by the `MaxResults` parameter. If there are more jobs in the list than the page size, Amazon Kendra returns the `NextPage` token. Include the token in the next request to the `ListFaqs` operation to return the next page of FAQs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 800.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListIndices

Lists the Amazon Kendra indexes 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. 207).

The request accepts the following data in JSON format.

MaxResults (p. 131)

The maximum number of data sources to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

NextToken (p. 131)

If the previous response was incomplete (because there is more data to retrieve), Amazon Kendra returns a pagination token in the response. You can use this pagination token to retrieve the next set of indexes (`DataSourceSummaryItems`).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 800.

Required: No

Response Syntax

```
{
   "IndexConfigurationSummaryItems": [
      {
         "CreatedAt": number,
         "Id": "string",
         "Name": "string",
         "Status": "string",
         "UpdatedAt": 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.

IndexConfigurationSummaryItems (p. 131)

An array of summary information for one or more indexes.

Type: Array of IndexConfigurationSummary (p. 187) objects

NextToken (p. 131)

If the response is truncated, Amazon Kendra returns this token that you can use in the subsequent request to retrieve the next set of indexes.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 800.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Query

Searches an active index. Use this API to search your documents using query. The `Query` operation enables to do faceted search and to filter results based on document attributes.

It also enables you to provide user context that Amazon Kendra uses to enforce document access control in the search results.

Amazon Kendra searches your index for text content and question and answer (FAQ) content. By default the response contains three types of results.

- Relevant passages
- Matching FAQs
- Relevant documents

You can specify that the query return only one type of result using the `QueryResultTypeConfig` parameter.

Request Syntax

```
{
   "AttributeFilter": {
      "AndAllFilters": [
         "AttributeFilter"
      ],
      "ContainsAll": {
         "Key": "string",
         "Value": {
            "DateValue": number,
            "LongValue": number,
            "StringListValue": [ "string" ],
            "StringValue": "string"
         }
      },
      "ContainsAny": {
         "Key": "string",
         "Value": {
            "DateValue": number,
            "LongValue": number,
            "StringListValue": [ "string" ],
            "StringValue": "string"
         }
      },
      "EqualsTo": {
         "Key": "string",
         "Value": {
            "DateValue": number,
            "LongValue": number,
            "StringListValue": [ "string" ],
            "StringValue": "string"
         }
      },
      "GreaterThan": {
         "Key": "string",
         "Value": {
            "DateValue": number,
            "LongValue": number,
            "StringListValue": [ "string" ],
            "StringValue": "string"
         }
```

```
      },
      "GreaterThanOrEquals": {
         "Key": "string",
         "Value": {
            "DateValue": number,
            "LongValue": number,
            "StringListValue": [ "string" ],
            "StringValue": "string"
         }
      },
      "LessThan": {
         "Key": "string",
         "Value": {
            "DateValue": number,
            "LongValue": number,
            "StringListValue": [ "string" ],
            "StringValue": "string"
         }
      },
      "LessThanOrEquals": {
         "Key": "string",
         "Value": {
            "DateValue": number,
            "LongValue": number,
            "StringListValue": [ "string" ],
            "StringValue": "string"
         }
      },
      "NotFilter": "AttributeFilter",
      "OrAllFilters": [
         "AttributeFilter"
      ]
   },
   "Facets": [
      {
         "DocumentAttributeKey": "string"
      }
   ],
   "IndexId": "string",
   "PageNumber": number,
   "PageSize": number,
   "QueryResultTypeFilter": "string",
   "QueryText": "string",
   "RequestedDocumentAttributes": [ "string" ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

AttributeFilter (p. 133)

Enables filtered searches based on document attributes. You can only provide one attribute filter; however, the `AndAllFilters`, `NotFilter`, and `OrAllFilters` parameters contain a list of other filters.

The `AttributeFilter` parameter enables you to create a set of filtering rules that a document must satisfy to be included in the query results.

Type: AttributeFilter (p. 157) object

Required: No

Facets (p. 133)

An array of documents attributes. Amazon Kendra returns a count for each attribute key specified. You can use this information to help narrow the search for your user.

Type: Array of Facet (p. 181) objects

Required: No

IndexId (p. 133)

The unique identifier of the index to search. The identifier is returned in the response from the CreateIndex (p. 102) operation.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

PageNumber (p. 133)

Query results are returned in pages the size of the `PageSize` parameter. By default, Amazon Kendra returns the first page of results. Use this parameter to get result pages after the first one.

Type: Integer

Required: No

PageSize (p. 133)

Sets the number of results that are returned in each page of results. The default page size is 100.

Type: Integer

Required: No

QueryResultTypeFilter (p. 133)

Sets the type of query. Only results for the specified query type are returned.

Type: String

Valid Values: `DOCUMENT | QUESTION_ANSWER | ANSWER`

Required: No

QueryText (p. 133)

The text to search for.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

Required: Yes

RequestedDocumentAttributes (p. 133)

An array of document attributes to include in the response. No other document attributes are included in the response. By default all document attributes are included in the response.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[a-zA-Z0-9_][a-zA-Z0-9_-]*`

Required: No

Response Syntax

```
{
   "FacetResults": [
      {
         "DocumentAttributeKey": "string",
         "DocumentAttributeValueCountPairs": [
            {
               "Count": number,
               "DocumentAttributeValue": {
                  "DateValue": number,
                  "LongValue": number,
                  "StringListValue": [ "string" ],
                  "StringValue": "string"
               }
            }
         ]
      }
   ],
   "QueryId": "string",
   "ResultItems": [
      {
         "AdditionalAttributes": [
            {
               "Key": "string",
               "Value": {
                  "TextWithHighlightsValue": {
                     "Highlights": [
                        {
                           "BeginOffset": number,
                           "EndOffset": number,
                           "TopAnswer": boolean
                        }
                     ],
                     "Text": "string"
                  }
               },
               "ValueType": "string"
            }
         ],
         "DocumentAttributes": [
            {
               "Key": "string",
               "Value": {
                  "DateValue": number,
                  "LongValue": number,
                  "StringListValue": [ "string" ],
                  "StringValue": "string"
               }
            }
         ],
         "DocumentExcerpt": {
            "Highlights": [
               {
                  "BeginOffset": number,
```

```
                "EndOffset": number,
                "TopAnswer": boolean
             }
          ],
          "Text": "string"
       },
       "DocumentId": "string",
       "DocumentTitle": {
          "Highlights": [
             {
                "BeginOffset": number,
                "EndOffset": number,
                "TopAnswer": boolean
             }
          ],
          "Text": "string"
       },
       "DocumentURI": "string",
       "Id": "string",
       "Type": "string"
    }
  ],
  "TotalNumberOfResults": 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.

FacetResults (p. 136)

Contains the facet results. A `FacetResult` contains the counts for each attribute key that was specified in the `Facets` input parameter.

Type: Array of FacetResult (p. 182) objects

QueryId (p. 136)

The unique identifier for the search. You use `QueryId` to identify the search when using the feedback API.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 36.

ResultItems (p. 136)

The results of the search.

Type: Array of QueryResultItem (p. 191) objects

TotalNumberOfResults (p. 136)

The number of items returned by the search. Use this to determine when you have requested the last set of results.

Type: Integer

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StartDataSourceSyncJob

Starts a synchronization job for a data source. If a synchronization job is already in progress, Amazon Kendra returns a `ResourceInUseException` exception.

Request Syntax

```
{
   "Id": "string",
   "IndexId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Id (p. 139)

The identifier of the data source to synchronize.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

IndexId (p. 139)

The identifier of the index that contains the data source.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Response Syntax

```
{
   "ExecutionId": "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.

ExecutionId (p. 139)

Identifies a particular synchronization job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceInUseException

HTTP Status Code: 400

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StopDataSourceSyncJob

Stops a running synchronization job. You can't stop a scheduled synchronization job.

Request Syntax

```
{
   "Id": "string",
   "IndexId": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Id (p. 141)

The identifier of the data source for which to stop the synchronization jobs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

IndexId (p. 141)

The identifier of the index that contains the data source.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[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. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

SubmitFeedback

Enables you to provide feedback to Amazon Kendra to improve the performance of the service.

Request Syntax

```
{
   "ClickFeedbackItems": [
      {
         "ClickTime": number,
         "ResultId": "string"
      }
   ],
   "IndexId": "string",
   "QueryId": "string",
   "RelevanceFeedbackItems": [
      {
         "RelevanceValue": "string",
         "ResultId": "string"
      }
   ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

ClickFeedbackItems (p. 143)

Tells Amazon Kendra that a particular search result link was chosen by the user.

Type: Array of ClickFeedback (p. 161) objects

Required: No

IndexId (p. 143)

The identifier of the index that was queried.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

QueryId (p. 143)

The identifier of the specific query for which you are submitting feedback. The query ID is returned in the response to the Query (p. 133) operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 36.

Required: Yes

RelevanceFeedbackItems (p. 143)

Provides Amazon Kendra with relevant or not relevant feedback for whether a particular item was relevant to the search.

Type: Array of RelevanceFeedback (p. 195) objects

Required: No

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. 205).

AccessDeniedException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ResourceUnavailableException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateDataSource

Updates an existing Amazon Kendra data source.

Request Syntax

```
{
   "Configuration": {
      "DatabaseConfiguration": {
         "AclConfiguration": {
            "AllowedGroupsColumnName": "string"
         },
         "ColumnConfiguration": {
            "ChangeDetectingColumns": [ "string" ],
            "DocumentDataColumnName": "string",
            "DocumentIdColumnName": "string",
            "DocumentTitleColumnName": "string",
            "FieldMappings": [
               {
                  "DataSourceFieldName": "string",
                  "DateFieldFormat": "string",
                  "IndexFieldName": "string"
               }
            ]
         },
         "ConnectionConfiguration": {
            "DatabaseHost": "string",
            "DatabaseName": "string",
            "DatabasePort": number,
            "SecretArn": "string",
            "TableName": "string"
         },
         "DatabaseEngineType": "string",
         "VpcConfiguration": {
            "SecurityGroupIds": [ "string" ],
            "SubnetIds": [ "string" ]
         }
      },
      "S3Configuration": {
         "AccessControlListConfiguration": {
            "KeyPath": "string"
         },
         "BucketName": "string",
         "DocumentsMetadataConfiguration": {
            "S3Prefix": "string"
         },
         "ExclusionPatterns": [ "string" ],
         "InclusionPrefixes": [ "string" ]
      },
      "SharePointConfiguration": {
         "CrawlAttachments": boolean,
         "DocumentTitleFieldName": "string",
         "FieldMappings": [
            {
               "DataSourceFieldName": "string",
               "DateFieldFormat": "string",
               "IndexFieldName": "string"
            }
         ],
         "SecretArn": "string",
         "SharePointVersion": "string",
         "Urls": [ "string" ],
         "VpcConfiguration": {
```

```
            "SecurityGroupIds": [ "string" ],
            "SubnetIds": [ "string" ]
         }
      }
   },
   "Description": "string",
   "Id": "string",
   "IndexId": "string",
   "Name": "string",
   "RoleArn": "string",
   "Schedule": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Configuration (p. 145)

Configuration information for a Amazon Kendra data source.

Type: DataSourceConfiguration (p. 167) object

Required: No

Description (p. 145)

The new description for the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

Required: No

Id (p. 145)

The unique identifier of the data source to update.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: Yes

IndexId (p. 145)

The identifier of the index that contains the data source to update.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Name (p. 145)

The name of the data source to update. The name of the data source can't be updated. To rename a data source you must delete the data source and re-create it.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: No

RoleArn (p. 145)

The Amazon Resource Name (ARN) of the new role to use when the data source is accessing resources on your behalf.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Required: No

Schedule (p. 145)

The new update schedule for the data source.

Type: String

Required: No

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. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateIndex

Updates an existing Amazon Kendra index.

Request Syntax

```
{
   "Description": "string",
   "DocumentMetadataConfigurationUpdates": [
      {
         "Name": "string",
         "Relevance": {
            "Duration": "string",
            "Freshness": boolean,
            "Importance": number,
            "RankOrder": "string",
            "ValueImportanceMap": {
               "string" : number
            }
         },
         "Search": {
            "Displayable": boolean,
            "Facetable": boolean,
            "Searchable": boolean
         },
         "Type": "string"
      }
   ],
   "Id": "string",
   "Name": "string",
   "RoleArn": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 207).

The request accepts the following data in JSON format.

Description (p. 149)

A new description for the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `^\P{C}*$`

Required: No

DocumentMetadataConfigurationUpdates (p. 149)

The document metadata to update.

Type: Array of DocumentMetadataConfiguration (p. 179) objects

Array Members: Minimum number of 0 items. Maximum number of 500 items.

Required: No

Id (p. 149)

The identifier of the index to update.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: Yes

Name (p. 149)

The name of the index to update.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: No

RoleArn (p. 149)

A new IAM role that gives Amazon Kendra permission to access your Amazon CloudWatch logs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Required: No

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. 205).

AccessDeniedException

HTTP Status Code: 400

ConflictException

HTTP Status Code: 400

InternalServerException

HTTP Status Code: 500

ResourceNotFoundException

HTTP Status Code: 400

ThrottlingException

HTTP Status Code: 400

ValidationException

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
- 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:

AccessControlListConfiguration

Access Control List files for the documents in a data source.

Contents

KeyPath

Path to the AWS S3 bucket that contains the ACL files.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

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
- AWS SDK for Ruby V3

AclConfiguration

Provides information about the column that should be used for filtering the query response by groups.

Contents

AllowedGroupsColumnName

A list of groups, separated by semi-colons, that filters a query response based on user context. The document is only returned to users that are in one of the groups specified in the `UserContext` field of the Query (p. 133) operation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z][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
- AWS SDK for Ruby V3

AdditionalResultAttribute

Contents

Key

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

Value

Type: AdditionalResultAttributeValue (p. 156) object

Required: Yes

ValueType

Type: String

Valid Values: `TEXT_WITH_HIGHLIGHTS_VALUE`

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
- AWS SDK for Ruby V3

AdditionalResultAttributeValue

An attribute returned with a document from a search.

Contents

TextWithHighlightsValue

The text associated with the attribute and information about the highlight to apply to the text.

Type: TextWithHighlights (p. 204) 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
- AWS SDK for Ruby V3

AttributeFilter

Provides filtering the query results based on document attributes.

Contents

AndAllFilters

Performs a logical `AND` operation on all supplied filters.

Type: Array of AttributeFilter (p. 157) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: No

ContainsAll

Returns true when a document contains all of the specified document attributes.

Type: DocumentAttribute (p. 176) object

Required: No

ContainsAny

Returns true when a document contains any of the specified document attributes.

Type: DocumentAttribute (p. 176) object

Required: No

EqualsTo

Performs an equals operation on two document attributes.

Type: DocumentAttribute (p. 176) object

Required: No

GreaterThan

Performs a greater than operation on two document attributes. Use with a document attribute of type `Integer` or `Long`.

Type: DocumentAttribute (p. 176) object

Required: No

GreaterThanOrEquals

Performs a greater or equals than operation on two document attributes. Use with a document attribute of type `Integer` or `Long`.

Type: DocumentAttribute (p. 176) object

Required: No

LessThan

Performs a less than operation on two document attributes. Use with a document attribute of type `Integer` or `Long`.

Type: DocumentAttribute (p. 176) object

Required: No

LessThanOrEquals

Performs a less than or equals operation on two document attributes. Use with a document attribute of type `Integer` or `Long`.

Type: DocumentAttribute (p. 176) object

Required: No

NotFilter

Performs a logical `NOT` operation on all supplied filters.

Type: AttributeFilter (p. 157) object

Required: No

OrAllFilters

Performs a logical `OR` operation on all supplied filters.

Type: Array of AttributeFilter (p. 157) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

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
- AWS SDK for Ruby V3

BatchDeleteDocumentResponseFailedDocument

Provides information about documents that could not be removed from an index by the BatchDeleteDocument (p. 90) operation.

Contents

ErrorCode

The error code for why the document couldn't be removed from the index.

Type: String

Valid Values: `InternalError | InvalidRequest`

Required: No

ErrorMessage

An explanation for why the document couldn't be removed from the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^\P{C}*$`

Required: No

Id

The identifier of the document that couldn't be removed from the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

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
- AWS SDK for Ruby V3

BatchPutDocumentResponseFailedDocument

Provides information about a document that could not be indexed.

Contents

ErrorCode

The type of error that caused the document to fail to be indexed.

Type: String

Valid Values: `InternalError | InvalidRequest`

Required: No

ErrorMessage

A description of the reason why the document could not be indexed.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^\P{C}*$`

Required: No

Id

The unique identifier of the document.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

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
- AWS SDK for Ruby V3

ClickFeedback

Gathers information about when a particular result was clicked by a user. Your application uses the SubmitFeedback (p. 143) operation to provide click information.

Contents

ClickTime

The Unix timestamp of the data and time that the result was clicked.

Type: Timestamp

Required: Yes

ResultId

The unique identifier of the search result that was clicked.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 73.

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
- AWS SDK for Ruby V3

ColumnConfiguration

Provides information about how Amazon Kendra should use the columns of a database in an index.

Contents

ChangeDetectingColumns

One to five columns that indicate when a document in the database has changed.

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 100.

Pattern: `^[a-zA-Z][a-zA-Z0-9_]*$`

Required: Yes

DocumentDataColumnName

The column that contains the contents of the document.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z][a-zA-Z0-9_]*$`

Required: Yes

DocumentIdColumnName

The column that provides the document's unique identifier.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z][a-zA-Z0-9_]*$`

Required: Yes

DocumentTitleColumnName

The column that contains the title of the document.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z][a-zA-Z0-9_]*$`

Required: No

FieldMappings

An array of objects that map database column names to the corresponding fields in an index. You must first create the fields in the index using the UpdateIndex (p. 149) operation.

Type: Array of DataSourceToIndexFieldMapping (p. 172) objects

Array Members: Minimum number of 1 item. Maximum number of 100 items.

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
- AWS SDK for Ruby V3

ConnectionConfiguration

Provides the information necessary to connect to a database.

Contents

DatabaseHost

The name of the host for the database. Can be either a string (host.subdomain.domain.tld) or an IPv4 or IPv6 address.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 253.

Required: Yes

DatabaseName

The name of the database containing the document data.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z][a-zA-Z0-9_]*$`

Required: Yes

DatabasePort

The port that the database uses for connections.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 65535.

Required: Yes

SecretArn

The Amazon Resource Name (ARN) of credentials stored in AWS Secrets Manager. The credentials should be a user/password pair. For more information, see Using a Database Data Source. For more information about AWS Secrets Manager, see What Is AWS Secrets Manager in the *AWS Secrets Manager* user guide.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Required: Yes

TableName

The name of the table that contains the document data.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z][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
- AWS SDK for Ruby V3

DatabaseConfiguration

Provides the information necessary to connect a database to an index.

Contents

AclConfiguration

Information about the database column that provides information for user context filtering.

Type: AclConfiguration (p. 154) object

Required: No

ColumnConfiguration

Information about where the index should get the document information from the database.

Type: ColumnConfiguration (p. 162) object

Required: Yes

ConnectionConfiguration

The information necessary to connect to a database.

Type: ConnectionConfiguration (p. 164) object

Required: Yes

DatabaseEngineType

The type of database engine that runs the database.

Type: String

Valid Values: `RDS_AURORA_MYSQL | RDS_AURORA_POSTGRESQL | RDS_MYSQL | RDS_POSTGRESQL`

Required: Yes

VpcConfiguration

Provides information for connecting to an Amazon VPC.

Type: DataSourceVpcConfiguration (p. 173) 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
- AWS SDK for Ruby V3

DataSourceConfiguration

Configuration information for a Amazon Kendra data source.

Contents

DatabaseConfiguration

Provides information necessary to create a connector for a database.

Type: DatabaseConfiguration (p. 166) object

Required: No

S3Configuration

Provides information to create a connector for a document repository in an Amazon S3 bucket.

Type: S3DataSourceConfiguration (p. 196) object

Required: No

SharePointConfiguration

Provides information necessary to create a connector for a Microsoft SharePoint site.

Type: SharePointConfiguration (p. 201) 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
- AWS SDK for Ruby V3

DataSourceSummary

Summary information for a Amazon Kendra data source. Returned in a call to DescribeDataSource (p. 109).

Contents

CreatedAt

The UNIX datetime that the data source was created.

Type: Timestamp

Required: No

Id

The unique identifier for the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: No

Name

The name of the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: No

Status

The status of the data source. When the status is `ATIVE` the data source is ready to use.

Type: String

Valid Values: `CREATING | DELETING | FAILED | UPDATING | ACTIVE`

Required: No

Type

The type of the data source.

Type: String

Valid Values: `S3 | SHAREPOINT | DATABASE`

Required: No

UpdatedAt

The UNIX datetime that the data source was lasted updated.

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
- AWS SDK for Ruby V3

DataSourceSyncJob

Provides information about a synchronization job.

Contents

DataSourceErrorCode

If the reason that the synchronization failed is due to an error with the underlying data source, this field contains a code that identifies the error.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

EndTime

The UNIX datetime that the synchronization job was completed.

Type: Timestamp

Required: No

ErrorCode

If the `Status` field is set to `FAILED`, the `ErrorCode` field contains a the reason that the synchronization failed.

Type: String

Valid Values: `InternalError | InvalidRequest`

Required: No

ErrorMessage

If the `Status` field is set to `ERROR`, the `ErrorMessage` field contains a description of the error that caused the synchronization to fail.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^\P{C}*$`

Required: No

ExecutionId

A unique identifier for the synchronization job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

StartTime

The UNIX datetime that the synchronization job was started.

Type: Timestamp

Required: No

Status

The execution status of the synchronization job. When the `Status` field is set to `SUCCEEDED`, the synchronization job is done. If the status code is set to `FAILED`, the `ErrorCode` and `ErrorMessage` fields give you the reason for the failure.

Type: String

Valid Values: `FAILED | SUCCEEDED | SYNCING | INCOMPLETE | STOPPING | ABORTED`

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
- AWS SDK for Ruby V3

DataSourceToIndexFieldMapping

Maps a column or attribute in the data source to an index field. You must first create the fields in the index using the UpdateIndex (p. 149) operation.

Contents

DataSourceFieldName

The name of the column or attribute in the data source.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z][a-zA-Z0-9_]*$`

Required: Yes

DateFieldFormat

The type of data stored in the column or attribute.

Type: String

Length Constraints: Minimum length of 4. Maximum length of 40.

Required: No

IndexFieldName

The name of the field in the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 30.

Pattern: `^\P{C}*$`

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
- AWS SDK for Ruby V3

DataSourceVpcConfiguration

Provides information for connecting to an Amazon VPC.

Contents

SecurityGroupIds

A list of identifiers of security groups within your Amazon VPC. The security groups should enable Amazon Kendra to connect to the data source.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[-0-9a-zA-Z]+`

Required: Yes

SubnetIds

A list of identifiers for subnets within your Amazon VPC. The subnets should be able to connect to each other in the VPC, and they should have outgoing access to the Internet through a NAT device.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 6 items.

Length Constraints: Minimum length of 1. Maximum length of 200.

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
- AWS SDK for Ruby V3

Document

A document in an index.

Contents

AccessControlList

Information to use for user context filtering.

Type: Array of Principal (p. 190) objects

Array Members: Minimum number of 1 item. Maximum number of 200 items.

Required: No

Attributes

Custom attributes to apply to the document. Use the custom attributes to provide additional information for searching, to provide facets for refining searches, and to provide additional information in the query response.

Type: Array of DocumentAttribute (p. 176) objects

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Required: No

Blob

The contents of the document.

Documents passed to the `Blob` parameter must be base64 encoded. Your code might not need to encode the document file bytes if you're using an AWS SDK to call Amazon Kendra operations. If you are calling the Amazon Kendra endpoint directly using REST, you must base64 encode the contents before sending.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 153600.

Required: No

ContentType

The file type of the document in the `Blob` field.

Type: String

Valid Values: `PDF | HTML | MS_WORD | PLAIN_TEXT | PPT`

Required: No

Id

A unique identifier of the document in the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: Yes

S3Path

Information required to find a specific file in an Amazon S3 bucket.

Type: S3Path (p. 198) object

Required: No

Title

The title of the document.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

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
- AWS SDK for Ruby V3

DocumentAttribute

A custom attribute value assigned to a document.

Contents

Key

The identifier for the attribute.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[a-zA-Z0-9_][a-zA-Z0-9_-]*`

Required: Yes

Value

The value of the attribute.

Type: DocumentAttributeValue (p. 177) object

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
- AWS SDK for Ruby V3

DocumentAttributeValue

The value of a custom document attribute. You can only provide one value for a custom attribute.

Contents

DateValue

A date value expressed as seconds from the Unix epoch.

Type: Timestamp

Required: No

LongValue

A long integer value.

Type: Long

Required: No

StringListValue

A list of strings.

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 2048.

Required: No

StringValue

A string, such as "department".

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

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
- AWS SDK for Ruby V3

DocumentAttributeValueCountPair

Provides the count of documents that match a particular attribute when doing a faceted search.

Contents

Count

The number of documents in the response that have the attribute value for the key.

Type: Integer

Required: No

DocumentAttributeValue

The value of the attribute. For example, "HR."

Type: DocumentAttributeValue (p. 177) 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
- AWS SDK for Ruby V3

DocumentMetadataConfiguration

Specifies the properties of a custom index field.

Contents

Name

The name of the index field.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 30.

Required: Yes

Relevance

Provides manual tuning parameters to determine how the field affects the search results.

Type: Relevance (p. 193) object

Required: No

Search

Provides information about how the field is used during a search.

Type: Search (p. 199) object

Required: No

Type

The data type of the index field.

Type: String

Valid Values: `STRING_VALUE | STRING_LIST_VALUE | LONG_VALUE | DATE_VALUE`

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
- AWS SDK for Ruby V3

DocumentsMetadataConfiguration

Document metadata files that contain information such as the document access control information, source URI, document author, and custom attributes. Each metadata file contains metadata about a single document.

Contents

S3Prefix

A prefix used to filter metadata configuration files in the AWS S3 bucket. The S3 bucket might contain multiple metadata files. Use `S3Prefix` to include only the desired metadata files.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

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
- AWS SDK for Ruby V3

Facet

Information a document attribute

Contents

DocumentAttributeKey

The unique key for the document attribute.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

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
- AWS SDK for Ruby V3

FacetResult

The facet values for the documents in the response.

Contents

DocumentAttributeKey

The key for the facet values. This is the same as the `DocumentAttributeKey` provided in the query.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `[a-zA-Z0-9_][a-zA-Z0-9_-]*`

Required: No

DocumentAttributeValueCountPairs

An array of key/value pairs, where the key is the value of the attribute and the count is the number of documents that share the key value.

Type: Array of DocumentAttributeValueCountPair (p. 178) 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
- AWS SDK for Ruby V3

FaqStatistics

Provides statistical information about the FAQ questions and answers contained in an index.

Contents

IndexedQuestionAnswersCount

The total number of FAQ questions and answers contained in the index.

Type: Integer

Valid Range: Minimum value of 0.

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
- AWS SDK for Ruby V3

FaqSummary

Provides information about a frequently asked questions and answer contained in an index.

Contents

CreatedAt

The UNIX datetime that the FAQ was added to the index.

Type: Timestamp

Required: No

Id

The unique identifier of the FAQ.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: No

Name

The name that you assigned the FAQ when you created or updated the FAQ.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: No

Status

The current status of the FAQ. When the status is `ACTIVE` the FAQ is ready for use.

Type: String

Valid Values: `CREATING | UPDATING | ACTIVE | DELETING | FAILED`

Required: No

UpdatedAt

The UNIX datetime that the FAQ was last updated.

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
- AWS SDK for Ruby V3

Highlight

Provides information that you can use to highlight a search result so that your users can quickly identify terms in the response.

Contents

BeginOffset

The zero-based location in the response string where the highlight starts.

Type: Integer

Required: Yes

EndOffset

The zero-based location in the response string where the highlight ends.

Type: Integer

Required: Yes

TopAnswer

Indicates whether the response is the best response. True if this is the best response; otherwise, false.

Type: Boolean

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
- AWS SDK for Ruby V3

IndexConfigurationSummary

A summary of information about an index.

Contents

CreatedAt

The Unix timestamp when the index was created.

Type: Timestamp

Required: Yes

Id

A unique identifier for the index. Use this to identify the index when you are using operations such as `Query`, `DescribeIndex`, `UpdateIndex`, and `DeleteIndex`.

Type: String

Length Constraints: Fixed length of 36.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9-]*`

Required: No

Name

The name of the index.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1000.

Pattern: `[a-zA-Z0-9][a-zA-Z0-9_-]*`

Required: No

Status

The current status of the index. When the status is `ACTIVE`, the index is ready to search.

Type: String

Valid Values: `CREATING | ACTIVE | DELETING | FAILED | SYSTEM_UPDATING`

Required: Yes

UpdatedAt

The Unix timestamp when the index was last updated by the `UpdateIndex` operation.

Type: Timestamp

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
- AWS SDK for Ruby V3

IndexStatistics

Provides information about the number of documents and the number of questions and answers in an index.

Contents

FaqStatistics

The number of question and answer topics in the index.

Type: FaqStatistics (p. 183) object

Required: Yes

TextDocumentStatistics

The number of text documents indexed.

Type: TextDocumentStatistics (p. 203) object

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
- AWS SDK for Ruby V3

Principal

Provides user and group information for document access filtering.

Contents

Access

Whether to allow or deny access to the principal.

Type: String

Valid Values: `ALLOW | DENY`

Required: Yes

Name

The name of the user or group.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 200.

Pattern: `^\P{C}*$`

Required: Yes

Type

The type of principal.

Type: String

Valid Values: `USER | GROUP`

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
- AWS SDK for Ruby V3

QueryResultItem

A single query result.

A query result contains information about a document returned by the query. This includes the original location of the document, a list of attributes assigned to the document, and relevant text from the document that satisfies the query.

Contents

AdditionalAttributes

Type: Array of AdditionalResultAttribute (p. 155) objects

Required: No

DocumentAttributes

An array of document attributes for the document that the query result maps to. For example, the document author (Author) or the source URI (SourceUri) of the document.

Type: Array of DocumentAttribute (p. 176) objects

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Required: No

DocumentExcerpt

An extract of the text in the document. Contains information about highlighting the relevant terms in the excerpt.

Type: TextWithHighlights (p. 204) object

Required: No

DocumentId

The unique identifier for the document.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

DocumentTitle

The title of the document. Contains the text of the title and information for highlighting the relevant terms in the title.

Type: TextWithHighlights (p. 204) object

Required: No

DocumentURI

The URI of the original location of the document.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^(https?|ftp|file):\/\/(.*)`

Required: No

Id

The unique identifier for the query result.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 73.

Required: No

Type

The type of document.

Type: String

Valid Values: `DOCUMENT | QUESTION_ANSWER | ANSWER`

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
- AWS SDK for Ruby V3

Relevance

Provides information for manually tuning the relevance of a field in a search. When a query includes terms that match the field, the results are given a boost in the response based on these tuning parameters.

Contents

Duration

Specifies the time period that the boost applies to. For example, to make the boost apply to documents with the field value within the last month, you would use "2628000s". Once the field value is beyond the specified range, the effect of the boost drops off. The higher the importance, the faster the effect drops off. If you don't specify a value, the default is 3 months. The value of the field is a numeric string followed by the character "s", for example "86400s" for one day, or "604800s" for one week.

Only applies to `DATE` fields.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 10.

Pattern: `[0-9]+[s]`

Required: No

Freshness

Indicates that this field determines how "fresh" a document is. For example, if document 1 was created on November 5, and document 2 was created on October 31, document 1 is "fresher" than document 2. You can only set the `Freshness` field on one `DATE` type field. Only applies to `DATE` fields.

Type: Boolean

Required: No

Importance

The relative importance of the field in the search. Larger numbers provide more of a boost than smaller numbers.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 10.

Required: No

RankOrder

Determines how values should be interpreted.

When the `RankOrder` field is `ASCENDING`, higher numbers are better. For example, a document with a rating score of 10 is higher ranking than a document with a rating score of 1.

When the `RankOrder` field is `DESCENDING`, lower numbers are better. For example, in a task tracking application, a priority 1 task is more important than a priority 5 task.

Only applies to `LONG` and `DOUBLE` fields.

Type: String

Valid Values: `ASCENDING | DESCENDING`

Required: No

ValueImportanceMap

A list of values that should be given a different boost when they appear in the result list. For example, if you are boosting a field called "department," query terms that match the department field are boosted in the result. However, you can add entries from the department field to boost documents with those values higher.

For example, you can add entries to the map with names of departments. If you add "HR",5 and "Legal",3 those departments are given special attention when they appear in the metadata of a document. When those terms appear they are given the specified importance instead of the regular importance for the boost.

Type: String to integer map

Key Length Constraints: Minimum length of 1. Maximum length of 50.

Valid Range: Minimum value of 1. Maximum value of 10.

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
- AWS SDK for Ruby V3

RelevanceFeedback

Provides feedback on how relevant a document is to a search. Your application uses the SubmitFeedback (p. 143) operation to provide relevance information.

Contents

RelevanceValue

Whether to document was relevant or not relevant to the search.

Type: String

Valid Values: `RELEVANT | NOT_RELEVANT`

Required: Yes

ResultId

The unique identifier of the search result that the user provided relevance feedback for.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 73.

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
- AWS SDK for Ruby V3

S3DataSourceConfiguration

Provides configuration information for a data source to index documents in an Amazon S3 bucket.

Contents

AccessControlListConfiguration

Provides the path to the S3 bucket that contains the user context filtering files for the data source.

Type: AccessControlListConfiguration (p. 153) object

Required: No

BucketName

The name of the bucket that contains the documents.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 63.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Required: Yes

DocumentsMetadataConfiguration

Document metadata files that contain information such as the document access control information, source URI, document author, and custom attributes. Each metadata file contains metadata about a single document.

Type: DocumentsMetadataConfiguration (p. 180) object

Required: No

ExclusionPatterns

A list of glob patterns for documents that should not be indexed. If a document that matches an inclusion prefix also matches an exclusion pattern, the document is not indexed.

For more information about glob patterns, see glob (programming) in *Wikipedia*.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 50.

Required: No

InclusionPrefixes

A list of S3 prefixes for the documents that should be included in the index.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 50.

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
- AWS SDK for Ruby V3

S3Path

Information required to find a specific file in an Amazon S3 bucket.

Contents

Bucket

The name of the S3 bucket that contains the file.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 63.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Required: Yes

Key

The name of the file.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

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
- AWS SDK for Ruby V3

Search

Provides information about how a custom index field is used during a search.

Contents

Displayable

Determines whether the field is returned in the query response. The default is `true`.

Type: Boolean

Required: No

Facetable

Indicates that the field can be used to create search facets, a count of results for each value in the field. The default is `false` .

Type: Boolean

Required: No

Searchable

Determines whether the field is used in the search. If the `Searchable` field is `true`, you can use relevance tuning to manually tune how Amazon Kendra weights the field in the search. The default is `true` for string fields and `false` for number and date fields.

Type: Boolean

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
- AWS SDK for Ruby V3

ServerSideEncryptionConfiguration

Provides the identifier of the AWS KMS customer master key (CMK) used to encrypt data indexed by Amazon Kendra. Amazon Kendra doesn't support asymmetric CMKs.

Contents

KmsKeyId

The identifier of the AWS KMS customer master key (CMK). Amazon Kendra doesn't support asymmetric CMKs.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

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
- AWS SDK for Ruby V3

SharePointConfiguration

Provides configuration information for connecting to a Microsoft SharePoint data source.

Contents

CrawlAttachments

`TRUE` to include attachments to documents stored in your Microsoft SharePoint site in the index; otherwise, `FALSE`.

Type: Boolean

Required: No

DocumentTitleFieldName

The Microsoft SharePoint attribute field that contains the title of the document.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z][a-zA-Z0-9_]*$`

Required: No

FieldMappings

A list of `DataSourceToIndexFieldMapping` objects that map Microsoft SharePoint attributes to custom fields in the Amazon Kendra index. You must first create the index fields using the UpdateIndex (p. 149) operation before you map SharePoint attributes. For more information, see Mapping Data Source Fields.

Type: Array of DataSourceToIndexFieldMapping (p. 172) objects

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Required: No

SecretArn

The Amazon Resource Name (ARN) of credentials stored in AWS Secrets Manager. The credentials should be a user/password pair. For more information, see Using a Microsoft SharePoint Data Source. For more information about AWS Secrets Manager, see What Is AWS Secrets Manager in the *AWS Secrets Manager* user guide.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1284.

Pattern: `arn:[a-z0-9-\.]{1,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[a-z0-9-\.]{0,63}:[^/].{0,1023}`

Required: Yes

SharePointVersion

The version of Microsoft SharePoint that you are using as a data source.

Type: String

Valid Values: `SHAREPOINT_ONLINE`

Required: Yes

Urls

The URLs of the Microsoft SharePoint site that contains the documents that should be indexed.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^(https?|ftp|file):\/\/(.*)`

Required: Yes

VpcConfiguration

Provides information for connecting to an Amazon VPC.

Type: DataSourceVpcConfiguration (p. 173) 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
- AWS SDK for Ruby V3

TextDocumentStatistics

Provides information about text documents indexed in an index.

Contents

IndexedTextDocumentsCount

The number of text documents indexed.

Type: Integer

Valid Range: Minimum value of 0.

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
- AWS SDK for Ruby V3

TextWithHighlights

Provides text and information about where to highlight the text.

Contents

Highlights

The beginning and end of the text that should be highlighted.

Type: Array of Highlight (p. 186) objects

Required: No

Text

The text to display to the user.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

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
- AWS SDK for Ruby V3

TimeRange

Provides a range of time.

Contents

EndTime

The UNIX datetime of the end of the time range.

Type: Timestamp

Required: No

StartTime

The UNIX datetime of the beginning of the time range.

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
- 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

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

ValidationError

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

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.