
Amazon CloudSearch

Developer Guide

API Version 2011-02-01



Amazon CloudSearch: Developer Guide

Copyright © 2014 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, Cloudfront, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

What Is Amazon CloudSearch?	1
Search Data Format	3
Search Domain Configuration	3
Search Requests	5
Getting Started	8
Step 1: Before You Begin	9
Step 2: Create a Search Domain	9
Step 3: Send Data for Indexing	13
Step 4: Search Your Amazon CloudSearch Domain	16
Step 5: Delete Your Amazon CloudSearch Movies Domain	21
Making API Requests	22
Regions and Endpoints	22
Making Configuration Requests	24
Request Authentication	25
Making Document Service Requests	26
Making Search Requests	27
Creating a Search Domain	29
Configuring Access for a Search Domain	34
Monitoring Search Domains	39
Getting Domain Information	39
Viewing Search Metrics	44
Tracking your Amazon CloudSearch Usage and Charges	45
Deleting a Domain	47
Preparing Your Data	50
Mapping Document Data to Index Fields	50
Creating SDF Batches	51
Document Versions	53
Adding and Updating Documents	54
Deleting Documents	54
Generating SDF (Experimental)	55
Configuring Index Fields	58
Adding Sources for a Field	59
Command Line Tools	60
AWS Management Console	61
API	66
Configuring Text Options	68
Configuring Stemming	68
Configuring Stopwords	71
Configuring Synonyms	74
Uploading Data	77
Indexing Document Data	82
Searching Your Data	85
Submitting Search Requests	86
Searching Text Fields	87
Using Boolean Operators in Text Searches	89
Using Wildcards in Text Searches	90
Searching for Phrases in Text Fields	91
Searching Literal Fields	91
Searching Uint Fields	92
Constructing Boolean Search Queries	93
Controlling Search Results	94
Getting Results as XML	94
Paginating Results	95
Retrieving Data from Index Fields	95
Sorting Results	96
Getting and Using Facet Information	96
Getting Facet Information for Text and Literal Fields	97
Getting Facet Information for Uint Fields	97

Getting Facet Information for Particular Values	98
Sorting Facet Information	99
Using Facet Information	100
Tuning Search Requests	103
Customizing Result Ranking	105
Configuring Rank Expressions	106
Defining Rank Expressions in Search Requests	107
Defining Rank Expressions in a Domain Configuration	107
Using Relative Field Weighting to Customize Text Relevance	110
Comparing Rank Expressions	111
Ranking Search Results	113
Constraining Search Results	113
Searching and Ranking Results by Geographic Location	114
Representing Locations	114
Searching Within an Area	115
Sorting Results by Distance	116
Searching DynamoDB Data	117
Using the Amazon CloudSearch Console	123
Amazon CloudSearch Dashboard	123
Domain Dashboard	124
Domain Status	125
Search Domains	125
Processing	125
Needs Indexing	125
Run a Test Search	126
Rank Comparison	126
Upload Data	128
Search Count Metrics	130
Top Documents Metrics	130
Top Searches Metrics	130
Access Policies	131
Indexing Options	131
Rank Expressions	132
Stems	133
Stopwords	134
Synonyms	135
Command Line Tool Reference	136
Using the Command Line Tools	136
Prerequisites	137
Installing the Command Line Tools	137
Running the Amazon CloudSearch Commands	139
Common Options	139
cs-configure-access-policies	140
cs-configure-fields	143
cs-configure-ranking	146
cs-configure-text-options	148
cs-create-domain	150
cs-configure-from-sdf	151
cs-delete-domain	153
cs-describe-domain	154
cs-index-documents	155
cs-post-sdf	157
Experimental Tools	158
cs-generate-sdf	158
Configuration API Reference	164
Actions	165
CreateDomain	166
DefineIndexField	167

DefineRankExpression	169
DeleteDomain	171
DeleteIndexField	172
DeleteRankExpression	174
DescribeDefaultSearchField	176
DescribeDomains	177
DescribeIndexFields	178
DescribeRankExpressions	179
DescribeServiceAccessPolicies	180
DescribeStemmingOptions	181
DescribeStopwordOptions	182
DescribeSynonymOptions	183
IndexDocuments	184
UpdateDefaultSearchField	185
UpdateServiceAccessPolicies	187
UpdateStemmingOptions	189
UpdateStopwordOptions	191
UpdateSynonymOptions	193
Data Types	194
AccessPoliciesStatus	195
CreateDomainResult	195
DefaultSearchFieldStatus	196
DefineIndexFieldResult	196
DefineRankExpressionResult	197
DeleteDomainResult	197
DeleteIndexFieldResult	197
DeleteRankExpressionResult	197
DescribeDefaultSearchFieldResult	198
DescribeDomainsResult	198
DescribeIndexFieldsResult	198
DescribeRankExpressionsResult	199
DescribeServiceAccessPoliciesResult	199
DescribeStemmingOptionsResult	199
DescribeStopwordOptionsResult	200
DescribeSynonymOptionsResult	200
DomainStatus	200
IndexDocumentsResult	202
IndexField	202
IndexFieldStatus	203
LiteralOptions	204
NamedRankExpression	204
OptionStatus	205
RankExpressionStatus	206
ServiceEndpoint	207
SourceAttribute	207
SourceData	208
SourceDataMap	208
SourceDataTrimTitle	209
StemmingOptionsStatus	209
StopwordOptionsStatus	210
SynonymOptionsStatus	210
TextOptions	211
UIntOptions	212
UpdateDefaultSearchFieldResult	212
UpdateServiceAccessPoliciesResult	212
UpdateStemmingOptionsResult	213
UpdateStopwordOptionsResult	213
UpdateSynonymOptionsResult	213

Common Parameters	213
Common Errors	215
Document Service API Reference	217
documents/batch	217
documents/batch JSON API	218
documents/batch XML API	221
Search API Reference	227
search	227
Search Requests	228
Search Response	234
Search Status Codes	238
Handling Errors	241
Troubleshooting	243
Text Processing	247
Limits	251
Articles and Tutorials	253
Using Amazon CloudSearch with MySQL/Amazon RDS	255
Abstract	255
Full-text Search	255
Integrating Amazon CloudSearch with MySQL	257
Conclusion	264
Amazon CloudSearch Glossary	266
Document History	270

What Is Amazon CloudSearch?

Topics

- [Search Data Format in Amazon CloudSearch \(p. 3\)](#)
- [Search Domain Configuration in Amazon CloudSearch \(p. 3\)](#)
- [Search Requests in Amazon CloudSearch \(p. 5\)](#)
- [Automatic Scaling in Amazon CloudSearch \(p. 6\)](#)

Amazon CloudSearch is a fully-managed service in the cloud that makes it easy to set up, manage, and scale a search solution for your website. Amazon CloudSearch enables you to search large collections of data such as web pages, document files, forum posts, or product information. With Amazon CloudSearch, you can quickly add search capabilities to your website without having to become a search expert or worry about hardware provisioning, setup, and maintenance. As your volume of data and traffic fluctuates, Amazon CloudSearch automatically scales to meet your needs.

You can use Amazon CloudSearch to index and search both structured data and plain text. Amazon CloudSearch supports full text search, searching within fields, prefix searches, Boolean searches, and faceting. You can get search results in JSON or XML, sort and filter results based on field values, and rank results alphabetically, numerically, or according to custom rank expressions.

To build a search solution with Amazon CloudSearch, you:

- **Create and configure a search domain.** A [search domain](#) encapsulates your searchable data and the search instances that handle your search requests. You set up a separate domain for each different data set you want to search.
- **Upload the data you want to search to your domain.** Amazon CloudSearch automatically indexes your data and deploys the search index to one or more search instances.
- **Search your domain.** You send a [search request](#) to your domain's search endpoint as an HTTP/HTTPS GET request.

Note

Looking for sample code? The [cloudsearchdemos](#) repository on GitHub contains sample application code that demonstrates how to use Amazon CloudSearch. The first sample, [CloudSearchGeoSpatial](#), shows how you can implement [location-based searching and sorting \(p. 114\)](#).

The rest of this section introduces the key concepts and terms that will help you understand what you need to do to build a search solution with Amazon CloudSearch:

- [Search Data Format in Amazon CloudSearch \(p. 3\)](#) describes the kinds of data you can search and the format used to submit data to Amazon CloudSearch for indexing.
- [Search Domain Configuration in Amazon CloudSearch \(p. 3\)](#) describes the options you can configure to control how your data is indexed and what information you can retrieve when you search.
- [Search Requests in Amazon CloudSearch \(p. 5\)](#) describes how search requests are submitted and processed.

For a high-level overview of Amazon CloudSearch, service highlights, and pricing information, see the [Amazon CloudSearch detail page](#). The rest of this guide describes how to use Amazon CloudSearch and provides detailed information about the APIs and command line tools. If you are new to Amazon CloudSearch, you should begin with [Getting Started with Amazon CloudSearch \(p. 8\)](#). For more information about working with your own data sets, see [Preparing Your Data for Amazon CloudSearch \(p. 50\)](#). For more information about constructing searches with the Amazon CloudSearch query language, see [Searching Your Data with Amazon CloudSearch \(p. 85\)](#).

The following table lets you jump directly to specific task or reference topics.

How Do I?	Relevant Sections
Set up my first search domain	Getting Started with Amazon CloudSearch (p. 8)
Manage my search domains	Creating an Amazon CloudSearch Domain (p. 29) Configuring Access for an Amazon CloudSearch Domain (p. 34) Getting Information about an Amazon CloudSearch Domain (p. 39) Deleting an Amazon CloudSearch Domain (p. 47)
Configure index and search options for my domains	Configuring Index Fields for an Amazon CloudSearch Domain (p. 58) Configuring Text Options for an Amazon CloudSearch Domain (p. 68) Customizing Result Ranking with Amazon CloudSearch (p. 105)
Format my data for Amazon CloudSearch	Preparing Your Data for Amazon CloudSearch (p. 50)
Upload and index my data	Uploading Data to an Amazon CloudSearch Domain (p. 77) Indexing Document Data with Amazon CloudSearch (p. 82)
Search my domains	Searching Your Data with Amazon CloudSearch (p. 85)
Install and use the Amazon CloudSearch command line tools	Amazon CloudSearch Command Line Tool Reference (p. 136)
Get more information about the Amazon CloudSearch APIs	Making Amazon CloudSearch API Requests (p. 22) Amazon CloudSearch Configuration API Reference (p. 164) Amazon CloudSearch Document Service API Reference (p. 217) Amazon CloudSearch Search API Reference (p. 227) Limits in Amazon CloudSearch (p. 251)

Search Data Format in Amazon CloudSearch

The collection of data that you want to search (sometimes referred to as your [corpus](#)) can consist of unstructured full-text documents, semi-structured documents such as those formatted in mark-up languages like XML, or structured data that conforms to a strict data model. To make your data searchable, you describe it using the Search Data Format (SDF) and upload the resulting SDF data to your search domain.

Each item that you want to be able to return as a search result (such as a forum post or web page) is represented as a [document](#) in SDF. Every document has a unique id, a version number, and one or more fields that contain the data that you want to search and return in results. An SDF [batch](#) is a collection of add and delete requests for individual documents. SDF batches must be valid JSON or XML and conform to the SDF data conventions.

Amazon CloudSearch generates a [search index](#) from your SDF data according to your domain's configuration options. As your data changes, you submit SDF updates to add, change, or delete documents from your index. Updates are applied continuously, so your changes become searchable in near real-time.

For information about how to represent your data in SDF, see [Preparing Your Data for Amazon CloudSearch \(p. 50\)](#). To see the JSON schema for SDF, go to [JSON documents/batch Requests \(p. 218\)](#). To see the XML schema for SDF, go to [XML documents/batch Requests \(p. 221\)](#).

Search Domain Configuration in Amazon CloudSearch

To build an index from your SDF data, Amazon CloudSearch needs to know what data you want to search, what data you want to be able to include in the search results, what data you want to use as facets, and if any custom stopwords, synonyms, and stems need to be defined for your data set. You define this metadata in your domain configuration by configuring indexing and text options. In your domain configuration, you also specify access policies to control who can send data updates and search your domain, and rank expressions to customize how search results are ranked.

Indexing Options

A domain's indexing options configure the index fields that will be included in the search index. An index field represents a named field and value pair that you want to store in your index. You configure an index field for each SDF document field that will be searched, used as a facet, or returned in search results.

Index Fields

Every index field has a unique name and a source that specifies one or more SDF document fields. The sources are used to populate the index field. If no source is specified, the source defaults to the SDF document field that has the same name as the index field. An index field definition also includes meta-information such as:

- The index field type.
- Whether a literal field is searchable (Text and uint fields are always searchable.)
- Whether the value of a text or literal field can be returned in results/ (Uint fields are always returnable.)
- Whether facet counts can be calculated for a text or literal field. (Facet counts can always be calculated for uint fields.)

Amazon CloudSearch supports three types of index fields:

- `text`—contains arbitrary alphanumeric data. For example, a text field might contain a name, description, or the entire body of a document. Text fields are always searchable and Amazon CloudSearch performs text processing on them according to the stopwords, synonyms, and stems you configure in your domain's text options.
- `literal`—contains an identifier or other data that you want to be able to match exactly. Unlike text fields, Amazon CloudSearch does not perform any text processing on literal fields. Literal fields can be used for fields that have a small set of possible values, as well as for more arbitrary values like email addresses or titles where an exact match is important. Literal fields are frequently used to enable faceted searches where you want to count the number of exact matches for a particular value.
- `uint`—contains an unsigned integer value. For example, you might use a `uint` field for a field that contains a quantity or numerical rating, or for a date field that contains a `time_t` value.

For information about how to configure index fields for Amazon CloudSearch, see [Configuring Index Fields for an Amazon CloudSearch Domain](#) (p. 58).

Facets

A facet is an index field that represents a category that you want to use to refine and filter search results. When you submit search requests to Amazon CloudSearch, you can request facet information to find out how many hits share the same value in a facet. You can display this information along with the search results and use it to enable users to interactively refine their searches. (This is often referred to as faceted navigation or faceted search.)

A facet can be any numeric field or a text or literal field that has faceting enabled in your domain configuration. To request facet information in your search request, you specify:

- One or more facets
- Facet constraints that specify the particular values you want to count (optional)
- How you want the facet values to be sorted in the results (optional)

For each facet, Amazon CloudSearch calculates the number of hits that share the same value. If you specify constraints, the facet counts are calculated only for values that match the constraints. Only constraints that have matches are included in the facet results.

Note

Values from a facet-enabled text or literal field cannot be returned in the search results. Text and literal fields can be facet-enabled or result-enabled, but not both. If you want to return the value from an SDF document field as well as use the field as a facet, create two index fields that use the same SDF document field as a source and make one result-enabled, and the other facet-enabled.

For information about configuring facets, see [Configuring Index Fields for an Amazon CloudSearch Domain](#) (p. 58). For information about using facet information to support faceted navigation, see [Getting and Using Facet Information in Amazon CloudSearch](#) (p. 96).

Text Options

During indexing, Amazon CloudSearch performs a number of text-processing steps on text fields. First, Amazon CloudSearch strips punctuation and splits the text into individual terms that are indexed separately. For example, the string `spider-man` would be split into two terms, `spider` and `man`.

Text fields are then processed using the domain-specific stopwords, stemming, and synonym dictionaries:

- Stopwords configured for the domain are excluded from the index. For example, the stopwords dictionary generally contains insignificant, frequently occurring terms such as "a", "and", and "the" that would result in a massive number of matches if they were included in the index.

- Related words are mapped to a common stem according to the stemming dictionary configured for the domain. For example, the stemming dictionary might map "running" and "ran" to the stem "run".
- Synonyms are mapped according to the synonym dictionary configured for the domain. For example, the synonym dictionary might define "colt" and "filly" as synonyms for "horse".

Amazon CloudSearch defines a default stopword dictionary that you can fine-tune for your application. Stemming and synonym dictionaries are application-specific and are empty by default. For information about how to configure stopwords, stems, and synonyms for your domain, see [Configuring Text Options for an Amazon CloudSearch Domain](#) (p. 68).

For more information about how Amazon CloudSearch normalizes and tokenizes text and applies configured text options when indexing text fields and processing search requests, see [Text Processing in Amazon CloudSearch](#) (p. 247).

Access Policies

Access to your search domain's endpoints is restricted by IP address so that only authorized hosts can submit documents and send search requests. IP address authorization is used only to control access to the document and search endpoints. All Amazon CloudSearch configuration requests must be authenticated using standard AWS authentication.

Amazon CloudSearch access policies are specified using the AWS Identity and Access Management (IAM) [Access Policy Language](#).

For information about how to configure access policies for your domain, see [Configuring Access for an Amazon CloudSearch Domain](#) (p. 34).

Rank Expressions

You can customize how search results are ranked by defining your own rank expressions. Rank expressions are numeric expressions that can be used at search time to calculate a score for every document that matches the search. A rank expression uses standard numeric operators and functions and can reference uint fields, other rank expressions, a document's text_relevance score. When you submit search requests, you specify the rank expression(s) you want to use to rank or constrain the search results.

A document's text_relevance score indicates how relevant a particular search hit is to the search request. To calculate the relevance score, Amazon CloudSearch takes into account how many times the search terms appear (term frequency) and how close the search terms are to each other (proximity).

For information about how to configure rank expressions for your domain, see [Customizing Result Ranking with Amazon CloudSearch](#) (p. 105).

Search Requests in Amazon CloudSearch

You submit search requests to your domain's search endpoint as HTTP/HTTPS GET requests. You can perform free text and Boolean searches, and specify a variety of options to constrain your search, request facet information, control ranking, and specify what you want to be returned in the results. You can get search results in either JSON or XML. By default, Amazon CloudSearch returns results in JSON.

When you submit a search request, Amazon CloudSearch performs text-processing on the search terms. The search terms are tokenized, stopwords are removed, and stems are mapped according to the domain configuration.

Once this preprocessing is complete, Amazon CloudSearch looks up the search terms in the index and identifies all of the documents that match the request. To generate a response, Amazon CloudSearch

processes this list of search hits to filter and rank the matching documents and compute facets. Amazon CloudSearch then returns the response in JSON or XML.

By default, Amazon CloudSearch returns search results ranked according to the hits' `text_relevance` scores. Alternatively, your request can specify the index field or rank expression that you want to use to sort the hits. For example, you might want to rank hits by an index field that contains the price or a rank expression that calculates popularity.

For more information about searching, ranking, and paginating results, see [Searching Your Data with Amazon CloudSearch \(p. 85\)](#).

Automatic Scaling in Amazon CloudSearch

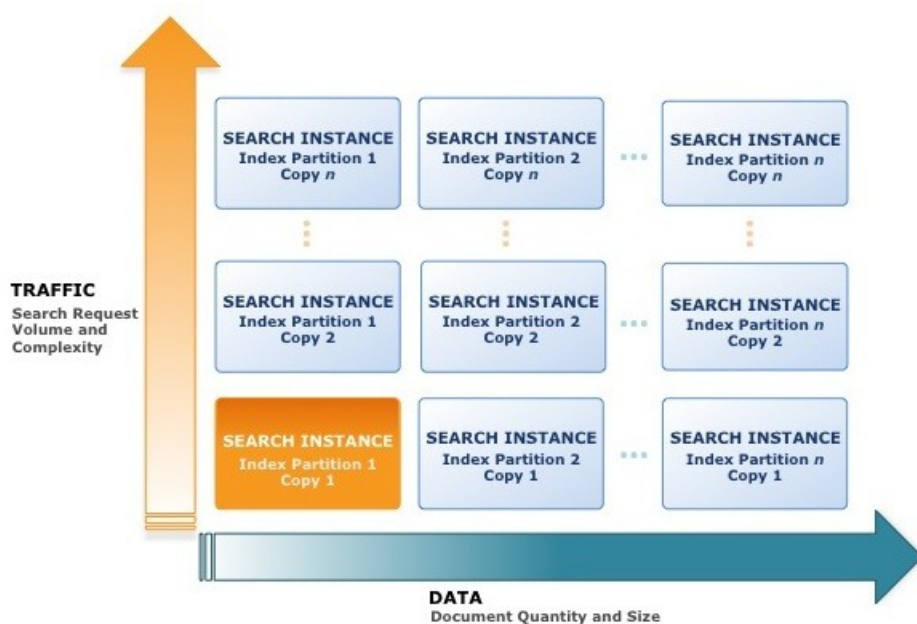
A search domain has one or more search instances, each with a finite amount of RAM and CPU resources for indexing data and processing requests. The number of search instances deployed for a domain depends on the documents in your collection and the volume and complexity of your search requests.

As a managed service, Amazon CloudSearch determines the size and number of search instances required to deliver low latency, high throughput search performance. When you upload your data and configure your index, Amazon CloudSearch builds an index and picks the appropriate initial search instance type. As you use your search domain, Amazon CloudSearch automatically scales to accommodate the amount of data uploaded to the domain and the volume and complexity of search requests.

Note

At this time, scaling is completely automatic. Amazon CloudSearch does not provide a mechanism for choosing a particular search instance type or configuring the desired number of instances.

When you create a search domain, a single instance is deployed for the domain. As the following illustration shows, you always have at least one instance for your domain and additional instances are added as the volume of data or traffic increases.



Scaling for Data

When the amount of data you add to your domain exceeds the capacity of the initial search instance type, Amazon CloudSearch automatically scales your search domain to a larger search instance type. Once a domain exceeds the capacity of the largest search instance type, Amazon CloudSearch partitions the search index across multiple search instances. (The number of search instances required to hold the index partitions is sometimes referred to as the domain's width.)

Conversely, if the volume of data in your domain shrinks, your domain is scaled down to fewer search instances or a smaller search instance type to minimize costs.

Note

If your domain has scaled up to accommodate your index size and you delete a large number of documents, the domain scales down the next time the full index is rebuilt. While this is periodically done automatically, to scale down as quickly as possible you can explicitly [rebuild the index \(p. 82\)](#) when you are done deleting documents.

Scaling for Traffic

As your search request volume or complexity increases, it takes more processing power to handle the load. A high volume of document uploads also increases the load on a domain's search instances. When a search instance nears its maximum load, Amazon CloudSearch automatically deploys a duplicate search instance to provide additional processing power. (The number of duplicate search instances is sometimes referred to as the domain's depth.)

Conversely, when traffic drops, Amazon CloudSearch removes unneeded search instances to minimize costs. For example, a new domain might scale up to handle the initial influx of documents, and scale back down once you have finished uploading your data and are only submitting updates.

If your domain experiences a sudden surge in traffic, Amazon CloudSearch will automatically deploy additional search instances. However, it takes a few minutes to set up the new instances, so you might see an increase in 5xx errors until they are ready to start processing requests. For more information about handling 5xx errors, see [Handling Errors in Amazon CloudSearch \(p. 241\)](#).

Keep in mind that the type and complexity of your search requests can impact overall search performance and in some cases increase the number of search instances required to operate your domain. For more information, see [Tuning Search Requests in Amazon CloudSearch \(p. 103\)](#). Submitting a high volume of small or single-document batches can also have a negative impact on your search domain's performance.

Getting Started with Amazon CloudSearch

Topics

- [Step 1: Before You Begin with Amazon CloudSearch](#) (p. 9)
- [Step 2: Create an Amazon CloudSearch Domain](#) (p. 9)
- [Step 3: Send Data to Amazon CloudSearch for Indexing](#) (p. 13)
- [Step 4: Search Your Amazon CloudSearch Domain](#) (p. 16)
- [Step 5: Delete Your Amazon CloudSearch Movies Domain](#) (p. 21)

To start searching your data with Amazon CloudSearch, you simply:

- Create and configure a search domain
- Upload and index the data you want to search
- Send search requests to your domain

This tutorial shows you how to get up and running using the AWS Management Console for Amazon CloudSearch. To make it even easier to get started, we've generated a sample data set of over 5,000 popular movie titles that you can download and examine, upload to your own search domain, and submit search queries against to see how Amazon CloudSearch works.

Using the AWS Management Console and the sample movie data, you'll quickly have your own searchable movie database running in Amazon CloudSearch.

To begin, [Get Signed Up](#).

The following video steps through this tutorial and shows how to create your first search domain through the console: [Getting Started with Amazon CloudSearch](#).

Step 1: Before You Begin with Amazon CloudSearch

To use Amazon CloudSearch, you need an Amazon Web Services (AWS) account. Your AWS account enables you to access Amazon CloudSearch and other AWS services, such as Amazon Simple Storage Service (Amazon S3) and Amazon Elastic Compute Cloud (Amazon EC2). As with other AWS services, you pay only for the Amazon CloudSearch resources you use. There are no sign up fees and charges are not incurred until you create a search domain.

If you already have an AWS account, you are automatically signed up for Amazon CloudSearch.

Note

For console access, use your IAM user name and password to sign in to the [AWS Management Console](#) using the [IAM sign-in page](#). IAM lets you securely control access to AWS services and resources in your AWS account. For more information about getting credentials, see [How Do I Get Security Credentials?](#) in the *AWS General Reference*.

To create an AWS account

1. Go to <https://aws.amazon.com> and click **Sign Up Now**.
2. Follow the instructions to sign up. You will need to enter payment information before you can begin using Amazon CloudSearch.

Step 2: Create an Amazon CloudSearch Domain

An Amazon CloudSearch domain encapsulates a collection of data you want to search, the search instances that process your search requests, and a configuration that controls how your data is indexed and searched. You create a separate search domain for each collection of data you want to make searchable. For each domain, you configure indexing options that describe the fields you want to include in your index and how you want to use them, text options that define domain-specific stopwords, stems, and synonyms, rank expressions that you can use to customize how search results are ranked, and access policies that control access to the domains document and search endpoints.

You interact with a search domain to:

- Configure index and search options
- Submit data for indexing
- Perform searches

Each domain has a unique endpoint through which you submit search requests to the domain. For example, the endpoint for a domain called *movies* created in the US East (Northern Virginia) Region might be:

```
search.123456789012-movies.us-east-1.cloudsearch.amazonaws.com
```

When creating a search domain, you specify a unique name for the domain. Domain names must start with a letter or number and be at least 3 and no more than 28 characters long. The allowed characters are: a-z, 0-9, and hyphen (-). By default, new domains are created in the US East (Northern Virginia) Region. To create a domain in another region, you must explicitly specify the region when creating the domain.

To configure the new domain, you need to specify:

Amazon CloudSearch Developer Guide

Step 2: Create a Search Domain

- The index fields you want to be able to search, use as facets, and return in search results.
- Access policies for the domain's document service and search service endpoints.

This tutorial shows you how to create and interact with a domain using the Amazon CloudSearch console. For information about how to use the command line tools and APIs, see [Creating an Amazon CloudSearch Domain](#).

Important

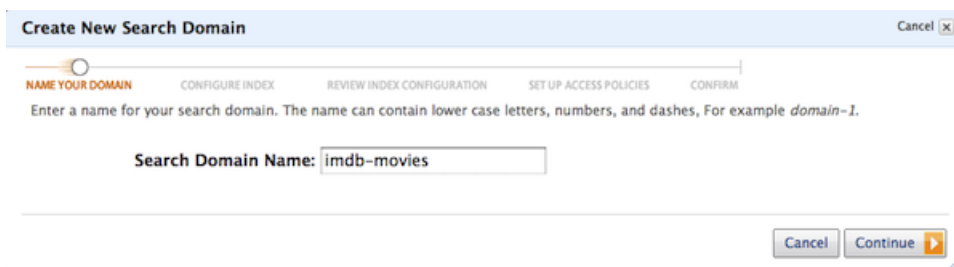
The domain you're about to create will be live and you will incur the standard Amazon CloudSearch usage fees for the domain until you delete it. For more information about Amazon CloudSearch usage rates, go to the [Amazon CloudSearch detail page](#).

To create your movies domain

1. Go to the Amazon CloudSearch console at [Amazon CloudSearch console](#).
2. On the Welcome to Amazon CloudSearch page, click **Create Your First Search Domain**.



3. On the **NAME YOUR DOMAIN** step, enter a name for your new domain and click **Continue**. Domain names must start with a letter or number and be at least 3 and no more than 28 characters. Domain names can contain the following characters: a-z (lower case), 0-9, and - (hyphen). Upper case letters and underscores are not allowed.



4. On the **CONFIGURE INDEX** step, click **Use a predefined configuration**, select **IMDB movies (demo)**, and click **Continue**. You can also automatically configure a search domain by choosing the predefined configuration for the type of data you want to index, or by uploading a sample of your data.

○ Use a predefined configuration
Load configuration for indexing: **IMDB movies (demo)**

- On the **REVIEW INDEX CONFIGURATION** step, review the index fields that will be configured. Five fields are configured automatically for the imdb-movie data: actor, director, genre, title, and year.
 - The actor, director, and title fields are text fields and will be searched by default if no search field is specified in a search request. The contents of those fields can also be returned in search results.
 - The genre field is configured as a literal field and is designated as a facet so it can be used to sort and filter the results. Because it's a facet, it cannot be returned in the search results—if you want to retrieve contents of the genre field when you search, you can configure an additional field with the same source data and make it result-enabled. (For more information, see [Configuring Index Fields for an Amazon CloudSearch Domain](#).)
 - The year field is configured as a uint field. You cannot change the configuration of a uint field—uint fields are always search-enabled, facet-enabled, and result-enabled.

When you are finished reviewing the indexing options, click **Continue**.

Create New Search Domain [Cancel]

NAME YOUR DOMAIN | CONFIGURE INDEX | **REVIEW INDEX CONFIGURATION** | SET UP ACCESS POLICIES | CONFIRM

The suggested index configuration is shown below. You can edit these fields or add additional fields. Click **Continue** when you are finished making changes.

Suggested Index Field Configuration

Name	Type	Search	Facet	Result	Default Value	Source	Remove
actor	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		[add]	<input type="checkbox"/>
director	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		[add]	<input type="checkbox"/>
genre	literal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		[add]	<input type="checkbox"/>
title	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		[add]	<input type="checkbox"/>
year	uint	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		[add]	<input type="checkbox"/>

< Back

- On the **SET UP ACCESS POLICIES** step, click **Recommended rules** and click **Continue**. The recommended rules allow access to the search endpoint from all IP addresses, and restrict access to the document service to the IP address you specify.

Important

If you do not configure access rules for your search domain, you will only be able to interact with the domain through the Amazon CloudSearch console. By default, the document service and search service endpoints are configured to block all IP addresses.

Keep in mind that if you do not have a static IP address, you must re-authorize your computer whenever your IP address changes. If your IP address is assigned dynamically, it is also likely that you're sharing that address with other computers on your network. This means that when you authorize the IP address, all computers that share it will be able to access your search domain's document service endpoint.

Amazon CloudSearch Developer Guide Step 2: Create a Search Domain

Create New Search Domain Cancel

NAME YOUR DOMAIN CONFIGURE INDEX REVIEW INDEX CONFIGURATION **SET UP ACCESS POLICIES** CONFIRM

To authorize or block IP addresses, add one or more access policy rules. You can always access all services through the console, regardless of the rules defined here.

Your IP address appears to be **192.0.2.10**. [change](#)

Set my policy to:

- Recommended rules (Search service: Allow all. Document service: Your IP address only)
- Allow only my IP address access to all services (can be used for testing)
- Allow everyone access to all services (not recommended because anyone can upload documents)
- Deny everyone access to all services (except through the console)
- Copy access policy from another domain

Current Policy:

Effect	Service	IP Ranges	Remove
Allow	Search	0.0.0.0 / 0	<input type="checkbox"/>
Allow	Document	192.0.2.10 / 32	<input type="checkbox"/>

[Add a New Rule](#)

[< Back](#) Cancel Continue

7. On the **CONFIRM** step, review the domain configuration and click **Confirm** to create your domain.

Create New Search Domain Cancel

NAME YOUR DOMAIN CONFIGURE INDEX REVIEW INDEX CONFIGURATION SET UP ACCESS POLICIES **CONFIRM**

Review the information below, then click **Confirm**.

Search Domain Name imdb-movies [Edit Domain Name](#)

Index Fields 5 index fields:

- actor
- director
- genre
- title
- show 1 more...

[Edit Index Fields](#)

Access Policies 2 rules:

- Allow search service for 0.0.0.0/0
- Allow document service for 192.0.2.10/32

[Edit Access Policies](#)

[< Back](#) Cancel Confirm

8. Once the domain has been created, click **OK** to exit the Create New Search Domain wizard and go to the domain's dashboard.

Create New Search Domain Cancel

NAME YOUR DOMAIN CONFIGURE INDEX REVIEW INDEX CONFIGURATION SET UP ACCESS POLICIES **CONFIRM**

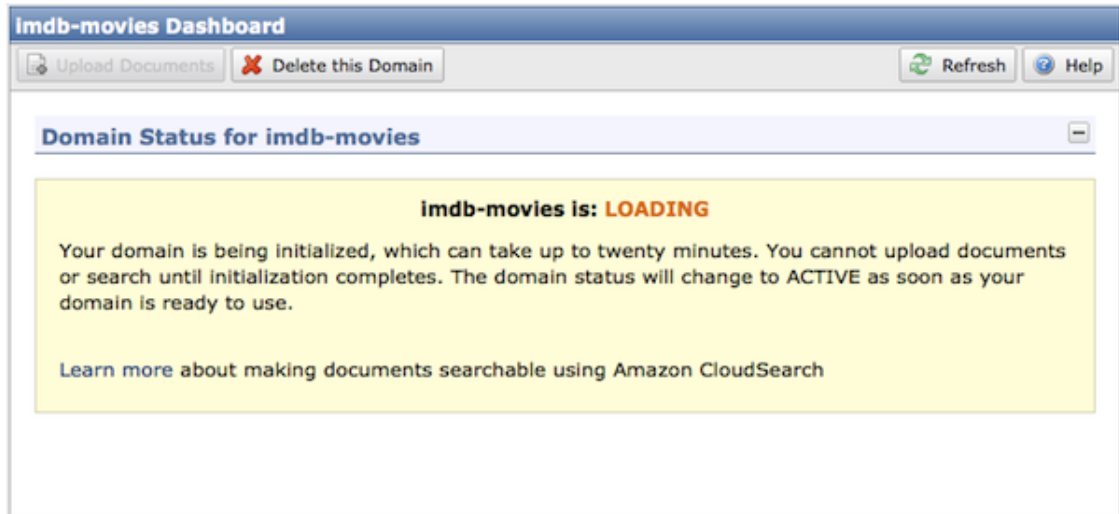
Congratulations

You have successfully created the domain **movies**. The domain will take several minutes to initialize. Once the domain is initialized, you may upload documents, and make any other changes to the domain.

OK

When you create a new domain, Amazon CloudSearch initializes resources for the domain, which can take around half an hour. During this initialization process, the status of the domain will be **LOADING**.

You can begin uploading the data you want to search as soon as the domain status changes to PROCESSING. Once the status changes to ACTIVE, your domain will be fully-functional and available to process search requests.



Note

While you can start uploading documents through the console once the domain status reaches the PROCESSING state, you won't be able to upload data through the command line tools or document service API until the domain status is ACTIVE.

Step 3: Send Data to Amazon CloudSearch for Indexing

You upload the data you want to search to your domain so that Amazon CloudSearch can build and deploy a searchable index. The format used to submit documents to Amazon CloudSearch is called Search Data Format (SDF). The AWS Management Console can automatically generate SDF from several types of files:

- Comma Separated Value (.csv)
- Adobe Portable Document Format (.pdf)
- HTML (.htm, .html)
- Microsoft Excel (.xls, .xlsx)
- Microsoft PowerPoint (.ppt, .pptx)
- Microsoft Word (.doc, .docx)
- Text Documents (.txt)
- JSON Documents (.json)
- XML Documents (.xml)

For most file types, including JSON and XML, Amazon CloudSearch adds a single add document operation to the SDF batch for each source file. If metadata is available for the file, the metadata is mapped to corresponding document fields—the fields generated from the document metadata vary depending on the file type. The contents of the source file are parsed into a single text field. If the file contains more than 1 MB of data, the data mapped to the text field is truncated so that the document does not exceed 1 MB.

CSV files are handled differently. When you upload a CSV file, Amazon CloudSearch uses the contents of the first row to define the document fields, and creates a separate document each following row. If there is a column header called *docid*, the values in that column are used as the document IDs. If necessary, the docid values are normalized to conform to the allowed character set: a-z (lower-case letters), 0-9, and _ (underscore). If there is no docid column, a unique ID is generated for each document based on the filename and row number. Similarly, if there is a column called *version*, the values in that column are used as the versions for the document updates. Version numbers must be specified as 32-bit unsigned integers. If there is no version column, version numbers are generated based on a timestamp.

If you upload multiple types of files, CSV files are parsed row-by-row, and non-CSV files are treated as individual documents.

The sample IMDB movies data is already formatted as SDF and contains add requests for over 5,000 popular movies. Each add request specifies a unique ID for the movie, a document version number, and fields that contain the movie data such as title and genre.

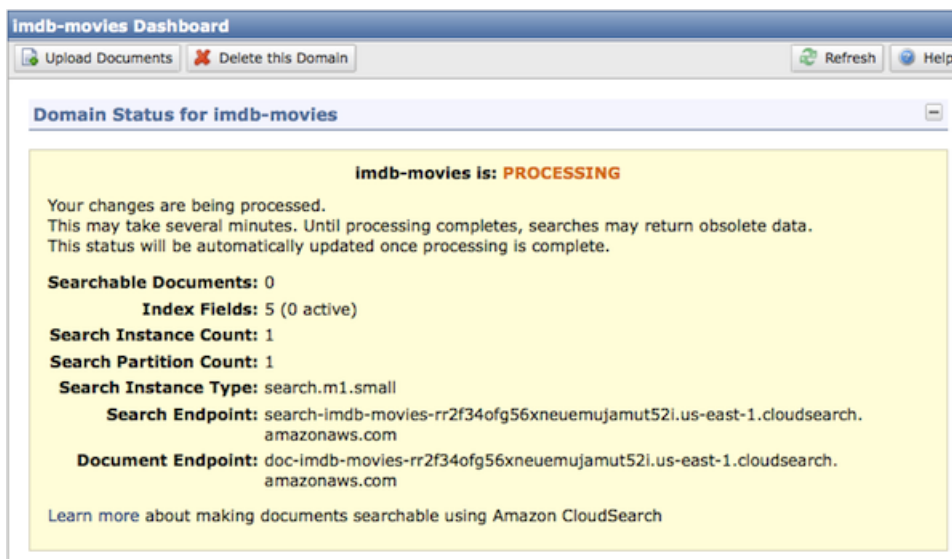
This tutorial shows how to submit data through the Amazon CloudSearch console, but you can also [generate SDF](#) and [upload data](#) with the command line tools, and submit SDF batches through the [DocumentsBatch API](#).

To add the sample data to your movies domain

1. Go to the Amazon CloudSearch console at [Amazon CloudSearch console](#).
2. In the **Navigation** panel, click the name of your movies domain to view the domain dashboard.
3. At the top of the domain dashboard, click the **Upload Documents** button.

Note

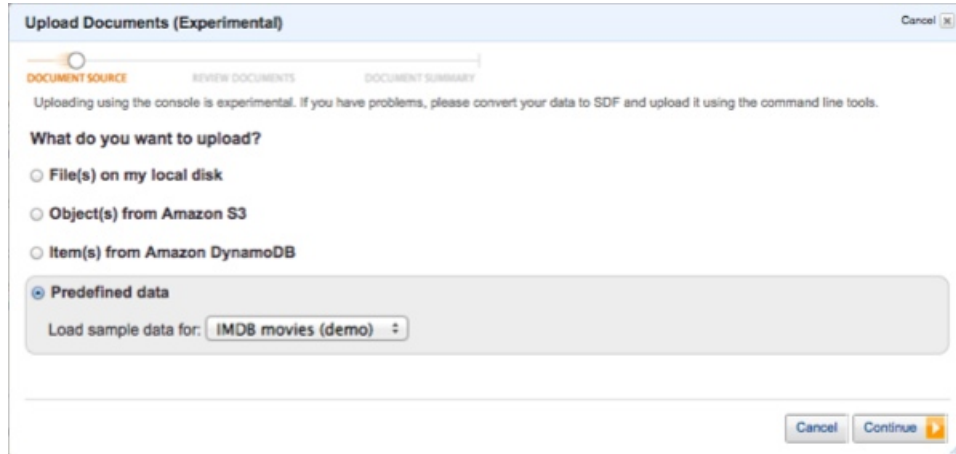
The **Upload Documents** button is available once the domain status is PROCESSING or ACTIVE. You will not be able to search uploaded documents until the domain status is ACTIVE.



4. On the **DOCUMENT SOURCE** step, select **Predefined data**, choose **IMDB movies (demo)**, and click **Continue**.

Amazon CloudSearch Developer Guide

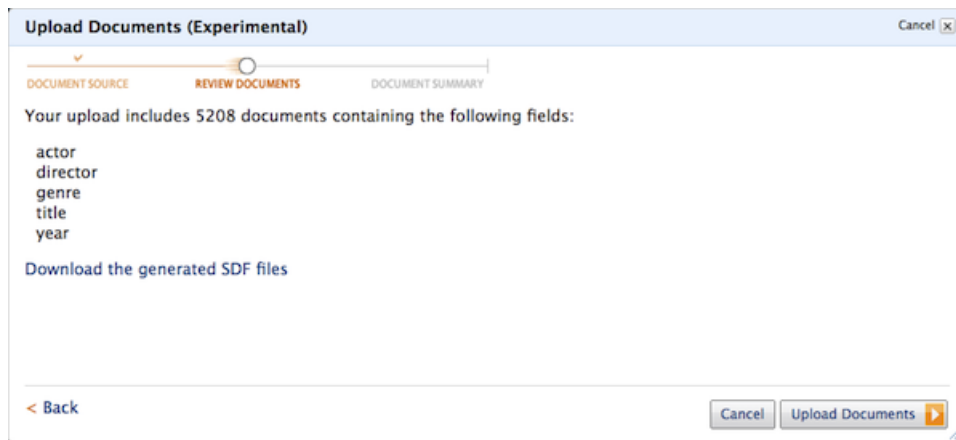
Step 3: Send Data for Indexing



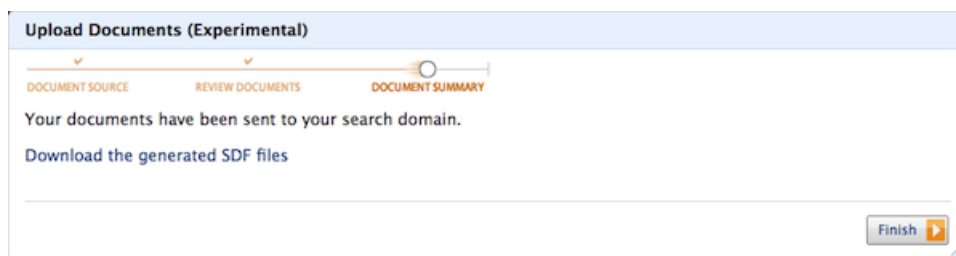
5. On the **REVIEW DOCUMENTS** step, review the upload summary and click **Upload Documents** to send the data to your domain for indexing.

Note

If you'd like to see what the SDF data looks like, click **Download the generated SDF files**. For more information about SDF and preparing your own data, see [Preparing Your Data for Amazon CloudSearch](#).



6. On the **DOCUMENT SUMMARY** step, click **Finish** to return to the domain dashboard.



That's it! You now have a fully functional Amazon CloudSearch domain that you can start searching. The data is automatically indexed in near real-time, so you can start searching your domain right away.

Step 4: Search Your Amazon CloudSearch Domain

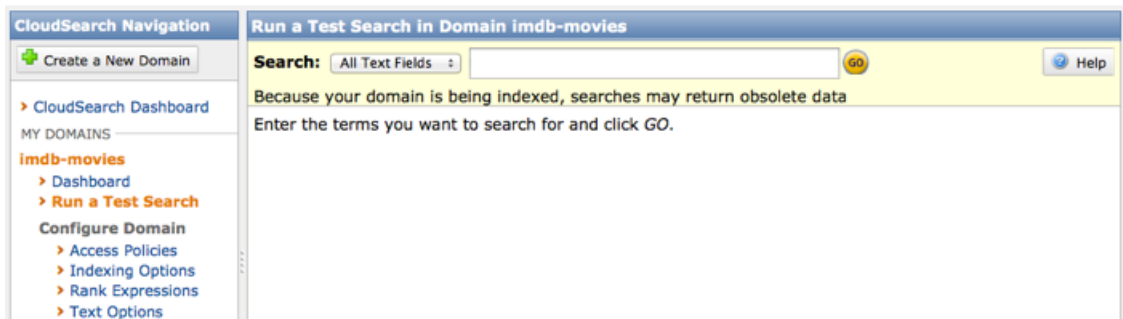
You can use the search tester in the Amazon CloudSearch console to perform simple text searches. To perform more complex Boolean queries, you can submit search requests through a Web browser or send HTTP requests using cURL or any HTTP library.

Searching with the Search Tester

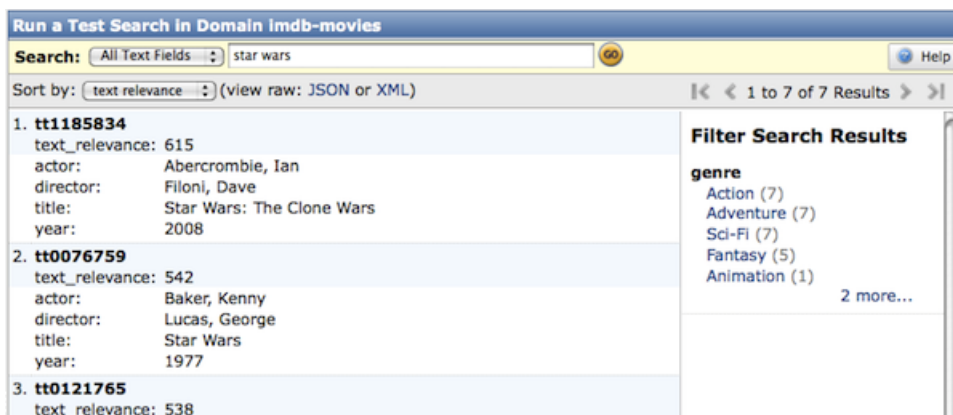
The search tester enables you to choose which fields you want to search, sort the results, and browse any facets that are configured for the domain. By default, results are sorted according to an automatically-generated relevance score, *text_relevance*. (For more information about customizing how results are ranked, see [Customizing Result Ranking with Amazon CloudSearch](#).)

To search your domain

1. Go to the Amazon CloudSearch console at [Amazon CloudSearch console](#).
2. In the **Navigation** panel, click the name of your movies domain.
3. In the **Navigation** panel, click the **Run a Test Search** link for your movies domain.



4. Select the field(s) you want to search, enter the text you want to search for, and click **Go**.



To view the HTTP search request that was sent to your domain's search endpoint and the JSON or XML response returned by Amazon CloudSearch, click the **view raw** link for the response format you want to see.

You can copy and paste the request URL to submit the request and view the response from a Web browser. Requests can be sent via HTTP or HTTPS.

Submitting Search Requests from a Web Browser

To perform more complex searches, you can submit your own search requests directly to your search endpoint. You can perform simple and Boolean searches and specify a variety of options to constrain your search, request facet information, customize ranking, and control what information is returned in the results.

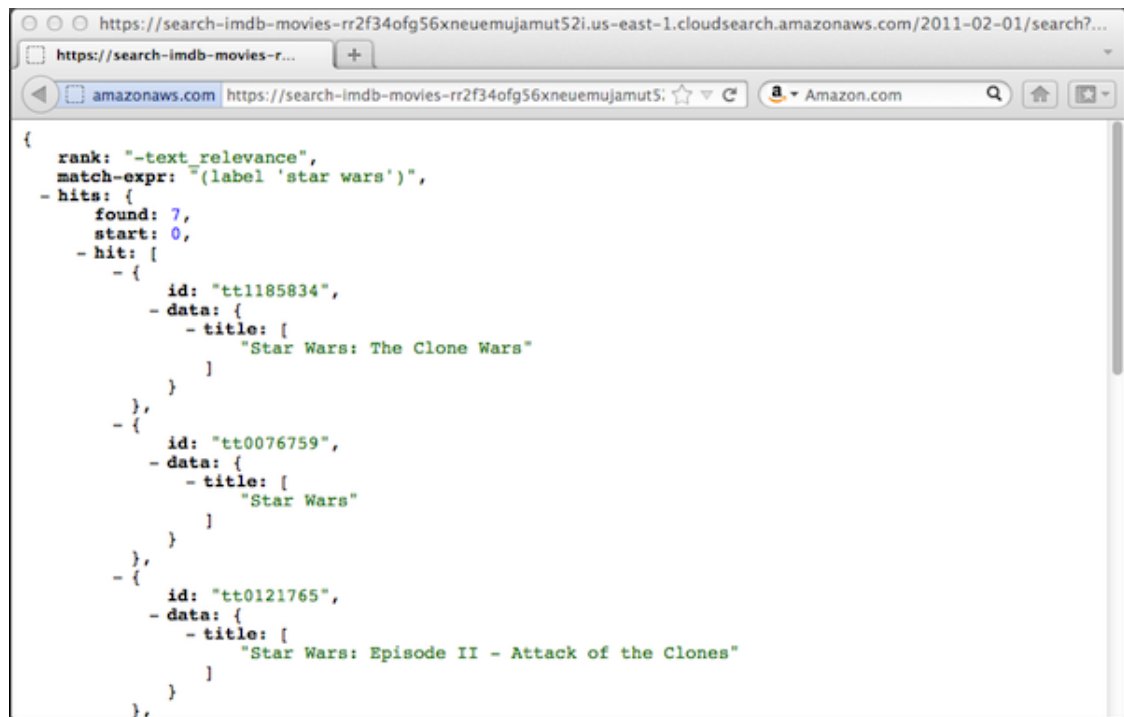
For example, to search your movies domain and get the titles of all of the available *Star Wars* movies, append the following search string to your search endpoint. (2011-02-01 is the API version and must be specified.)

```
/2011-02-01/search?q=star+wars&return-fields=title
```

Note

Your domain's search endpoint is shown on the domain dashboard. You can also perform a search from the AWS Management Console, view the raw request and response, and copy the request URL from the Search Request field.

By default, Amazon CloudSearch returns the response in JSON. You can also get the search results formatted in XML by specifying the `results-type=xml` parameter, `results-type=xml`. (Errors are always returned in JSON.) The following image shows the results of the previous query.



Filtering Results

You can use the Boolean query option, `bq`, to find documents that have particular numeric attributes. You can filter based on an exact value in a field, an inequality, or a range of values, as in these examples:

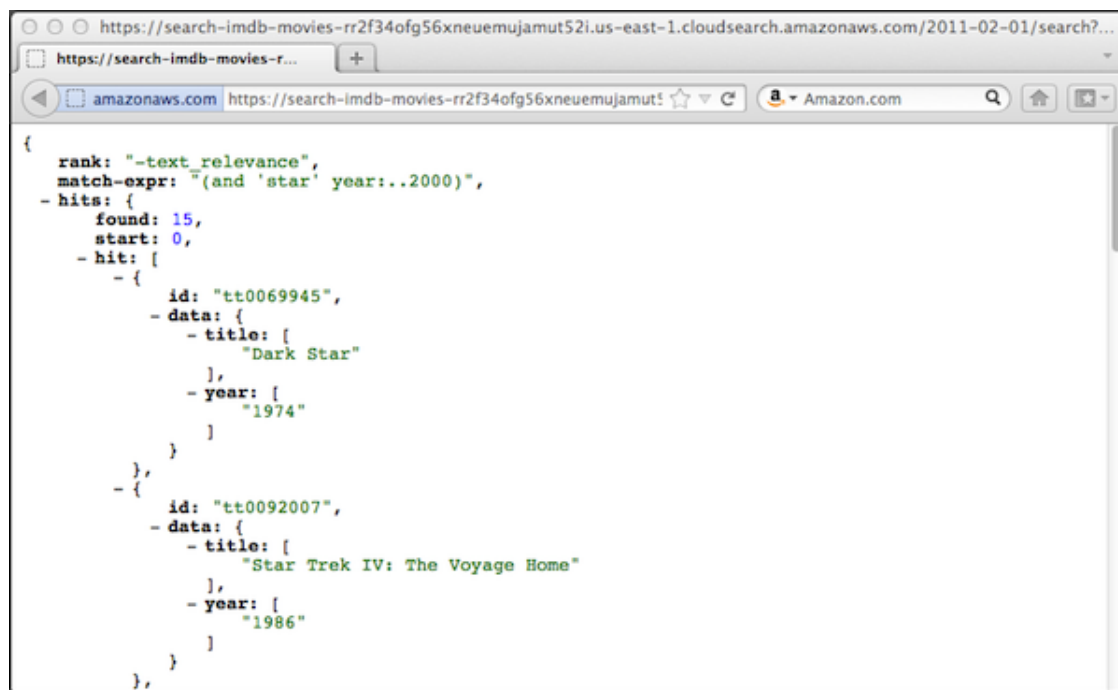
- `bq=year:2000` matches documents with the year 2000.

- `bq=year:2000..` matches documents with a year greater than or equal to 2000
- `bq=year:..2000` matches documents with a year less than or equal to 2000
- `bq=year:2000..2011` matches documents with a year between 2000 and 2011, inclusive.

For example, the following Boolean query searches for "star", finds all of the matching movies that were released before 2000, and returns title and year of each one:

```
2011-02-01/search?bq=(and 'star' year:..2000)&return-fields=title,year
```

The response shows the number of matching documents and the requested fields for each hit.



```
{
  rank: "-text_relevance",
  match-expr: "(and 'star' year:..2000)",
  - hits: {
    found: 15,
    start: 0,
    - hit: {
      - {
        id: "tt0069945",
        - data: {
          - title: [
            "Dark Star"
          ],
          - year: [
            "1974"
          ]
        }
      },
      - {
        id: "tt0092007",
        - data: {
          - title: [
            "Star Trek IV: The Voyage Home"
          ],
          - year: [
            "1986"
          ]
        }
      }
    }
  }
}
```

For more information about constructing search queries, see [Searching Your Data with Amazon CloudSearch](#).

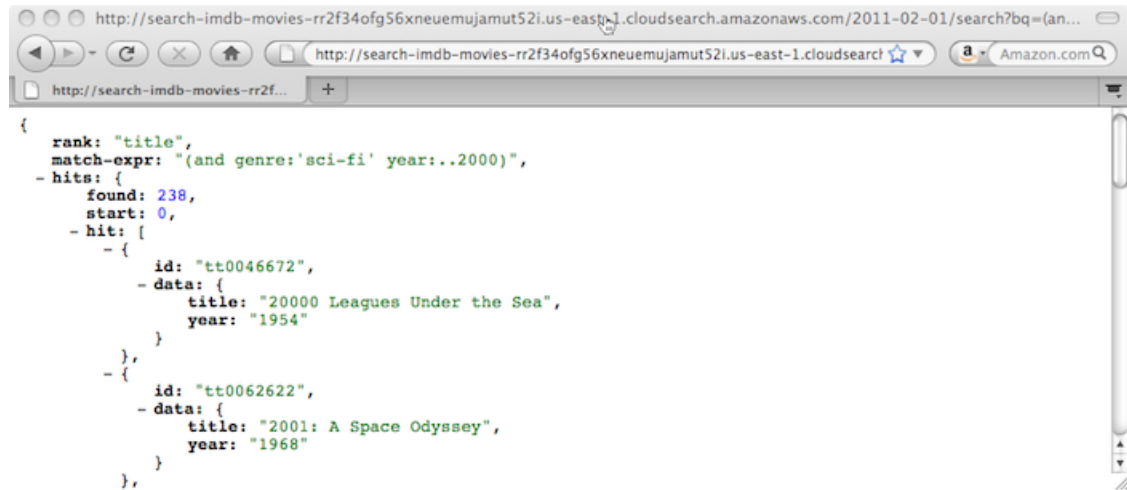
Ranking the Search Results

By default, Amazon CloudSearch ranks the search results according to an automatically generated `text_relevance` score. You can change how results are ranked by specifying the `rank` option in your search request to specify the field or rank expression you want to use for ranking. (A rank expression is a custom numeric expression that can be evaluated for each document in the set of matching documents. For information about defining your own rank expressions, see [Customizing Result Ranking with Amazon CloudSearch](#).)

If you specify a text field with the rank option, the results are sorted alphabetically according to that field. For example, to rank results from your movies domain alphabetically by title, add `&rank=title` to your query string:

```
2011-02-01/search?bq=(and genre:'sci-fi' year:..2000)&return-
fields=title,year&rank=title
```


When you rank alphabetically, the results are sorted in ascending order by default. Any values that begin with a numeral are listed before the first *A* entry:



```
{
  rank: "title",
  match-expr: "(and genre:'sci-fi' year:..2000)",
  - hits: {
    found: 238,
    start: 0,
    - hit: [
      - {
        id: "tt0046672",
        - data: {
          title: "2000 Leagues Under the Sea",
          year: "1954"
        }
      },
      - {
        id: "tt0062622",
        - data: {
          title: "2001: A Space Odyssey",
          year: "1968"
        }
      }
    ]
  }
}
```

Similarly, you can specify an integer field with the `rank` option to sort the results numerically.

By default, when you rank alphabetically or numerically, results are returned in ascending order. You can prefix the field name with a minus (-) if you want the results returned in descending order. If you specify a comma separated list of fields or rank expressions, the first field or expression is used as the primary sort criteria, the second is used as the secondary sort criteria, and so on.

For more information about ranking results, see [Customizing Result Ranking with Amazon CloudSearch](#)

Getting Facet Information

A facet is an index field that represents a category that you want to use to refine and filter search results. When you submit search requests to Amazon CloudSearch, you can request facet information to find out how many hits share the same value in a facet. You can display this information along with the search results and use it to enable users to interactively refine their searches. (This is often referred to as faceted navigation or faceted search.)

A facet can be any numeric field or a text or literal field that has faceting enabled in your domain configuration. To request facet information in your search request, you specify:

- One or more facets
- Facet constraints that specify the particular values you want to count (optional)
- How you want the facet values to be sorted in the results (optional)

For each facet, Amazon CloudSearch calculates the number of hits that share the same value. If you specify constraints, the facet counts are calculated only for values that match the constraints. Only constraints that have matches are included in the facet results.

Note

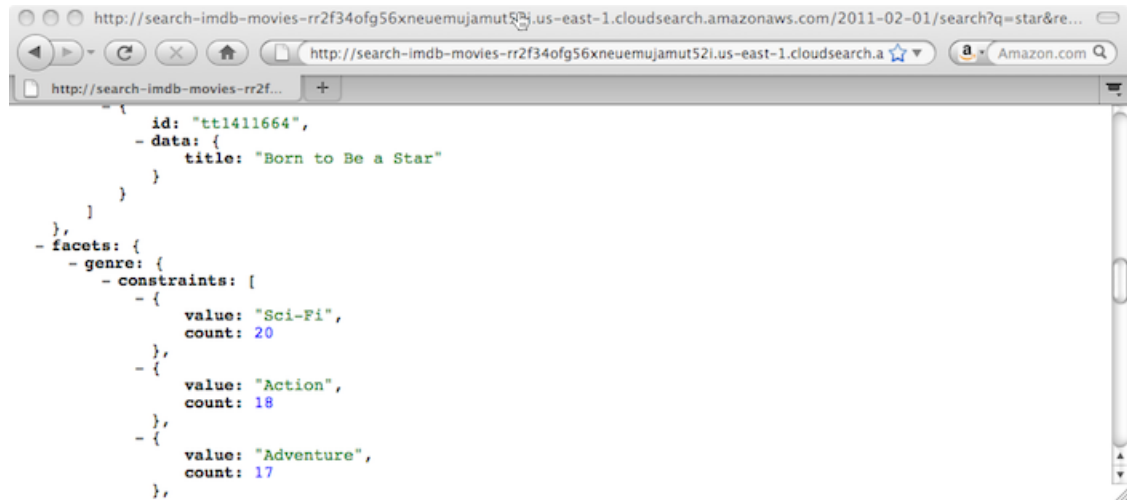
Values from a facet-enabled text or literal field cannot be returned in the search results. Text and literal fields can be facet-enabled or result-enabled, but not both. If you want to return the value from an SDF document field as well as use the field as a facet, create two index fields that use the same SDF document field as a source and make one result-enabled, and the other facet-enabled.

To get facet counts with your search results

- Use the *facet* option to specify the fields for which you want to compute facets. For the sample IMDB movies data faceting is enabled for one field, *genre*.

```
/2011-02-01/search?q=star&return-fields=title&facet=genre
```

The facets appear below the hits in the results.

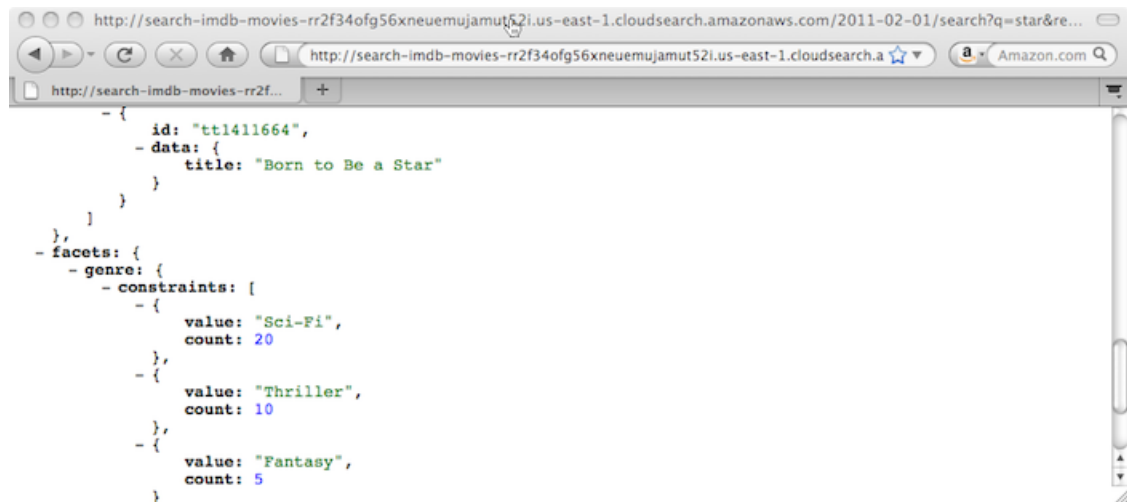


```
    {
      id: "tt1411664",
      - data: {
        title: "Born to Be a Star"
      }
    },
  ],
  - facets: {
    - genre: {
      - constraints: [
        - {
          value: "Sci-Fi",
          count: 20
        },
        - {
          value: "Action",
          count: 18
        },
        - {
          value: "Adventure",
          count: 17
        }
      ],
    }
  }
}
```

If you want to compute facet counts for selected values of a facet field, you can set facet constraints for the field. Facet constraints do not constrain the results themselves, only the facet counts that are returned. For example, the following request only counts the movies that are in the Sci-Fi, Fantasy, or Thriller genres:

```
/2011-02-01/search?q=star&return-fields=title&facet=genre&facet-genre-constraints='Sci-Fi', 'Fantasy', 'Thriller'
```

This constrains the facet counts to the three specified values:



```
    {
      id: "tt1411664",
      - data: {
        title: "Born to Be a Star"
      }
    },
  ],
  - facets: {
    - genre: {
      - constraints: [
        - {
          value: "Sci-Fi",
          count: 20
        },
        - {
          value: "Thriller",
          count: 10
        },
        - {
          value: "Fantasy",
          count: 5
        }
      ],
    }
  }
}
```

For more information about faceted searches, see [Getting and Using Facet Information in Amazon CloudSearch](#).

Step 5: Delete Your Amazon CloudSearch Movies Domain

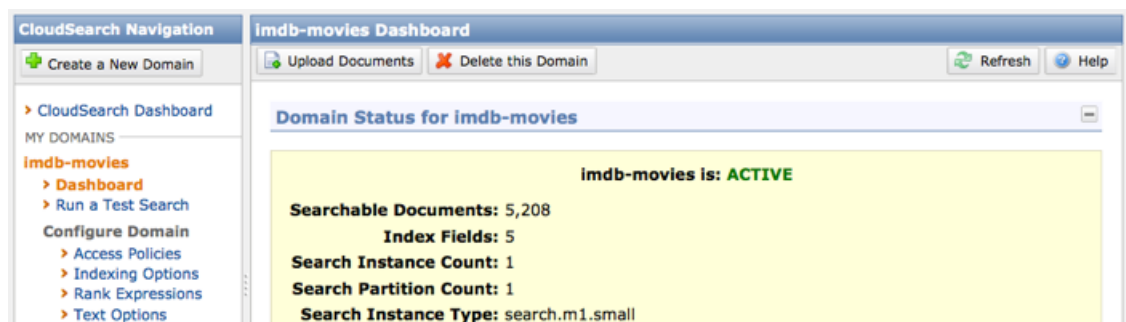
When you are finished experimenting with your movies domain, you need to delete it to avoid incurring additional usage fees.

Important

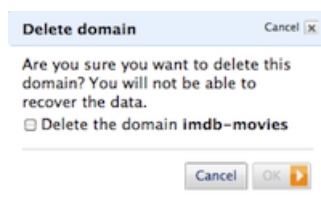
Deleting a domain deletes the index associated with the domain and takes the domain's document and search endpoints offline permanently.

To delete your imdb-movies domain

1. Go to the Amazon CloudSearch console at [Amazon CloudSearch console](#).
2. In the **Navigation** panel, click the name of your movies domain to view to the domain dashboard.
3. At the top of the domain dashboard, click the **Delete this Domain** button.



4. In the **Delete Domain** dialog box, select the **Delete the domain** option and click **OK** to permanently remove the domain and all of its data.



Note

It can take around 15 minutes to delete the domain and its resources. Until then, the domain status will be *BEING DELETED*.

Wondering where to go next? [What is Amazon CloudSearch](#) has a guide to the rest of the Amazon CloudSearch developer documentation. For more information about the Amazon CloudSearch query language, see [Searching Your Data with Amazon CloudSearch](#). If you're ready to set up a domain with your own data, see [Preparing Your Data for Amazon CloudSearch](#) and [Uploading Data to an Amazon CloudSearch Domain](#) for information about formatting and submitting your data to Amazon CloudSearch.

Making Amazon CloudSearch API Requests

Topics

- [Regions and Endpoints for Amazon CloudSearch \(p. 22\)](#)
- [Making Configuration Requests in Amazon CloudSearch \(p. 24\)](#)
- [Making Document Service Requests in Amazon CloudSearch \(p. 26\)](#)
- [Making Search Requests in Amazon CloudSearch \(p. 27\)](#)

This section describes how to make requests using the Amazon CloudSearch APIs. [Regions and Endpoints for Amazon CloudSearch \(p. 22\)](#) describes the endpoints you use to contact the Amazon CloudSearch configuration service, document service, and search service. The following sections describe how to submit requests to each of the services.

Regions and Endpoints for Amazon CloudSearch

Amazon CloudSearch provides separate endpoints for accessing the configuration, search, and document services.

The configuration service is accessed through a general, region-specific endpoint, `cloudsearch.region.amazonaws.com`. For example, `cloudsearch.us-east-1.amazonaws.com`. For a current list of supported regions and endpoints, see [Regions and Endpoints](#). To access the search and document services, you use Amazon CloudSearch domain-specific endpoints:

- `http://doc-domainname-domainid.us-east-1.cloudsearch.amazonaws.com`—the document service endpoint is used to submit documents to the domain for indexing.
- `http://search-domainname-domainid.us-east-1.cloudsearch.amazonaws.com`—the search endpoint is used to submit search requests to the domain.

You must specify the API version in every Amazon CloudSearch request. The current Amazon CloudSearch API version is 2011-02-01. For example:

```
http://search-movies-h2pc7ftfnsdlqh6pqqawbfrhu.us-east-1.cloudsearch.amazonaws.com/2011-02-01/search?q=star
```

Specifying a Search Domain's Region

To create, configure, and upload documents to search domains in regions other than the default region, you need to specify the region when using the Amazon CloudSearch command line tools, Configuration and Document Service APIs, and the AWS Management Console. When submitting search requests to a search domain, the region is part of the domain-specific search endpoint.

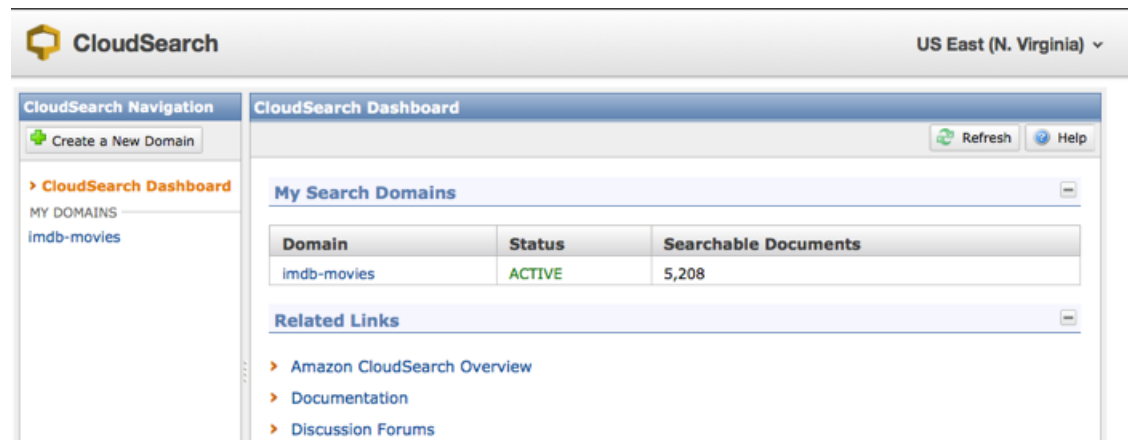
Command Line Tools

By default, the Amazon CloudSearch command line tools submit requests to the configuration service in the US East (Northern Virginia) Region: `cloudsearch.us-east-1.amazonaws.com`. To create and interact with search domains in other regions, you must specify the `--endpoint` parameter or set the `CS_ENDPOINT` environment variable. For example, to create a domain in the EU (Ireland) Region, specify the `eu-west-1` endpoint when you call `cs-create-domain`:

```
cs-create-domain -d myeuomain --endpoint cloudsearch.eu-west-1.amazonaws.com
```

AWS Management Console

The default region when you access the AWS Management console is the US West (Oregon) Region (`us-west-2`). To create and interact with search domains in other regions, use the Region selector in the top right corner of the page:



API

When using the Amazon CloudSearch configuration API directly, you submit requests directly to the endpoint for the region you want to use. For example, `cloudsearch.us-east-1.amazonaws.com`. Similarly, the search and document service endpoints for a particular search domain specify the region that the domain resides in. For example:

```
http://search-imdb-movies-6a15twom732paymbowxljtujhi.us-east-1.cloudsearch.amazonaws.com
http://doc-imdb-movies-6a15twom732paymbowxljtujhi.us-east-1.cloudsearch.amazonaws.com
```

Making Configuration Requests in Amazon CloudSearch

Amazon CloudSearch configuration requests are submitted using the AWS Query protocol. AWS Query requests are HTTP or HTTPS requests submitted via HTTP GET or POST with a query parameter named *Action*.

The endpoint for Amazon CloudSearch configuration requests is region-specific: `cloudsearch.region.amazonaws.com`. For example, `cloudsearch.us-east-1.amazonaws.com`. For a current list of supported regions and endpoints, see [Regions and Endpoints](#).

Important

The easiest way to submit configuration requests is to use the Amazon CloudSearch command line tools or the AWS SDK for Java, .NET, PHP, Ruby, and Python (Boto). The command line tools and SDKs handle the signing process for you and ensure that Amazon CloudSearch configuration requests are properly formed. For more information about using the command line tools, see [Amazon CloudSearch Command Line Tool Reference \(p. 136\)](#). For more information about the AWS SDKs, see [AWS Software Development Kits](#).

Structure of a Configuration Request

This guide shows Amazon CloudSearch configuration requests as URLs, which can be used directly in a browser. The URL contains three parts:

- Endpoint—the Web service entry point to act on, `cloudsearch.us-east-1.amazonaws.com`.
- Action—the Amazon CloudSearch configuration action you want to perform. For a complete list of actions, see [Actions \(p. 165\)](#).
- Parameters—any request parameters required for the specified action. Each query request must also include some common parameters to handle authentication. For more information, see [Request Authentication \(p. 25\)](#).

You must specify the `Version` parameter in every Amazon CloudSearch configuration request. The current Amazon CloudSearch API version is 2011-02-01.

For example, the following GET request creates a new search domain called *movies*:

```
https://cloudsearch.us-east-1.amazonaws.com
?Action=CreateDomain
&DomainName=movies
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120712/us-east-1/cloudsearch/aws4
_request
&X-Amz-Date=2012-07-12T21:41:29.094Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=c7600a00fea082dac002b247f9d6812f25195fbaf7f0a6fc4ce08a39666c6a10
3c8dcb
```

Note

Although the GET requests are shown as URLs, the parameter values are shown unencoded to make them easier to read. Keep in mind that you must URL encode parameter values when submitting requests.

Request Authentication

Requests submitted to the Configuration API are authenticated using your AWS access keys. You must include authorization parameters and a digital signature in every request. Amazon CloudSearch supports AWS Signature Version 4. For detailed signing instructions, see [Signature V4 Signing Process](#) in the AWS General Reference.

To create a signature for a request, you create a canonicalized version of the query string and compute an [RFC 2104-compliant](#) HMAC signature using a signing key derived from your AWS Secret Access key.

Note

If you are just getting started signing your own AWS requests, take a look at how the SDKs implement signing. The source for most of the AWS SDKs is available at <https://github.com/aws>. For example, [AWS4Signer.java](#) is the Signature V4 signer for the Java SDK.

For example, to construct a `CreateDomain` request, you need the following information:

```
Region name: us-east-1
Service name: cloudsearch
API version: 2011-02-01
Date: 2012-07-12T21:41:29.094Z
Access key: AKIAIOSFODNN7EXAMPLE
Secret key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Action: CreateDomain
Action Parameters: DomainName=movies
```

The canonical query string for a `CreateDomain` request looks like this:

```
Action=CreateDomain
&DomainName=movies
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120712/us-east-1/cloudsearch/aws4_request
&X-Amz-Date=2012-07-12T21:41:29.094Z
&X-Amz-SignedHeaders=host
```

The string to sign looks like this:

```
AWS4-HMAC-SHA256
20120712T214129Z
20120712/us-east-1/cloudsearch/aws4_request
a4cf362487306de739da7c697220a47373da10975702b0d9f80b6a6a7477df4a
```

The final signed request looks like this:

```
https://cloudsearch.us-east-1.amazonaws.com
?Action=CreateDomain
&DomainName=movies
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120712/us-east-1/cloudsearch/aws4_request
&X-Amz-Date=2012-07-12T21:41:29.094Z
```

```
&X-Amz-SignedHeaders=host  
&X-Amz-Signature=c7600a00fea082dac002b247f9d6812f25195fbaf7f0a6fc4ce08a39666c6a10
```

Making Document Service Requests in Amazon CloudSearch

The document service API is a REST-style API that has a single resource, `documents/batch`. You submit `documents/batch` requests to your domain's unique document service endpoint to add, update, or delete documents. A `documents/batch` request is an HTTP request, as defined by [RFC 2616](#). All `documents/batch` requests must be submitted using HTTP POST.

You must specify the Amazon CloudSearch API version in every document service request. The current Amazon CloudSearch API version is 2011-02-01. The version number precedes the resource name in the request, for example `POST /2011-02-01/documents/batch`.

Access Control

Access to a search domain's document service is restricted by IP address so that only authorized hosts can submit document changes. By default, your search domain will not accept document service requests from any IP addresses. You must authorize specific IP addresses or address ranges before you can submit documents through the command line tools or APIs. You can configure access policies from the Amazon CloudSearch console, using the `cs-configure-access-policies` command, or with the `UpdateServiceAccessPolicies` configuration action.

Request Headers

A `documents/batch` request must include the following headers:

- `Content-Length`—the length of the request body, in bytes.
- `Content-Type`—the type of data in the request body, `application/json` or `application/xml`.
- `Host`—your domain's document service endpoint.

By default, the response to a `documents/batch` request is returned in JSON. You can set the `Accept` header to `application/xml` if you want an XML response.

Request Body

The body of a `documents/batch` request contains a JSON or XML description of the document operations you want to perform. This description conforms to the Search Data Format (SDF). For more information about SDF, see [Preparing Your Data for Amazon CloudSearch \(p. 50\)](#).

Example Request

```
POST /2011-02-01/documents/batch HTTP/1.1  
Accept: application/json  
Content-Length: 1176  
Content-Type: application/json  
Host: doc.imdb-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com
```



```
[
  {
    "type": "add",
    "id": "tt0484562",
    "version": 1337648735,
    "lang": "en",
    "fields": {
      "title": "The Seeker: The Dark Is Rising",
      "director": "Cunningham, David L.",
      "genre": ["Adventure", "Drama", "Fantasy", "Thriller"],
      "actor": ["McShane, Ian", "Eccleston, Christopher", "Conroy, Frances"]
    }
  },
  {
    "type": "delete",
    "id": "tt0434409",
    "version": 1337648735
  }
]
```

Making Search Requests in Amazon CloudSearch

The search API is a REST-style API that has a single resource, `search`. You submit search requests to your domain's unique search service endpoint. Search requests are HTTP requests, as defined by [RFC 2616](#). Search requests must be submitted using HTTP GET.

You must specify the Amazon CloudSearch API version in every search request. The current Amazon CloudSearch API version is 2011-02-01. The version number precedes the resource name in the request, for example `GET /2011-02-01/search`.

Access Control

Access to a search domain's search service is restricted by IP address so that only authorized hosts can submit search requests. By default, your domain will not accept search requests from any IP addresses. You have to authorize specific IP addresses or address ranges before you can search the domain. You can do this from the Amazon CloudSearch console, through the `cs-configure-access-policies` command, or with the `UpdateServiceAccessPolicies` configuration action.

Request Headers

A search request must include the `HOST` header, which specifies your domain's search service endpoint.

Optionally, you can also specify the following headers:

- `Cache-Control`—forces the revalidation of results when a cached result document would otherwise be returned.

By default, the response to a search request is returned in JSON. You can set the `Accept` header to `application/xml` if you want an XML response.

Request Parameters

Search parameters are specified in the query string. For more information about constructing searches, see [Searching Your Data with Amazon CloudSearch \(p. 85\)](#).

Example Request

```
GET /2011-02-01/search?q=star+wars&return-fields=title HTTP/1.1
Host: search-imdb-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com
```

Creating an Amazon CloudSearch Domain

A search domain encapsulates your data, metadata, and index configuration options such as text processing options and ranking options. Each domain is searched separately. If you have multiple collections of data that you want to make searchable, you can create a separate search domain for each collection. When you create a new search domain, you must also [configure access policies \(p. 34\)](#), [configure indexing options \(p. 58\)](#), and [upload documents \(p. 77\)](#) before you can start [submitting search requests \(p. 85\)](#).

When you create a search domain, you must give it a unique name. Domain names must start with a letter or number and be at least 3 and no more than 28 characters long. The allowed characters are: a-z, 0-9, and hyphen (-). Upper case letters, underscores (_), and other special characters are not allowed in domain names.

You can choose the AWS region where you want to create your search domain. Typically, you should choose the region closest to your operations. For example, if you reside in Europe, create your search domain in the EU (Ireland) Region (eu-west-1). For a current list of supported regions and endpoints, see [Regions and Endpoints](#). For more information about choosing a region, see [Regions and Endpoints for Amazon CloudSearch \(p. 22\)](#).

Note

Amazon CloudSearch domains in different regions are entirely independent. For example, if you create a search domain called *my-domain* in us-east-1, and another domain called *my-domain* in eu-west-1, they are completely independent and do not share any data.

Each domain has unique endpoints through which you upload data for indexing and submit search requests. For example, the endpoints for a domain called imdb-movies might be:

```
search-imdb-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com
doc-imdb-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com
```

Important

By default, access to a new domain's document and search endpoints is blocked for all IP addresses. You must configure access policies for the domain to be able to submit search requests to the domain's search endpoint and upload data from the command line or through the domain's document endpoint. You can upload documents and search the domain through the Amazon CloudSearch console without configuring access policies.

You can create a search domain using the `cs-create-domain` (p. 30) command, from the [Amazon CloudSearch console](#) (p. 30), or using the `CreateDomain` (p. 33) configuration action.

Command Line Tools

You use the `cs-create-domain` (p. 150) command to create search domains. For information about installing and setting up the Amazon CloudSearch command line tools, see [Amazon CloudSearch Command Line Tool Reference](#) (p. 136).

To create a domain

- Run the `cs-create-domain` command and specify the name of the domain you want to create with the `--domain-name` option. For example, to create a domain called *movies*:

```
cs-create-domain --domain-name movies
=====
Creating domain [movies]
Domain endpoints are currently being created. Use cs-describe-domain
to check for endpoints.
```

It can take around half an hour to create endpoints for a new domain. By default, the `cs-create-domain` command returns immediately. If you specify the `--wait` option, the `cs-create-domain` command returns once your domain's endpoints are active.

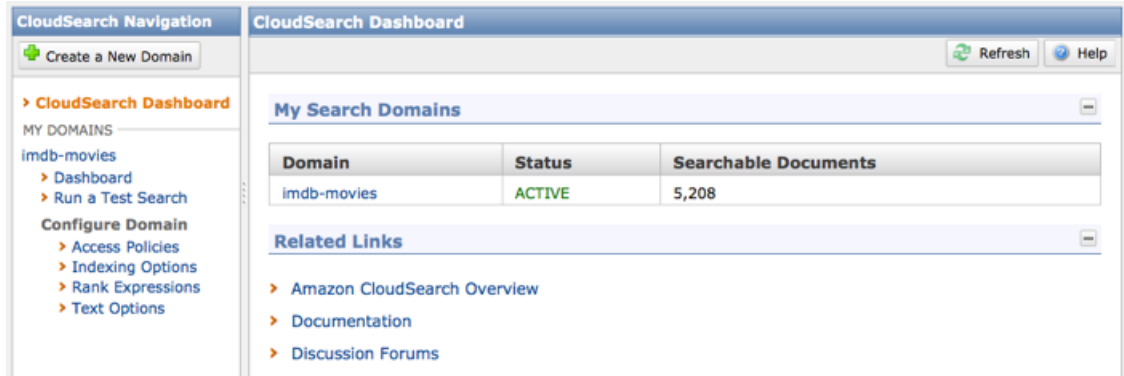
You can use the `cs-describe-domain` command to view a summary of the domain's status and configuration. For more information, see [Getting Information about an Amazon CloudSearch Domain](#) (p. 39).

AWS Management Console

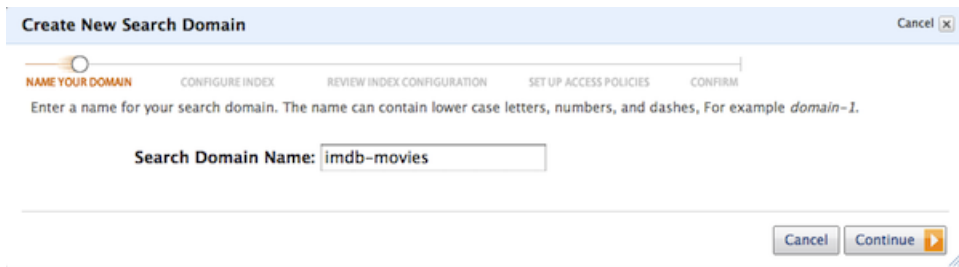
The Amazon CloudSearch console enables you to easily create new search domains and provides a variety of options for configuring indexing options, including [Cloning an Existing Domain's Indexing Options](#) (p. 64).

To create a domain

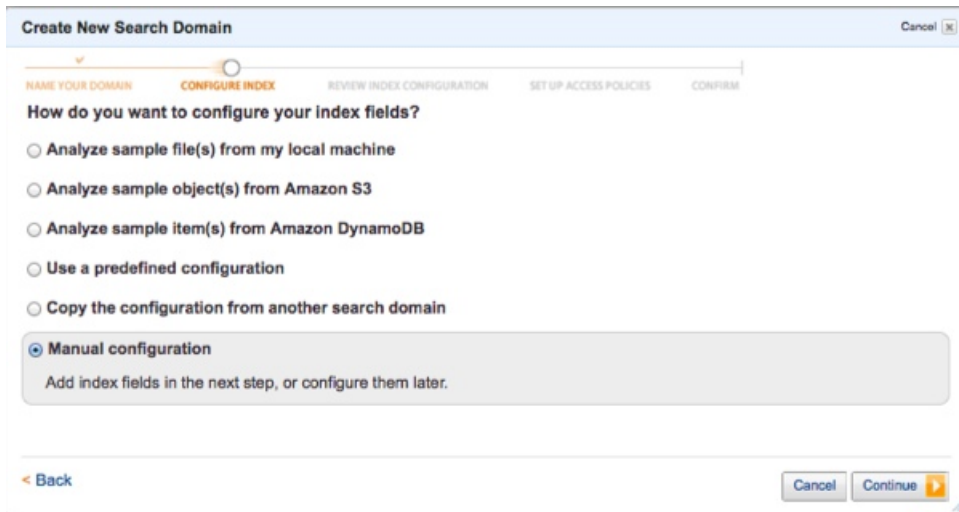
1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. At the top of the **Navigation** pane, click **Create a New Domain**. (If you are creating a domain for the first time, click **Create Your First Search Domain** on the Welcome page.)



3. In the **NAME YOUR DOMAIN** step, enter a name for your new domain and click **Continue**. Domain names must start with a letter or number and be at least 3 and no more than 28 characters long. Domain names can contain the following characters: a-z (lower case), 0-9, and - (hyphen). Upper case letters, underscores (_), and other special characters are not allowed in domain names.



4. In the **CONFIGURE INDEX** step, select **Manual Configuration** and click **Continue**. You can configure index fields and access policies when you first create the domain, or simply create a domain and configure it later. For more information about using the Amazon CloudSearch console to configure the domain, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#) and [Configuring Access for an Amazon CloudSearch Domain \(p. 34\)](#).



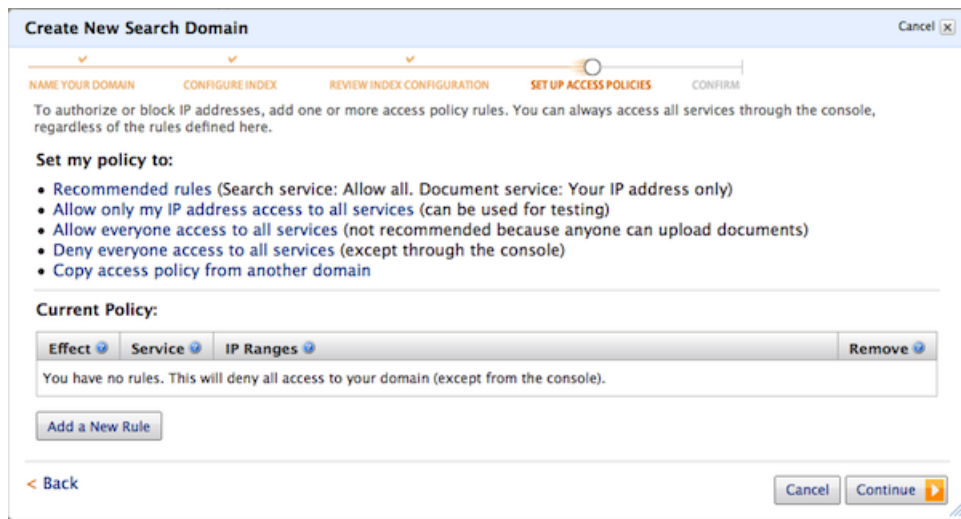
5. In the **REVIEW INDEX CONFIGURATION** step, click **Continue** to configure the index fields later. For more information about configuring index fields, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).



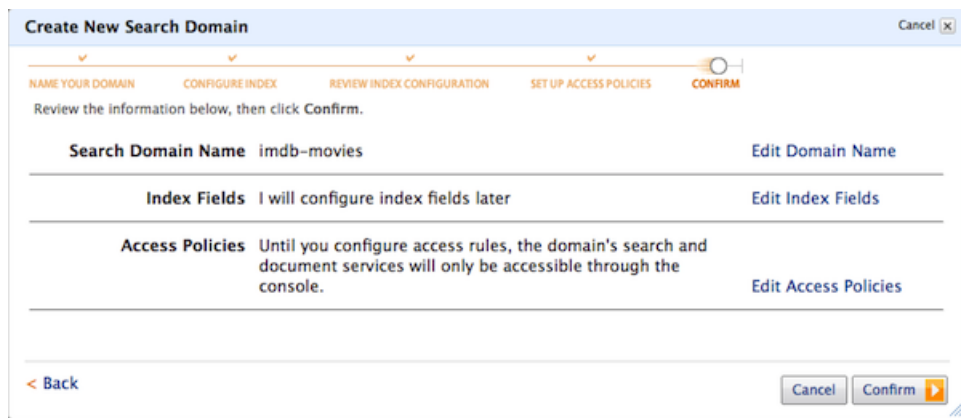
6. In the **SET UP ACCESS POLICIES** step, click **Continue** to set up access policies later. For more information about configuring access policies, see [Configuring Access for an Amazon CloudSearch Domain](#) (p. 34).

Note

If you don't configure access policies, you will only be able to upload documents and submit search queries through the console. By default, the Document and Search endpoints are configured to block all IP addresses.



7. In the **CONFIRM** step, review the domain configuration and click **Confirm** to create your domain.



8. Once the domain has been created, click **OK** to exit the Create New Search Domain wizard and go to the domain's dashboard.



API

You use the [CreateDomain](#) (p. 166) configuration action to create new domains. For example:

```
https://cloudsearch.us-east-1.amazonaws.com?Action=CreateDomain
&DomainName=movies
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120328/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-03-28T21:54:28.711Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=f5f82e71838707de1f72bfc42cc021e0324e1befa5df7c39c2ac25c61b3c8dcb
```

Note

Amazon CloudSearch configuration requests are authenticated using your access key ID and secret access key. For more information about signing requests, see [Request Authentication](#) (p. 25).

Configuring Access for an Amazon CloudSearch Domain

Access to a search domain is restricted by IP address so that only authorized hosts can submit documents and send search requests. You can authorize individual IP addresses or address ranges (subnets). IP addresses are specified in the standard Classless Inter-Domain Routing (CIDR) format, for example 10.24.34.0/24 specifies the range 10.24.34.0 - 10.24.34.255, while 10.24.34.0/32 specifies the single IP address 10.24.34.0. For more information about CIDR notation, see [RFC 4632](#). Amazon CloudSearch access policies are specified using the AWS Identity and Access Management (IAM) [Access Policy Language](#).

When you first create a search domain, it will not accept search or document service requests from any IP addresses. You must authorize specific IP addresses or address ranges before you can submit requests to your domain's endpoints through the command line tools or Amazon CloudSearch APIs. After you authorize one or more IP addresses, you can [upload documents \(p. 77\)](#) with the `cs-post-sdf` command or by sending requests to the `documents/batch` API. After you upload your documents, you can start [submitting search requests \(p. 85\)](#).

You can also define access policy rules to deny particular addresses and address ranges. Deny rules take precedence over Allow rules.

Note

IP address authorization is used only to control access to the document and search APIs. The Amazon CloudSearch configuration API uses standard AWS authentication.

If you don't know your computer's IP address, you can go to <http://www.whatsmyip.org/> to find out what it is. Keep in mind that if you do not have a static IP address, you must re-authorize your computer whenever your IP address changes. If your IP address is assigned dynamically, it is also likely that you're sharing that address with other computers on your network. This means that when you authorize the IP address, all computers that share it will be able to access your search domain's document and search endpoints.

Note

If you have made changes to your domain that require indexing, changes to the domain's access policies will not take effect until it is re-indexed. If re-indexing is needed, it will be indicated in the response to your update access policies request and shown on the domain dashboard in the console.

You can configure your access policies using the `cs-configure-access-policies` (p. 35) command, from the [Amazon CloudSearch console](#) (p. 36), or by uploading an IAM policy document with the `UpdateServiceAccessPolicies` (p. 37) configuration action.

Command Line Tools

You use the `cs-configure-access-policies` (p. 140) command to configure access to your domain's document and search endpoints. Access to each endpoint is configured separately. You can retrieve your domain's current policy document using the `cs-configure-access-policies` (p. 140) command by specifying the `--retrieve` option. For information about installing and setting up the Amazon CloudSearch command line tools, see [Amazon CloudSearch Command Line Tool Reference](#) (p. 136).

To configure access to your domain's document and search endpoints

1. Run the `cs-configure-access-policies` command in the `--update` mode to create a policy document that allows access to the document and search services. You specify the IP address or address range you want to authorize using the `--allow` option. To block specified addresses or address ranges, use the `--deny` option. You specify which service(s) you want to configure using the `--service` option to specify *doc*, *search*, or *all*. For example:

```
cs-configure-access-policies --domain-name movies --update --allow 192.0.2.0
--service all
=====
Standardizing ip: 192.0.2.0; using: 192.0.2.0/32

[movies] Updating access policy:

{"Version":"2011-10-11","Id":"34f11d91-88d9-4e15-8ebe-05dffef103c6",
"Statement":[{"Sid":"1","Effect":"Allow","Action":"*",
"Resource":"arn:aws:cs:us-east-1:598352442322:search/movies",
"Condition":{"IpAddress":{"aws:SourceIp":["192.0.2.0/32"]}}},
{"Sid":"2","Effect":"Allow","Action":"*",
"Resource":"arn:aws:cs:us-east-1:598352442322:doc/movies",
"Condition":{"IpAddress":{"aws:SourceIp":["192.0.2.0/32"]}}}]}
```

Note

The Action name in the policy document is always set to the wildcard character (*). There are no specific action names supported at this time.

2. When prompted, enter `y` to confirm that you want to update the access policies for your domain.

```
Really update access policies for [movies] y/N: y
Your access policy update may take a few minutes to complete and its state
will change to Active when complete. To check the state, use
cs-configure-access-policies --retrieve-policy --service all
```

The `--update` option merges the specified policy rules with the existing policy document and uploads the revised policy document to the domain.

AWS Management Console

The Amazon CloudSearch console enables you to easily add access policy rules to authorize or block particular IP addresses or address ranges. The console provides four shortcuts for defining access policy rules:

- **Recommended rules**—enables anyone to search your data, but only you will be able to add and delete documents. Your domain's search endpoint will be reachable from any IP address, but only you will have access to the document endpoint.
- **Allow only my IP address access to all services**—only you will be able to search your data and add and delete documents. Your domain's endpoints will not be reachable from any other IP address.
- **Allow everyone access to all services**—enables *anyone* to search your data and add and delete documents from your domain. Your domain's endpoints will be accessible from any IP address.
- **Deny everyone access to all services**—your domain's document and search endpoints will not be directly accessible. You can only upload documents or submit search requests through the Amazon CloudSearch console.

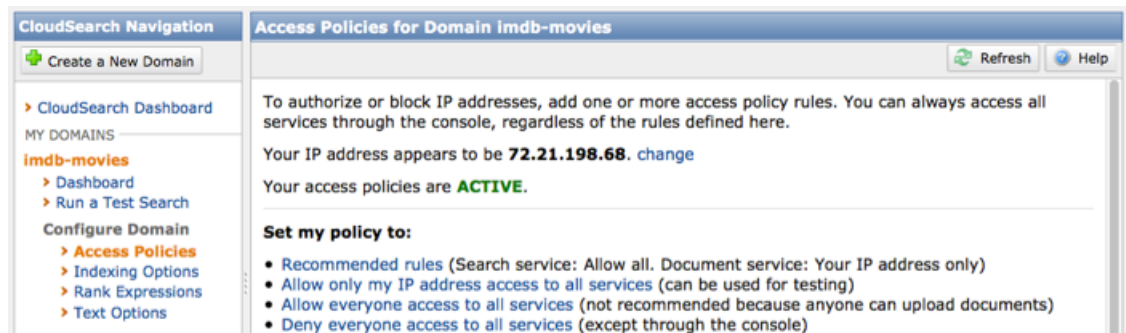
You can start with one of the shortcuts, and add additional rules to fine-tune access to your domain's endpoints. Deny rules take precedence over allow rules.

Note

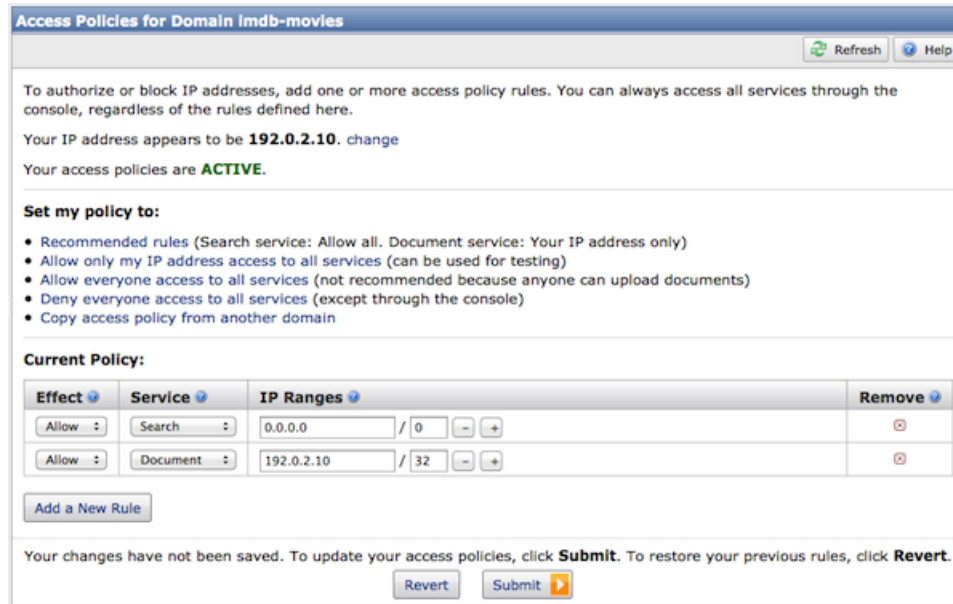
When you use the shortcuts, your IP address is automatically detected. If it's not correct or not the address you want to authorize, you can modify it before submitting your changes. You might need to work with your IT department to determine which IP addresses to authorize.

To add access policy rules

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain you want to configure, and then click the domain's **Access Policies** link.



3. In the domain's **Access Policies** pane, choose one of the shortcuts or enter the IP addresses you want to authorize or block. To add additional IP addresses or address ranges to the rule, click the add (+) icon in the **IP Ranges** column. To remove an address or range from the rule, click its delete (-) icon in the **IP Ranges** column. To add a new rule to the policy, click the **Add a New Rule** button. To remove a rule from the policy, click the remove (x) button in the **Remove** column.



- When you are done making changes to your access policy rules, click **Submit**. To exit without saving your changes, click **Revert**.

API

You use the [UpdateServiceAccessPolicies](#) (p. 187) configuration action to upload an IAM policy document that defines the access policies for your domain's document and search endpoints.

For example:

```
https://cloudsearch.us-east-1.amazonaws.com
?AccessPolicies={"Statement": [
{"Effect":"Allow", "Action": "*",
"Resource": "arn:aws:cs:us-east-1:360924696794:search/movies",
"Condition": { "IpAddress": { "aws:SourceIp": ["192.0.2.0/32"] } } },
{"Effect":"Allow", "Action": "*",
"Resource": "arn:aws:cs:us-east-1:360924696794:doc/movies",
"Condition": { "IpAddress": { "aws:SourceIp": ["192.0.2.0/32"] } } }
] }
&Action=UpdateServiceAccessPolicies
&DomainName=movies
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120330/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-03-30T19:27:45.110Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=801de749ab11a669925246f3d9454eee1dbc319f3352
3a4eb35a36ec93764e7d
```

Note

For readability, the request is shown without URL-encoding. Keep in mind that Amazon CloudSearch configuration requests must be URL-encoded. Amazon CloudSearch configuration

requests are authenticated using your access key ID and secret access key. For more information about signing requests, see [Request Authentication \(p. 25\)](#).

A policy document for Amazon CloudSearch contains a collection of statements that allow or deny access to the search and document service endpoints based on IP address. Note that the Action name is always set to the wildcard character (*). There are no specific action names supported at this time. You can retrieve your domain's current policy document with the [DescribeServiceAccessPolicies \(p. 180\)](#) action.

Access to each endpoint is configured separately. For example:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws:cs:us-east-1:123456789012:doc/movies",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "192.0.2.0/24"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws:cs:us-east-1:123456789012:search/movies",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "192.0.2.0/24"
        }
      }
    }
  ]
}
```

The Amazon Resource Name (ARN) for a domain's endpoints is of the form:

```
arn:aws:cs:us-east-1:awsaccountid:service/domain
```

The service can be either *doc* or *search*. The domain is the name of the domain for which you are configuring access. You can get a domain's ARNs with the `DescribeDomains` configuration action or the `cs-describe-domains` command. For more information, see [Getting Information about an Amazon CloudSearch Domain \(p. 39\)](#).

Monitoring Amazon CloudSearch Domains

Topics

- [Getting Information about an Amazon CloudSearch Domain \(p. 39\)](#)
- [Viewing Search Metrics for an Amazon CloudSearch Domain \(p. 44\)](#)
- [Tracking your Amazon CloudSearch Usage and Charges \(p. 45\)](#)

The AWS Management Console enables you to easily monitor your search domains. You can view configuration information for each domain, view and download search analytics data for a domain, and access detailed Amazon CloudSearch usage information. You can also get configuration information about particular domains through the command line tools and Configuration API.

Getting Information about an Amazon CloudSearch Domain

You can retrieve the following information about each of your search domains:

- **Domain Name**—The name of the domain.
- **Document Endpoint**—The endpoint through which you can submit document updates.
- **Search Endpoint**—The endpoint through which you can submit search requests.
- **Searchable Documents**—The number of documents that have been indexed.
- **Index Fields**—The name and type of each configured index field.
- **Rank Expressions**—The name and type of each rank expression.

When a domain is first created, the domain status will indicate that the domain is currently being activated and no other information will be available. Once your domain's document and search endpoints are available, the domain status will show the endpoint addresses that you can use to add data and submit search requests. If you haven't submitted any data for indexing, the number of searchable documents will be zero.

You can get domain information using the `cs-describe-domain` (p. 40) command, from the [Amazon CloudSearch console](#) (p. 41), or using the `DescribeDomains` (p. 43) configuration action.

This section also shows how you can view your domain's access policies and text options through the console. For information about accessing them through the command line tools or API, see [Configuring Access for an Amazon CloudSearch Domain](#) (p. 34) and [Configuring Text Options for an Amazon CloudSearch Domain](#) (p. 68).

Command Line Tools

You use the `cs-describe-domain` (p. 154) command to get information about your search domains. For information about installing and setting up the Amazon CloudSearch command line tools, see [Amazon CloudSearch Command Line Tool Reference](#) (p. 136).

To get domain information

- Run the `cs-describe-domain` command to get information about all of your domains. To get information about a specific domain, use the `--domain-name` option to specify the domain that you are interested in. To include information about all of the domain's configured index fields and rank expressions, specify the `--show-all` option. For example, the following request retrieves all of the available information about the domain named `movies`:

```
cs-describe-domain --domain-name movies --show-all

=== Domain Summary ===
Domain Name: movies
Document Service endpoint: doc-movies-h2pc7ftfnsdlqh6pqqawbftrhu.us-east-1.
cloudsearch.amazonaws.com
Search Service endpoint: search-movies-h2pc7ftfnsdlqh6pqqawbftrhu.us-east-1.
cloudsearch.amazonaws.com
SearchInstanceType: search.m1.small
SearchPartitionCount: 1
SearchInstanceCount: 1
Searchable Documents: 5208
Current configuration changes require a call to IndexDocuments: No

=== Domain Configuration ===

Access Policies:
=====
State: Active
{"Version":"2011-10-11","Id":"34f11d91-88d9-4e15-8ebe-05dffef103c6",
"Statement":[{"Sid":"1","Effect":"Allow","Action":"*",
"Resource":"arn:aws:cs:us-east-1:598352442322:search/movies",
"Condition":{"IpAddress":{"aws:SourceIp":["207.171.191.60/32"]}}},
{"Sid":"2","Effect":"Allow","Action":"*",
"Resource":"arn:aws:cs:us-east-1:598352442322:doc/movies",
"Condition":{"IpAddress":{"aws:SourceIp":["207.171.191.60/32"]}}}]

Fields:
=====
actor                Active                text (Result)
director             Active                text (Result)
genre                Active                literal (Search Facet)
title                Active                text (Result)
```

```
year                               Active                               uint ( )  
=====
```

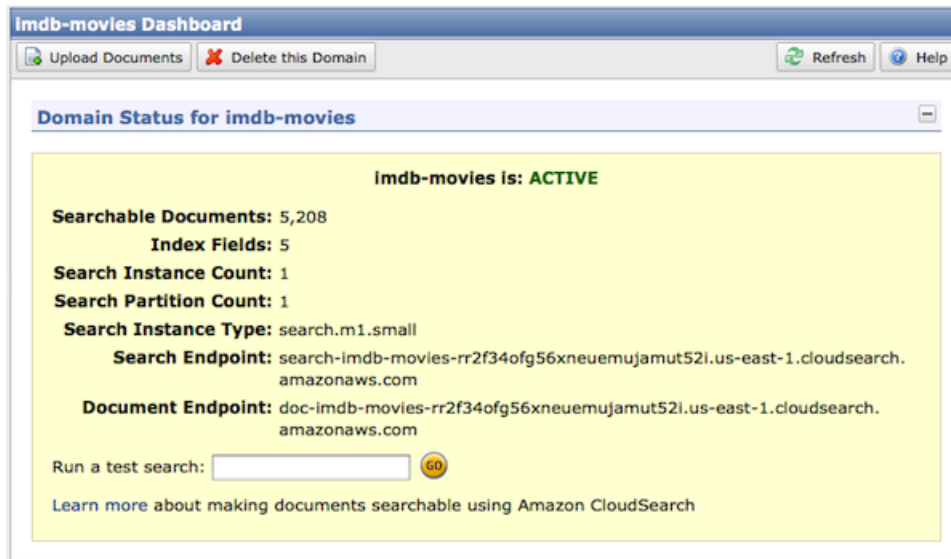
AWS Management Console

You can use the Amazon CloudSearch console to view information about all of your domains. The console's Amazon CloudSearch Dashboard shows a summary of all of the domains that you have created.



To view detailed information about a particular domain

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. Click the name of the domain in the **Navigation** pane. The Domain Dashboard shows the status summary for the selected domain.



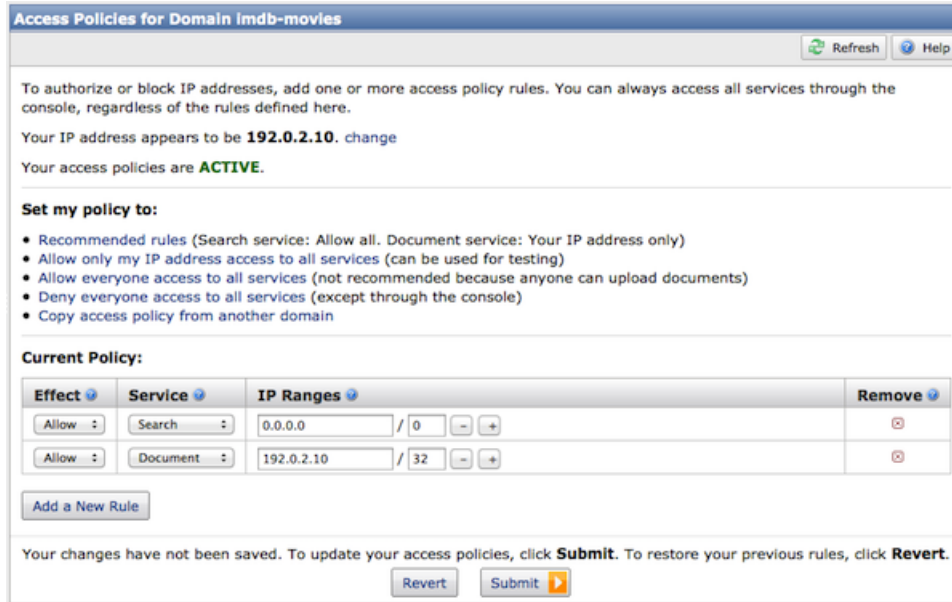
3. To view the index fields configured for the domain, click the domain's **Indexing Options** link in the **Navigation** pane. The following screen shot shows the index fields and the indexing options for the imdb-movies domain. For more information about index fields, see [Configuring Index Fields for an Amazon CloudSearch Domain](#) (p. 58).

Name	Status	Type	Search	Facet	Result	Default Value	Source	Remove
actor	Active	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		[add]	<input type="checkbox"/>
director	Active	text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		[add]	<input type="checkbox"/>
genre	Active	literal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		[add]	<input type="checkbox"/>
title	Active	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		[add]	<input type="checkbox"/>

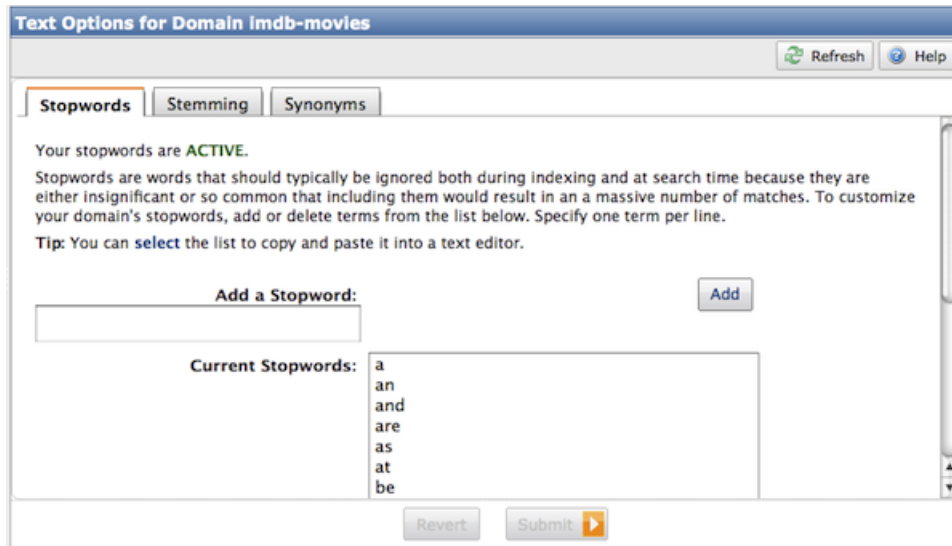
- To view the rank expressions configured for the domain, click the domain's **Rank Expressions** link in the **Navigation** pane. The following screen shot shows the rank expressions for the imdb-movies domain. For more information about rank expressions, see [Customizing Result Ranking with Amazon CloudSearch](#) (p. 105).

Name: myrank
Expression: ((0.3*popularity)/10.0)+((0.7*text_relevance))

- To view the access policies configured for the domain, click the domain's **Access Policies** link in the **Navigation** pane. The following screen shot shows the access policies for the imdb-movies domain. For more information about access policies, see [Configuring Access for an Amazon CloudSearch Domain](#) (p. 34).



6. To view the stopwords, synonyms, and stemming options configured for the domain, click the domain's **Text Options** link in the **Navigation** pane. The following screen shot shows the text options for the imdb-movies domain. For information about text options, see [Configuring Text Options for an Amazon CloudSearch Domain](#) (p. 68).



API

You use the [DescribeDomains](#) (p. 177) configuration action to get information about your domains. To get information about specific domains, specify the `DomainNames` parameter. For example, the following request shows how to get information about the `movies` and `imdb-movies` domains using the `DescribeDomains` configuration action:

```
https://cloudsearch.us-east-1.amazonaws.com
?Action=DescribeDomains
&DomainNames.member.1=movies
&DomainNames.member.2=imdb-movies
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120330/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-03-30T20:28:51.229Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=d8a4b3098bb37b73c48398db57315b272b92cbfcd
6b22ad1718c599b47466aea
```

If you omit the `DomainNames` parameter, `DescribeDomains` returns a summary of all your domains.

Note

Amazon CloudSearch configuration requests are authenticated using your access key ID and secret access key. For more information about signing requests, see [Request Authentication \(p. 25\)](#).

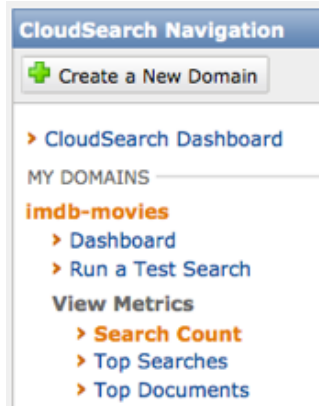
Viewing Search Metrics for an Amazon CloudSearch Domain

You can view Amazon CloudSearch analytics reports through the Amazon CloudSearch console and download the metrics data for further analysis. The following metrics are tracked for each search domain:

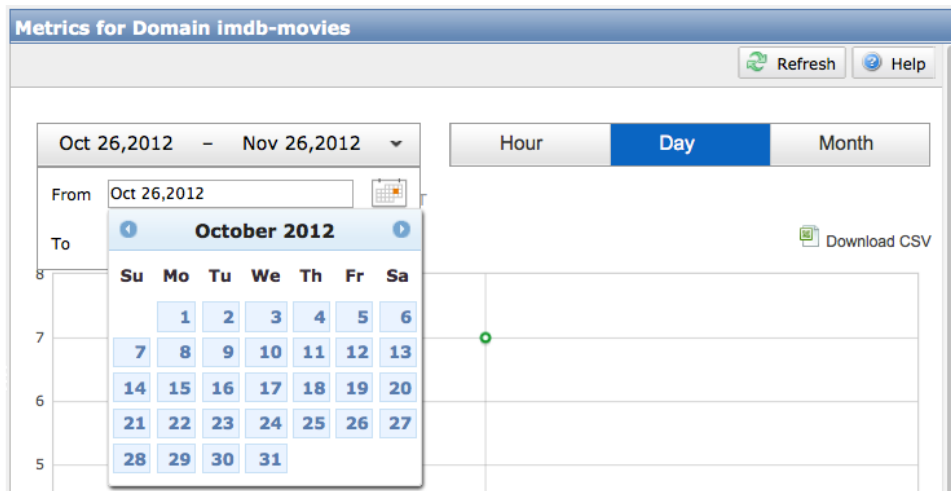
- **Total Searches**—The total number of searches. This is shown in the Search Count report.
- **Searches with No Results**—The number of searches for which no matching documents were found. This is shown in the Search Count report.
- **Top Searches**—The most frequent searches. This is shown in the Top Searches report.
- **Frequent Searches without Results**—The most frequent searches for which no matching documents were found. This is shown in the Top Searches report.
- **Top Documents**—The documents that were most frequently returned in search results. This is shown in the Top Documents report.

To view the analytics data for a search domain

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, select a domain.
3. Under **View Metrics**, select the analytics report that you want to view: **Search Count**, **Top Searches**, or **Top Documents**.



4. To view data for a different date or range of dates, hover over the date menu and select or enter the date(s) that you are interested in. Search count metrics are retained indefinitely. Top documents and Top searches metrics are updated hourly and retained for 13 months.



5. To download the data, click the **Download CSV** link.

Tracking your Amazon CloudSearch Usage and Charges

The AWS account activity page enables you to track your Amazon CloudSearch usage and charges.

To get your Amazon CloudSearch usage information

1. Go to aws.amazon.com and select **Account Activity** from the **My Account/Console** menu. (If you are not already logged in to the AWS portal, you will be prompted to enter your user name and password.)
2. Scroll down to the CloudSearch entry in the **Details** table and click **Download Usage Report**.

Details		
Expand All Services Collapse All Services		Printer Friendly Version
AWS Service Charges		\$44.80
+ Amazon Simple Storage Service		\$0.00
Download Usage Report »		
- Amazon CloudSearch		\$44.80
Download Usage Report »		
<i>November 1 - November 1, 2012</i>		
\$0.10 per search.m1.small instance-hour (or partial hour)	24 Hrs	2.40
<i>November 2 - November 30, 2012</i>		
\$0.10 per search.m1.small instance-hour (or partial hour)	424 Hrs	42.40

3. Specify the information you want to include in the report and click the download button for the data format you want to download. Reports can be downloaded as either XML or CSV.

Download Usage Report

Using the form below, you may create and download a report of your usage for the service you select:

Service:

Usage Types:

Operation:

Time Period:

Report Granularity:

Deleting an Amazon CloudSearch Domain

If you are no longer using a search domain, you must delete it to avoid incurring additional usage fees. Deleting a domain deletes the index associated with the domain and takes the domain's document and search endpoints offline permanently. It can take several minutes to completely remove a domain and all of its resources.

You can delete a domain using the `cs-delete-domain` (p. 47) command, from the [Amazon CloudSearch console](#) (p. 47), or using the `DeleteDomain` (p. 49) configuration action.

Command Line Tools

You use the `cs-delete-domain` (p. 153) command to delete a search domain and all of its resources. For information about installing and setting up the Amazon CloudSearch command line tools, see [Amazon CloudSearch Command Line Tool Reference](#) (p. 136).

To delete a domain

1. Run the `cs-delete-domain` command and specify the name of the domain you want to delete. For example, to delete the `movies` domain:

```
cs-delete-domain --domain-name movies
```

2. When prompted, enter `y` to confirm that you want to delete the domain.

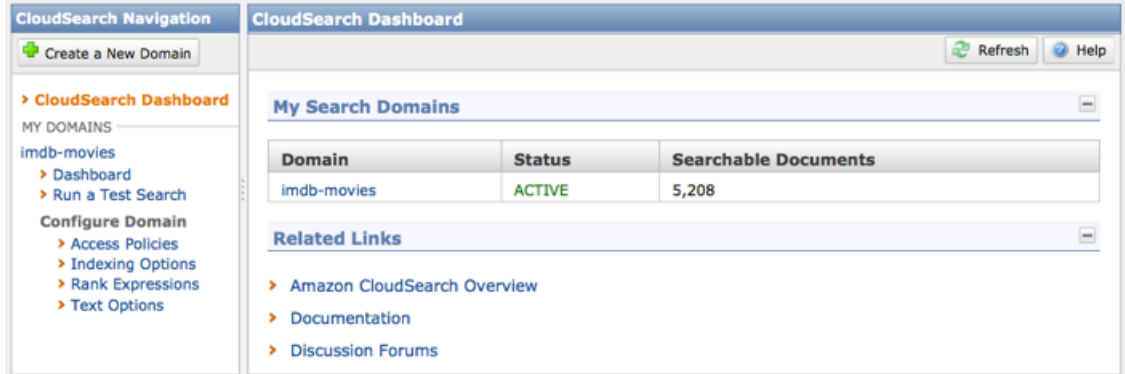
```
Really delete [movies] (y/N): y
```

AWS Management Console

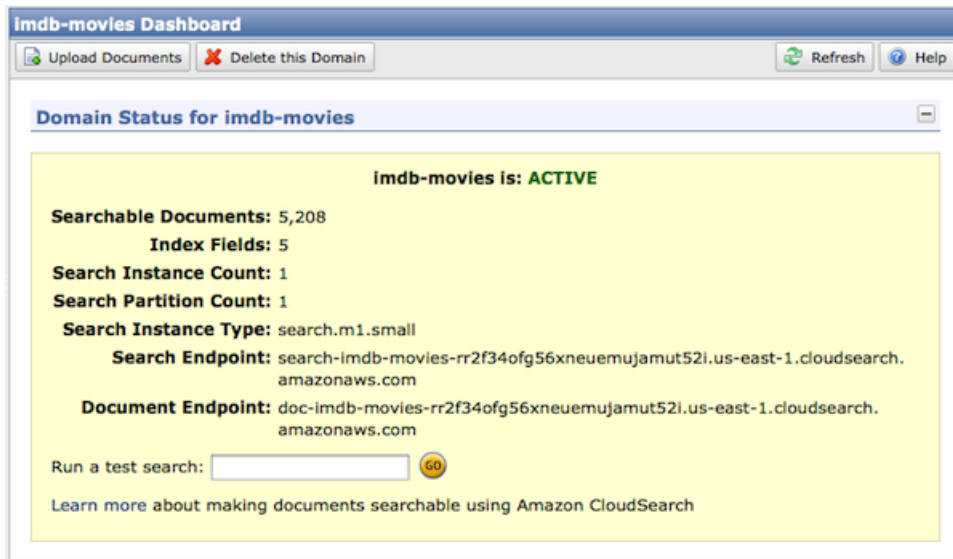
You can easily delete a domain from the domain dashboard in the Amazon CloudSearch console.

To delete a domain

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain you want to delete.



3. On the **Domain Dashboard**, click the **Delete this Domain** button.



4. In the **Delete Domain** dialog box, enable the checkbox and click **OK** to confirm that you want to delete the domain.



API

You use the [DeleteDomain \(p. 171\)](#) configuration action to remove a domain and all of its resources. The domain you want to delete is specified in the `DomainName` parameter. For example, to delete the *movies* domain:

```
https://cloudsearch.us-east-1.amazonaws.com
?Action=DeleteDomain
&DomainName=movies
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120330/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-03-30T20:39:24.716Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=d17fcb306c5466cba0264d911889b4408082767a68afc8635a613d9f
c6196a9f
```

Note

Amazon CloudSearch configuration requests are authenticated using your access key ID and secret access key. For more information about signing requests, see [Request Authentication \(p. 25\)](#).

Preparing Your Data for Amazon CloudSearch

Topics

- [Mapping Document Data to Index Fields in Amazon CloudSearch \(p. 50\)](#)
- [Creating SDF Batches in Amazon CloudSearch \(p. 51\)](#)

Before you can submit data to Amazon CloudSearch for indexing, you must describe it according to the Search Data Format (SDF). In SDF, each item that you want to be able to receive as a search result is represented as a document. Every document has a unique id (docid), a version number, and one or more fields that contain the data that you want to search and return in results. These document fields are used to populate the index fields you configure for your domain. For more information, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).

The [Mapping Document Data \(p. 50\)](#) section discusses how your source data relates to your domain's indexing options. [Creating SDF Batches \(p. 51\)](#) describes how to format your data in SDF.

For a detailed description of the SDF JSON and XML schemas, see the [Amazon CloudSearch Document Service API Reference \(p. 217\)](#).

Mapping Document Data to Index Fields in Amazon CloudSearch

You use the document fields defined in your SDF to populate the fields in your index. Any fields that occur in a document that are not used as a source for at least one index field are ignored and will not be searchable or returnable. Documents can contain a subset of the fields configured for the domain—every document does not have to contain all fields.

A document field can contain multiple values. When all sources are mapped to an index field, the total number of values in the index field cannot exceed 100. At search time, the document is returned as a hit if any of those values match the search query.

Document fields can contain alphanumeric or unsigned integer data. You can map unsigned integer data to a uint index field and use it to construct rank expressions and enable searches within a range of values.

You can map alphanumeric data to either text fields or literal fields. A document field can contain up to 1 MB of data.

A *uint* field contains a 32-bit unsigned integer. If you're mapping timestamps to a uint field, you have to strip off the milliseconds or the timestamp will overflow the uint field. Uint fields are always searchable and can always be returned in search results and used as facets.

A *text* field contains arbitrary alphanumeric data such as a name, description, or even the entire body of a document. Text fields are always searchable. They are tokenized during indexing and Amazon CloudSearch performs additional text processing on them according to the stopwords, synonyms, and stems you configure in your domain's text options. The contents of a text field can also be returned in search results or the field can be used as a facet, but not both. Amazon CloudSearch can return up to 2 KB of data from a text field—if the field contents exceed 2 KB, only the first 2 KB is included in the results. If a search request does not specify what field to search, by default Amazon CloudSearch searches all text fields. You can control what fields are searched by default by defining your own default search field. For more information, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).

A *literal* field contains an identifier or other data that you want to be able to match exactly. Unlike text fields, they are not tokenized—Amazon CloudSearch does not perform any text processing on literal fields. Literal fields can be used for fields that have a small set of possible values, as well as for more arbitrary values like email addresses or brand names where an exact match is important. Literal fields are frequently used to enable faceted searches where you want to count the number of exact matches for a particular value.

Creating SDF Batches in Amazon CloudSearch

You create SDF batches to describe the data that you want to make searchable. When you send SDF batches to a domain, the data is indexed automatically according to the domain's indexing and text options.

An SDF batch is a collection of add and delete operations that represent the documents you want to add, update, or delete from your domain. SDF batches can be described in either JSON or XML. The maximum batch size is 5 MB. The maximum size of an individual document is 1 MB. If you have a large number of documents, you must send updates in 5 MB batches.

Important

Whenever possible, you should group add and delete operations in batches that are close to the maximum batch size. Submitting a large volume of single-document batches to the document service can increase the time it takes for your changes to become visible in search results.

For each document in a batch, you must specify:

- The operation you want to perform: *add* or *delete*.
- A unique ID for the document (docid). A document ID can contain the following characters: a-z (lower-case letters), 0-9, and _ (underscore). Document IDs cannot begin with an underscore.
- A document version number for the add or delete operation. The version is used to guarantee that older updates aren't accidentally applied, and to provide control over the ordering of concurrent updates to the service. The document service guarantees that the update with the highest version will be applied and remain there until an add or delete operation with a higher version number and the same document ID is received. If you submit multiple add or delete operations with the same version number, which one takes precedence is undefined. You must increase the version number every time you submit a new add or delete operation for a document. For more information, see [Document Versions in Amazon CloudSearch \(p. 53\)](#).
- The document language as a two-letter language code, such as *en* for English. (Add operations only.)
- A name-value pair for each document field. When specifying SDF in JSON, the value for a field cannot be `null`. (Add operations only.)

For example, the following JSON SDF batch adds one document and deletes one document:

```
[
  {
    "type": "add",
    "id": "tt0484562",
    "version": 1,
    "lang": "en",
    "fields": {
      "title": "The Seeker: The Dark Is Rising",
      "director": "Cunningham, David L.",
      "genre": ["Adventure", "Drama", "Fantasy", "Thriller"],
      "actor": ["McShane, Ian", "Eccleston, Christopher", "Conroy, Frances",
        "Crewson, Wendy", "Ludwig, Alexander", "Cosmo, James",
        "Warner, Amelia", "Hickey, John Benjamin", "Piddock, Jim",
        "Lockhart, Emma"]
    }
  },
  {
    "type": "delete",
    "id": "tt0484575",
    "version": 2
  }
]
```

The same batch formatted in XML looks like this:

```
<batch>
  <add id="tt0484562" version="1" lang="en">
    <field name="title">The Seeker: The Dark Is Rising</field>
    <field name="director">Cunningham, David L.</field>
    <field name="genre">Adventure</field>
    <field name="genre">Drama</field>
    <field name="genre">Fantasy</field>
    <field name="genre">Thriller</field>
    <field name="actor">McShane, Ian</field>
    <field name="actor">Eccleston, Christopher</field>
    <field name="actor">Conroy, Frances</field>
    <field name="actor">Ludwig, Alexander</field>
    <field name="actor">Crewson, Wendy</field>
    <field name="actor">Warner, Amelia</field>
    <field name="actor">Cosmo, James</field>
    <field name="actor">Hickey, John Benjamin</field>
    <field name="actor">Piddock, Jim</field>
    <field name="actor">Lockhart, Emma</field>
  </add>
  <delete id="tt0484575" version="2" />
</batch>
```

Uploading SDF batches that contain invalid JSON or XML will produce unpredictable results. Processing stops when an error is encountered, but the preceding add and delete operations are applied to the domain. You can verify the validity of your JSON or XML data using tools such as `xmllint` and `jsonlint`.

Both JSON and XML batches can only contain UTF-8 characters that are valid in XML. Valid characters are the control characters tab (0009), carriage return (000D), and line feed (000A), and the legal characters of Unicode and ISO/IEC 10646. FFFE, FFFF, and the surrogate blocks D800–DBFF and DC00–DFFF are invalid and will cause errors. (For more information, see [Extensible Markup Language \(XML\) 1.0 \(Fifth Edition\)](#).) You can use the following regular expression to match invalid characters so you can remove them: `/[^\u0009\u000a\u000d\u0020-\u007F\uE000-\uFFFF]/` .

When formatting SDF in JSON, quotes (") and backslashes (\) within field values must be escaped with a backslash. For example:

```
"title": "Where the Wild Things Are"
"isbn": "0-06-025492-0"
"image": "images\\covers\\Where_The_Wild_Things_Are_(book)_cover.jpg"
"comment": "Sendak's \"Where the Wild Things Are\" is a children's classic."
```

When formatting SDF in XML, ampersands (&) and less-than symbols (<) within field values need to be represented with the corresponding entity references (& and <).

For example:

```
<field name="title">Little Cow &amp; the Turtle</field>
<field name="isbn">0-84466-4774</field>
<field name="image">images\\covers\\Little_Cow_&amp;_the_Turtle_(book)_cover.jpg</field>
<field name="comment">&lt;insert comment></field>
```

If you have large blocks of user-generated content, you might want to wrap the entire field in a CDATA section, rather than replacing every occurrence with the entity reference. For example:

```
<field name="comment">&lt;!CDATA[Monsters & mayhem--what's not to like! ]>
```

The command line tools and Amazon CloudSearch console include an experimental mechanism for automatically generating SDF from a variety of source documents.

Document Versions in Amazon CloudSearch

In some cases, you might have multiple processes submitting batches of documents to your domain. Because it's possible that these batches could contain changes to the same document, and that the batches might be received out of order, Amazon CloudSearch always applies the add or delete operation that has the *highest* version number. If the version number specified for a document is *lower* than in a previously-received operation, the change is ignored. If the version number specified for a document is the same as in a previously-received operation, the results are undefined. There's no way to predict which operation will take precedence.

Important

To update or delete an existing document, you *must* specify a new, larger version number. You do **not** specify the version that you want to update or delete.

To successfully update and delete documents, you have to keep track of the version numbers. One common approach is to use timestamps for versioning. Keep in mind that the version is stored as a 32-bit unsigned integer. If you're versioning with timestamps, you have to strip off the milliseconds or the timestamp will overflow. If you want to be able to query the index for your documents' version numbers, create a version field and populate it with the current version each time you update a document.

When deleting documents, note that deleting version `max(uint32_t)` will *permanently* remove the document from your domain. Because it is not possible to specify a higher version number, there is no way to add a later version of the document.

Adding and Updating Documents in Amazon CloudSearch

An add operation specifies either a new document that you want to add to the index or an existing document that you want to update.

When you add or update a document, you specify the document's ID, a new version number, the document language, and all of the fields the document contains. If a document field matches the name of an index field, it is automatically used as the source for that index field. You can also explicitly map one or more document fields to an index field. You don't have to specify every configured field for every document—documents can contain a subset of the configured fields. Document fields that are not used as a source for at least one index field are ignored during indexing.

Important

You *must* specify a new, larger version number every time you add or update a document. For more information, see [Document Versions in Amazon CloudSearch \(p. 53\)](#).

To add a document to a search domain

1. Specify an add operation in your SDF data that contains the id of the document you want to add and each of the fields that you want to be able to search or return in results. If you are updating an existing document, the version number must be greater than the document's current version number for the update to be applied. For example, the following operation would add version 1 of document tt0484562:

```
[
  {
    "type": "add",
    "id": "tt0484562",
    "version": 1,
    "lang": "en",
    "fields": {
      "title": "The Seeker: The Dark Is Rising",
      "director": "Cunningham, David L.",
      "genre": ["Adventure", "Drama", "Fantasy", "Thriller"],
      "actor": ["McShane, Ian", "Eccleston, Christopher", "Conroy, Frances",
        "Crewson, Wendy", "Ludwig, Alexander", "Cosmo, James",
        "Warner, Amelia", "Hickey, John Benjamin", "Piddock, Jim",
        "Lockhart, Emma"]
    }
  }
]
```

2. Send the SDF data to your domain. You can submit data updates through the Amazon CloudSearch console, using the `cs-post-sdf` command, or by posting a request directly to the domain's document service endpoint. For more information, see [Uploading Data to an Amazon CloudSearch Domain \(p. 77\)](#).

Deleting Documents in Amazon CloudSearch

A delete operation specifies a document that you want to remove from a domain's index. Once a document is deleted, it will no longer be searchable or returned in results.

Important

When deleting a document, you *must* specify a new, larger version number. You do **not** specify the version that you want to delete. For more information, see [Document Versions in Amazon CloudSearch \(p. 53\)](#).

When posting SDF updates to delete documents, you have to specify each document that you want to delete. If you want to start over with an empty domain that has the same configuration, you can use the console to clone the domain. For more information, see [Cloning an Existing Domain's Indexing Options](#) (p. 64).

If your domain has scaled up to accommodate your index size and you delete a large number of documents, the domain scales down the next time the full index is rebuilt. While this is periodically done automatically, to scale down as quickly as possible you can explicitly [rebuild the index](#) (p. 82) when you are done deleting documents.

Note

To delete documents, you upload SDF batches that contain delete operations. You are billed for the total number of document batches uploaded to your search domain, including batches that contain delete operations. For more information about Amazon CloudSearch pricing, see aws.amazon.com/cloudsearch/pricing/.

To delete a document from a search domain

1. Specify a delete operation in your SDF data that contains the id of the document you want to remove and an updated document version number. The version number must be greater than the document's current version number for the document to be deleted. For example, the following operation would remove version 1 of document tt0484575:

```
[
  {
    "type": "delete",
    "id": "tt0484575",
    "version": 2
  }
]
```

2. Send the SDF data to your domain. You can submit data updates through the Amazon CloudSearch console, using the `cs-post-sdf` command, or by posting a request directly to the domain's document service endpoint. For more information, see [Uploading Data to an Amazon CloudSearch Domain](#) (p. 77).

Generating SDF from Your Source Data in Amazon CloudSearch (Experimental)

The command line tools and Amazon CloudSearch console include an experimental mechanism for automatically generating SDF batches from several common file types: PDF, Microsoft Excel, Microsoft PowerPoint, Microsoft Word, CSV, text, HTML, JSON, and XML.

For most file types, including JSON and XML, Amazon CloudSearch adds a single add document operation to the SDF batch for each source file. If metadata is available for the file, the metadata is mapped to corresponding document fields—the fields generated from the document metadata vary depending on the file type. The contents of the source file are parsed into a single text field. If the file contains more than 1 MB of data, the data mapped to the text field is truncated so that the document does not exceed 1 MB.

CSV files are handled differently. When you generate SDF from a CSV file, Amazon CloudSearch uses the contents of the first row to define the document fields, and creates a separate document for each following row. If there is a column header called *docid*, the values in that column are used as the document IDs. If necessary, the docid values are normalized to conform to the allowed character set: a-z (lower-case letters), 0-9, and _ (underscore). If there is no docid column, a unique ID is generated for each document based on the filename and row number. Similarly, if there is a column called *version*, the values in that

column are used as the versions for the document updates. Version numbers must be specified as 32-bit unsigned integers. If there is no version column, version numbers are generated based on a timestamp.

If you upload multiple types of files, CSV files are parsed row-by-row, and non-CSV files are treated as individual documents.

Note

Currently, only CSV files are parsed to automatically extract custom field data and generate multiple documents. When processing XML and JSON files, each file is treated as a single document and the contents of the file are used to populate a single text field.

You can also generate SDF directly from data stored in DynamoDB. When processing DynamoDB data, a separate SDF document is created for each item read from the table.

Command Line Tools

You can use the `cs-generate-sdf` command to automatically generate SDF from local files or data stored in Amazon S3 or DynamoDB. To process multiple files stored locally or in Amazon S3, you can specify multiple `--source` options, or use wildcards in the path or Amazon S3 URI that specifies the location of your source data.

To generate SDF

- Run the `cs-generate-sdf` command. You must specify the `--source` option and either the `--output` option or the `--domain` option. If you are reading data from your local file system or S3, you can use the `--modified-after` option to restrict processing to files or Amazon S3 objects modified after a particular time. If you are reading data from an DynamoDB table, you can specify a start key to read from a particular point in the table. For more information about specifying options to control how data is read and processed, see [cs-generate-sdf \(p. 158\)](#).

```
cs-generate-sdf --source c:\myAmazingDataSet\*
--modified-after 2012-03-28T00:00
--output c:\myAmazingDataSet\SDF
```

To generate SDF from CSV data

- Run the `cs-generate-sdf` command. By default, when processing CSV files, each row will be parsed as a separate document. (If you specify the `--single-doc-per-csv` option when processing CSV files, each file will be treated as a single document.)

```
cs-generate-sdf --source c:\myAmazingDataSet\data1.csv
--output c:\myAmazingDataSet\SDF
```

Note

If you are processing multiple files, CSV files are processed as one document per row, and non-CSV files are processed as one document per file.

AWS Management Console

When you upload source documents or DynamoDB items through the Amazon CloudSearch console, they are automatically converted to SDF. You can use the console to upload up to 5 MB of data at a time. If you choose, you can download the generated SDF file. For more information about uploading data

through the console, see [Uploading Data to an Amazon CloudSearch Domain \(p. 77\)](#) and [Uploading DynamoDB Data \(p. 120\)](#).

Configuring Index Fields for an Amazon CloudSearch Domain

Topics

- [Adding Sources for an Amazon CloudSearch Index Field \(p. 59\)](#)
- [Command Line Tools \(p. 60\)](#)
- [AWS Management Console \(p. 61\)](#)
- [API \(p. 66\)](#)

Each document that you add to your search domain has a collection of fields that contain the data that can be searched or returned. The value of a field can be either text or a number. Every document must have a unique document ID, a version number, and at least one field.

In your domain configuration, you define all of the document fields you want to include in your index. Any fields that occur in a document that are not part of your domain configuration are ignored and will not be searchable or returnable. Documents can contain a subset of the fields configured for the domain—every document does not have to contain all fields.

Note

By default, if no search field is specified in a search request, Amazon CloudSearch searches all text fields configured for the domain. You can change this behavior by specifying a default search field for the domain using the `UpdateDefaultSearchField` (p. 185) configuration action.

Amazon CloudSearch supports three types of index fields:

- **text**—a text field contains arbitrary alphanumeric data. A text field is always searchable. The value of a text field can either be returned in search results or the field can be used as a facet. By default, text fields are *neither* result-enabled or facet-enabled.
- **literal**—a literal field contains an identifier or other data that you want to be able to match exactly. The value of a literal field can be returned in search results or the field can be used as a facet, but not both. By default, literal fields are not search-enabled, result-enabled, or facet-enabled.
- **uint**—a uint field contains an unsigned integer value. Uint fields are always searchable, the value of a uint field can always be returned in results, and faceting is always enabled. Uint fields can also be used in rank expressions.

Note

If your document data contains a text or literal field whose value you want to be able to return in results and also use as a facet, you can use the document field as a source for two different index fields and make one returnable, and enable faceting for the other.

When configuring index fields, you can specify:

- Whether literal fields can be searched
- Whether facets can be calculated for text or literal fields to enable filtering
- Whether the contents of a text or literal field can be returned in the search results
- A default value for the field
- Up to 20 data sources for the field

Note

Making text and literal fields result-enabled increases the size of your index, which can increase the cost of running your domain. When possible, it's best to retrieve large amounts of data from an external source, rather than embedding it in your index. Since it can take some time to apply document updates across the domain, critical data such as pricing information should be retrieved from an external source using the returned document IDs instead of returned from the index.

Field names must begin with a letter and be at least 3 and no more than 64 characters long. The allowed characters are: a-z (lower-case letters), 0-9, and _ (underscore). The names "body", "docid", and "text_relevance" are reserved names and cannot be specified as field names.

[Adding Sources for an Amazon CloudSearch Index Field \(p. 59\)](#) describes how document fields are used to populate your index fields. You can configure fields using the `cs-configure-from-sdf` or `cs-configure-fields` (p. 60) command, through the [Amazon CloudSearch console \(p. 61\)](#), or using the `DefineIndexField` (p. 66) configuration action.

Adding Sources for an Amazon CloudSearch Index Field

A source is a field in the SDF document data that you want to use to populate the index field. A field can have up to 20 sources. If you don't specify a source, the source field with the same name as the index field is used as the source. You can specify sources to:

- Combine the contents of multiple fields into a single field.
- Strip common title words from a field so you can use it for sorting.
- Map values from the source to a different set of values in your new index field.

When defining a source for an index field, you specify the name of the source field, a default value to use if the source field doesn't exist in the document data, and how you want to use the source field:

- **Copy**—takes the data from the source field and puts it into the index field without any modifications. Copy is often used when you want to add multiple sources for an index field so you can easily search across all of the source fields. For example, you might copy the data from two source fields called *actor* and *director* into a single searchable field called *people*.
- **Trim Title**—takes the contents of the source field and removes common title prefixes such as "the", and then populates the index field with the trimmed data. For example, if the source field contains the title "The Catcher in the Rye", the trimmed version stored in the index field would be "Catcher in the Rye". Trim Title is often used to populate an index field you can use for sorting.

- **Map**—takes a key found in the source field and maps it to a value you want to store in the index field. For example, you might map the keys *red* and *yellow* to the value *warm*, and *blue* and *green* to the value *cool*.

Command Line Tools

The command line tools provide two ways to configure your domain's index fields. You can:

- Use the `cs-configure-from-sdf` (p. 151) command to analyze one or more SDF batches and automatically configure index fields for your domain based on the content.
- Use the `cs-configure-fields` (p. 143) command to create and configure individual index fields.

Configuring Index Fields from SDF

To automatically configure your domain's index fields

1. Run the `cs-configure-from-sdf` command to analyze one or more SDF batches and configure fields for your domain. (For information about how to create SDF batches, see [Preparing Your Data for Amazon CloudSearch](#) (p. 50).) For example, to configure fields for the *movies* domain based on the SDF batch defined in `moviedata.json`:

```
cs-configure-from-sdf --domain-name movies --source moviedata.json
Detected source format for moviedata.json:json
Analyzing moviedata.json
-----
----
Existing field configuration for the domain - imdb-movies :
-----
----
Detected field configurations from all the sources :
genre                literal (Search Facet)
title                text (Result)
actor                text (Result)
director             text (Result)
-----
----
New proposed field configuration for the domain - imdb-movies :
genre                literal (Search Facet)
[NEW]
title                text (Result) [NEW]
actor                text (Result) [NEW]
director             text (Result) [NEW]
-----
----
```

2. When prompted, enter `y` to confirm that you want to configure your domain with the specified fields. (You can easily modify the configuration later through the console or using the `cs-configure-fields` command.)

```
Configure [imdb-movies] with analyzed fields y/N: y
```

Note

When you add or reconfigure index fields, you must rebuild your index for the changes to be visible in search results. For more information, see [Indexing Document Data with Amazon CloudSearch](#) (p. 82).

Configuring Individual Fields

To add an index field to your domain

- Run the `cs-configure-fields` (p. 143) command and specify the name of the new field with the `--name` option, and the field type with the `--type` option. For example, to add a uint year field to the movies domain:

```
cs-configure-fields --domain-name movies --name year --type uint
Updated 1 Index Field:
year                RequiresIndexDocuments          uint ()
```

By default, the source for the index field is the source field name of the same name. You can specify up to 20 `--source` options to configure sources for the index field. The values from all of the specified sources are concatenated and copied to the index field.

Note

When you add fields or reconfigure existing fields, you need to explicitly issue a request to re-index your data when you are done making configuration changes. For more information, see [Indexing Document Data with Amazon CloudSearch](#) (p. 82).

AWS Management Console

You can easily [configure individual index fields](#) (p. 61) for your domain through the **Indexing Options** panel in the Amazon CloudSearch console. You can also [copy all of the index fields from an existing domain](#) (p. 64) when you create a new domain.

Configuring Individual Fields

To configure a new index field

- Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
- In the **Navigation** pane, click the name of the domain that you want to configure, and then click the domain's **Indexing Options** link.
- To create a new index field, click **Add Index Field** to add a field specification to the list. (If you haven't created any fields yet, a blank field specification is shown on the Indexing Options page by default.)

Amazon CloudSearch Developer Guide Configuring Individual Fields

Indexing Options for Domain imdb-movies

Refresh Help

CloudSearch can help you define your index fields automatically using a configuration wizard or you can manually edit the configuration below.

Name	Status	Type	Search	Facet	Result	Default Value	Source	Remove
actor	Active	literal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		add	<input type="checkbox"/>
director	Active	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		add	<input type="checkbox"/>
genre	Active	literal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		add	<input type="checkbox"/>
title	Active	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		add	<input type="checkbox"/>
year	Active	uint	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		add	<input type="checkbox"/>

Add Index Field

Revert Submit

- Specify a unique name for the field and select the field type: *text*, *literal*, or *uint*. Field names must begin with a letter and be at least 3 and no more than 64 characters long. The allowed characters are: a-z (lower-case letters), 0-9, and _ (underscore). The names "body", "docid", and "text_relevance" are reserved names and cannot be used as custom field names.

Indexing Options for Domain imdb-movies

Refresh Help

CloudSearch can help you define your index fields automatically using a configuration wizard or you can manually edit the configuration below.

Name	Status	Type	Search	Facet	Result	Default Value	Source	Remove
actor	Active	literal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		add	<input type="checkbox"/>
director	Active	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		add	<input type="checkbox"/>
genre	Active	literal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		add	<input type="checkbox"/>
title	Active	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		add	<input type="checkbox"/>
year	Active	uint	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		add	<input type="checkbox"/>
rating	new	literal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		add	<input type="checkbox"/>

Add Index Field

Revert Submit

- To make a literal field searchable, enable the **Search** checkbox.
- To use a text or literal field as a facet to enable filtering, enable the **Facet** checkbox. (Note that the field can be facet-enabled, or result-enabled, but not both.)
- To allow a text or literal field value to be returned in search results, enable the **Result** checkbox. (Note that the field can be facet-enabled, or result-enabled, but not both.)
- Specify a default value for the field (optional). This value is used when no value is specified for the field in the document data.
- To add a source for the field:
 - Click the **add** link in the **Source** column.
 - In the **Add Source** dialog box, enter the name of the source field you want to use as a source for the specified index field.

Add Source Cancel x

Specify a source for the **rating** index field. The source can be any field defined in your SDF data. The index field will be populated with the data from the specified source according to the selected Transform Type. If no source is specified for an index field, the data from the SDF field of the same name is copied to the index field.

Choose a Transform Type to specify how the index field should be populated.

Source Field*:

Default Value:

Type: Copy Trim Title Map

Copy takes the data from the source field and puts it into the index field without any modifications. Copy is often used when you want to populate an index field with data from multiple sources. For example, you could copy the data from the *actor* and *director* source fields into a single index field called *people*.

Cancel Add ▶

- c. Enter a default value to use for the index field if the specified source field doesn't exist in the document data (optional).
- d. Select a **Transform Type** to specify how the index field should be populated: *Copy*, *Trim Title*, *Map*. For more information about using source fields, see [Adding Sources for an Amazon CloudSearch Index Field \(p. 59\)](#).
- e. If you select the *Map* transform type, enter one or more key-value pairs to specify how you want to map the source data to the index field.

Add Source Cancel x

Choose a Transform Type to specify how the index field should be populated.

Source Field*:

Default Value:

Type: Copy Trim Title Map

Map takes a key found in the source field and maps it to a value you want to store in the index field. For example, you might map the keys *red* and *yellow* to the value *warm*, and *blue* and *green* to *cool*.

Key	Value	
<input type="text" value="red"/>	<input type="text" value="warm"/>	<input type="button" value="Remove"/>
<input type="text" value="yellow"/>	<input type="text" value="warm"/>	<input type="button" value="Remove"/>
<input type="text" value="blue"/>	<input type="text" value="cool"/>	<input type="button" value="Remove"/>
<input type="text" value="green"/>	<input type="text" value="cool"/>	<input type="button" value="Remove"/>

Cancel Add ▶

- f. Click **Add** to save your changes.
10. To configure additional fields, click **Add Index Field** and repeat these configuration steps.
 11. When you are done configuring fields, click **Submit** to save your changes. To restore the previous field configurations, click **Revert**.

Cloning an Existing Domain's Indexing Options

During development and testing, you might want to create an empty domain that has the same index fields as an existing domain. You can do this through the Amazon CloudSearch console.

To clone a search domain:

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. At the top of the **Navigation** pane, click the **Create a New Domain** button.



3. In the **NAME YOUR DOMAIN** step, enter a name for the new domain and click **Continue**.
4. In the **CONFIGURE INDEX** step, select **Copy the configuration from another search domain**, choose the domain you want to copy, and click **Continue**.

Note

This will *only* copy the domain's indexing options, access policies and text options are not copied from the specified domain.



5. In the **REVIEW INDEX CONFIGURATION** step, make any adjustments you want and click **Continue**. For more information about configuring index fields, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).

Amazon CloudSearch Developer Guide Cloning an Existing Domain's Indexing Options

The screenshot shows the 'Create New Search Domain' wizard in the 'REVIEW INDEX CONFIGURATION' step. The progress bar indicates the current step. Below the progress bar, a message states: 'The suggested index configuration is shown below. You can edit these fields or add additional fields. Click Continue when you are finished making changes.'

Suggested Index Field Configuration

Name	Type	Search	Facet	Result	Default Value	Source	Remove
actor	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>	[add]	<input type="checkbox"/>
director	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>	[add]	<input type="checkbox"/>
genre	literal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>	[add]	<input type="checkbox"/>
title	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>	[add]	<input type="checkbox"/>
year	uint	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>	[add]	<input type="checkbox"/>

Buttons: Add Index Field, < Back, Cancel, Continue

- In the **SET UP ACCESS POLICIES** step, click **Continue** to select access policies for the new domain. Access policies are *not* automatically copied from the cloned domain. For more information about configuring access policies, see [Configuring Access for an Amazon CloudSearch Domain \(p. 34\)](#).

The screenshot shows the 'Create New Search Domain' wizard in the 'SET UP ACCESS POLICIES' step. The progress bar indicates the current step. Below the progress bar, a message states: 'To authorize or block IP addresses, add one or more access policy rules. You can always access all services through the console, regardless of the rules defined here.'

Set my policy to:

- Recommended rules (Search service: Allow all. Document service: Your IP address only)
- Allow only my IP address access to all services (can be used for testing)
- Allow everyone access to all services (not recommended because anyone can upload documents)
- Deny everyone access to all services (except through the console)
- Copy access policy from another domain

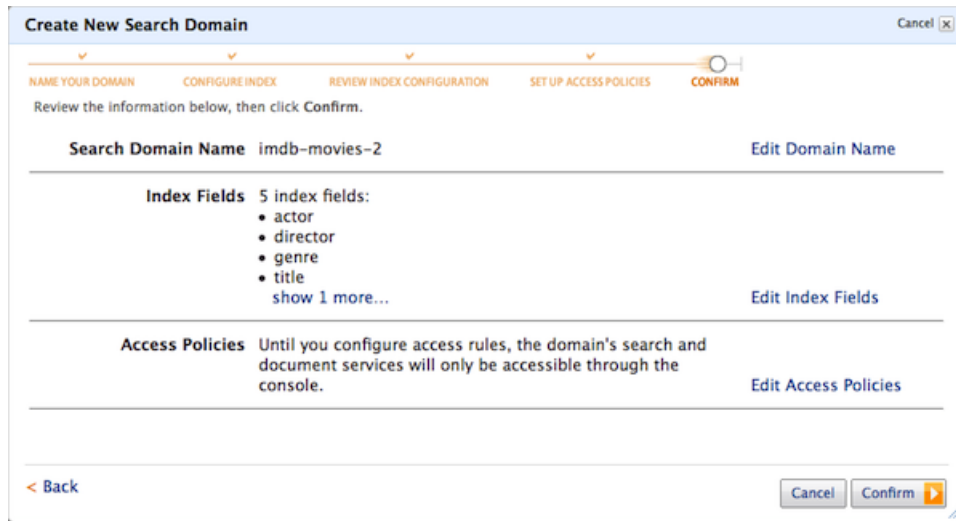
Current Policy:

Effect	Service	IP Ranges	Remove
			<input type="checkbox"/>

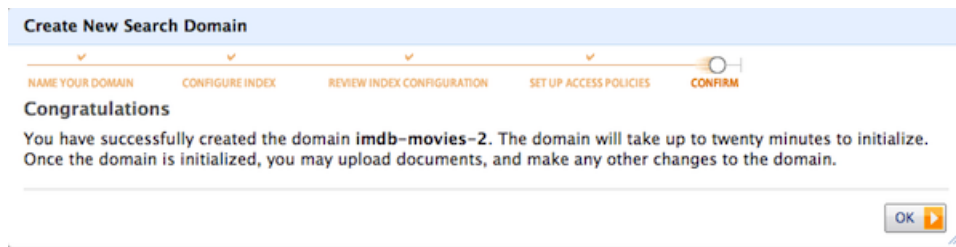
You have no rules. This will deny all access to your domain (except from the console).

Buttons: Add a New Rule, < Back, Cancel, Continue

- In the **CONFIRM** step, review the domain configuration and click **Confirm** to create your domain.



8. Once the domain has been created, click **OK** to exit the Create New Search Domain wizard and go to the domain's dashboard.



API

You use the [DefineIndexField](#) (p. 167) configuration action to add field definitions to your domain configuration. If the specified field already exists, `DefineIndexField` replaces it.

The type-specific options enable you to define a default value for a field, and enable or disable specific features for text and literal fields:

- `FacetEnabled`—controls whether facets can be calculated for this field. Calculating facets determines how many documents contain matching values for the field. Facet counts are not automatically returned for facet-enabled fields; they must be explicitly requested at search time. (Uint fields are always facet-enabled.)
- `ResultEnabled`—controls whether the contents of a text or literal field can be returned. (Uint fields are always returnable.)
- `SearchEnabled`—controls whether a literal field is searchable. (Text and uint fields are always searchable.)

For example, to create a uint index field called `year` and populate it with the data from the `yearreleased` field in the SDF data:


```
https://cloudsearch.us-east-1.amazonaws.com
?Action=DefineIndexField
&DomainName=movies
&IndexField.IndexFieldName=year
&IndexField.IndexFieldType=uint
&IndexField.SourceAttributes.member.1.SourceDataCopy.SourceName=yearreleased
&IndexField.SourceAttributes.member.1.SourceDataFunction=Copy
&IndexField.UIntOptions.DefaultValue=0
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120401/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-04-01T17:00:07.803Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=b291a01dd69a49e04f4a84862b38e0758c53cf93b76dd452cc802886b20
724bc
```

Note

Amazon CloudSearch configuration requests are authenticated using your access key ID and secret access key. For more information about signing requests, see [Request Authentication \(p. 25\)](#).

Configuring Text Options for an Amazon CloudSearch Domain

Topics

- [Configuring Stemming in Amazon CloudSearch \(p. 68\)](#)
- [Configuring Stopwords in Amazon CloudSearch \(p. 71\)](#)
- [Configuring Synonyms in Amazon CloudSearch \(p. 74\)](#)

Amazon CloudSearch enables you to specify the following text options to control how your domain's data is indexed:

- **Stems**—map related words to a common root word or stem.
- **Stopwords**—words that should be ignored during indexing and searching.
- **Synonyms**—words that have the same meaning as words that occur in your data and should produce the same search results.

Although the defaults work well in many cases, fine-tuning these dictionaries is one way to optimize the search results based on your knowledge of the data you are searching.

When you modify a domain's text options, you must explicitly [rebuild the index \(p. 82\)](#) for the changes to be reflected in search results.

During text processing, search terms are converted to lower-case, so stopwords, stems, and synonyms are not case-sensitive.

For more information about how Amazon CloudSearch normalizes and tokenizes text and applies configured text options when indexing text fields and processing search requests, see [Text Processing in Amazon CloudSearch \(p. 247\)](#).

Configuring Stemming in Amazon CloudSearch

A stemming dictionary maps related words to a common stem. A stem is typically the root or base word from which variants are derived. For example, *run* is the stem of *running* and *ran*. During indexing, Amazon CloudSearch uses the stemming dictionary when it performs text-processing on text fields. At search time, the stemming dictionary is used to perform text-processing on the search request. This enables

matching on variants of a word. For example, if you map the term *running* to the stem *run* and then search for *running*, the request matches documents that contain *run* as well as *running*.

Stems are specified as a collection of term and stem pairs. When you configure stemming options, the existing stemming dictionary is replaced with the mappings you specify. By default, Amazon CloudSearch does not define any stems. However, some basic algorithmic stemming is always performed, such as removing plural suffixes. (This is done whether or not you specify a custom stemming dictionary.)

The maximum size of a stemming dictionary is 500 KB.

You can configure stems using the `cs-configure-text-options` (p. 69) command, from the [Amazon CloudSearch console](#) (p. 69), or using the `UpdateStemmingOptions` (p. 70) configuration action.

Command Line Tools

You can use the `cs-configure-text-options` (p. 148) command to upload a text file that contains a list of term and stem pairs.

To configure stemming options

1. Create a text file for your stemming dictionary and specify one comma-separated *term*, *stem* pair per line. For example:

```
mice, mouse
```

```
people, person
```

```
running, run
```

2. Run the `cs-configure-text-options` command with the `--stems` option to upload the stemming dictionary to your domain:

```
cs-configure-text-options -d mydomain --stems stems.txt
Updating Stemming options
Read the stems file
Sent 3 token stem pairs.
```

3. If you are done making configuration changes, run the `cs-index-documents` command to rebuild the domain's index.

```
cs-index-documents -d mydomain
```

AWS Management Console

You can configure a domain's stemming options from the **Text Options** pane in the Amazon CloudSearch console.

To configure stemming options

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Text Options** link.

3. In the **Text Options** pane, click the **Stemming** tab.
4. For each *term*, *stem* pair you want to add to the stemming dictionary, enter the term and its stem and click the **Add** button. You can also edit the list directly or copy and paste the list into a text editor to make changes.

Text Options for Domain imdb-movies

Refresh Help

Stopwords **Stemming** Synonyms

Your stems are **ACTIVE**.
Stems map variations of a term to a common stem to enable matching on all the variants. To customize your domain's stemming options, edit the term and stem pairs in the list below. Specify one comma-separated *term*, *stem* pair per line.
Tip: You can **select** the list to copy and paste it into a text editor.

Add a Term:

Stem: **Add**

Current Stems:

Revert Submit

5. Click **Submit** to save your changes.
6. If you are done making configuration changes, click **Run Indexing** on the domain dashboard to rebuild the domain's index.

API

Use the [UpdateStemmingOptions](#) (p. 189) configuration action to upload a JSON-formatted stemming dictionary to your domain. A stemming dictionary has a single JSON object with one property, *stems*. The value of the *stems* property is an object that contains a collection of *string: value* pairs that map terms to their stems:

```
{"stems": {"term1": "stem1", "term2": "stem2", "term3": "stem3"}}
```

For example:

```
https://cloudsearch.us-east-1.amazonaws.com  
?Action=UpdateStemmingOptions  
&DomainName=movies  
&Stems={"stems": {"mice": "mouse", "people": "person", "running": "run"}}
```

```
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120402/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-04-02T21:43:50.884Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=4f7a17dc53fbd7e08b3d3a0c4d771466fe48d2739c8d6333ebe0261d
88941488
```

Configuring Stopwords in Amazon CloudSearch

Stopwords are words that should typically be ignored both during indexing and at search time because they are either insignificant or so common that including them would result in a massive number of matches.

During indexing, Amazon CloudSearch uses the stopwords dictionary when it performs text-processing on text fields. In most cases, stopwords are not included in the index. (If multiple stopwords appear consecutively in a text field, they are not filtered out. This enables searches for phrases such as "to be or not to be" to return valid results.) At search time, the stopwords dictionary is used to perform text-processing on the search request.

The stopwords dictionary must explicitly list each word you want to ignore. Wildcards and regular expressions are not supported.

By default, Amazon CloudSearch defines the following stopwords for English (en):

```
a
an
and
are
as
at
be
but
by
for
in
is
it
of
on
or
the
to
was
```

You can configure stopwords using the [cs-configure-text-options](#) (p. 71) command, from the [Amazon CloudSearch console](#) (p. 72), or using the [UpdateStopwordOptions](#) (p. 73) configuration action.

Command Line Tools

You can use the [cs-configure-text-options](#) (p. 148) command to upload a stopwords dictionary.

To configure stopwords

1. Create a text file that contains your stopwords dictionary. In the file, specify one stopword per line. For example:

```
the  
or  
and
```

2. Run the `cs-configure-text-options` command with the `--stopwords` option to upload the stopwords dictionary to your domain.

```
cs-configure-text-options -d mydomain --stopwords stopwords.txt  
Updating stop words options  
Sent 3 stop words.
```

3. If you are done making configuration changes, run the `cs-index-documents` command to rebuild the domain's index.

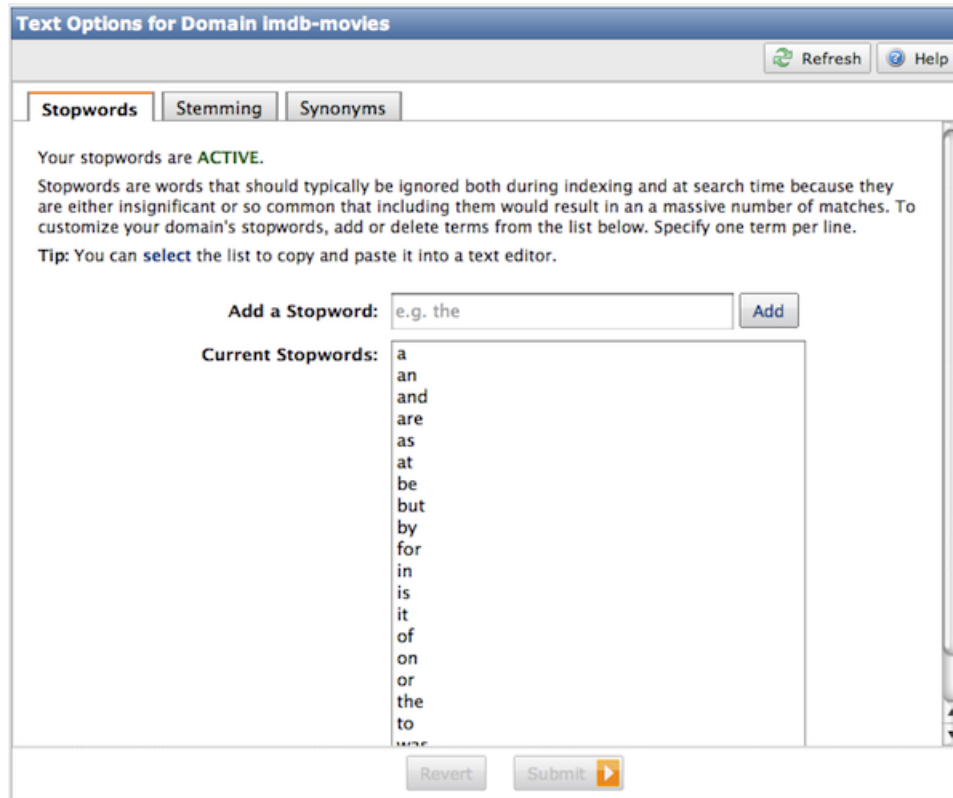
```
cs-index-documents -d mydomain
```

AWS Management Console

You can configure a domain's stopwords dictionary from the **Text Options** pane in the Amazon CloudSearch console.

To configure stopwords

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Text Options** link.
3. On the **Stopwords** tab, for each word you want to add to the stopwords dictionary, enter it in the **Add a Stopword** field and click **Add**. You can also edit the list directly or copy and paste the list into a text editor to make changes.



4. Click **Submit** to save your changes.
5. If you are done making configuration changes, click **Run Indexing** on the domain dashboard to rebuild the domain's index.

API

Use the [UpdateStopwordOptions](#) (p. 191) configuration action to upload a JSON-formatted stopword dictionary to your domain. A stopword dictionary has a single JSON object with one property, *stopwords*. The value of the stopwords property is an object that contains an array of strings:

```
{"stopwords": ["string1", "string2", "string3"]}
```

For example:

```
https://cloudsearch.us-east-1.amazonaws.com
?Action=UpdateStopwordOptions
&DomainName=movies
&Stopwords={"stopwords": ["a", "an", "the", "of"]}
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120402/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-04-02T21:47:23.216Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=47bc42cfb11561dffade2779bac6af7f6b53d76d2a729f9e62b2e528
d5eac319
```

Configuring Synonyms in Amazon CloudSearch

You can configure synonyms for terms that appear in the data you are searching. That way, if a user searches for the synonym rather than the indexed term, the results will include documents that contain the indexed term. For example, you might want to configure synonyms so that a search for "Rocky Four" or "Rocky 4" will match the movie titled "Rocky IV". To do that, you would configure *4* and *four* as synonyms of the indexed term *IV*.

If you want two terms to match the same documents, you must define them as synonyms of each other. For example:

```
cat, feline  
feline, cat
```

The synonym dictionary is used during indexing to configure mappings for terms that occur in text fields. No synonym processing is done on the search request. By default, Amazon CloudSearch does not define any synonyms.

You can configure synonyms using the `cs-configure-text-options` (p. 74) command, from the [Amazon CloudSearch console](#) (p. 74), or using the `UpdateSynonymOptions` (p. 75) configuration action.

Command Line Tools

You can use the `cs-configure-text-options` (p. 148) command to upload a text file that contains your synonym dictionary.

To configure synonyms

1. Create a text file that contains your synonym dictionary. Each line in the file should specify a term followed by a comma-separated list of its synonyms. For example:

```
cat, feline, kitten  
dog, canine, puppy  
horse, equine, colt
```

2. Run the `cs-configure-text-options` command with the `--synonyms` option to upload the synonym dictionary to your domain.

```
cs-configure-text-options -d mydomain --synonyms synonyms.txt
```

3. If you are done making configuration changes, run the `cs-index-documents` command to rebuild the domain's index.

```
cs-index-documents -d mydomain
```

AWS Management Console

You can configure a domain's synonyms from the **Text Options** pane in the Amazon CloudSearch console.

To configure synonyms

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Text Options** link.
3. In the **Text Options** pane, click the **Synonyms** tab.
4. For each term and synonym list that you want to add to your synonyms dictionary, enter the term in the **Add a Term** field and the comma-separated list of synonyms in the **Synonyms** field. You can also edit the list directly or copy and paste the list into a text editor to make changes.

The screenshot shows the 'Text Options for Domain imdb-movies' interface. At the top, there are 'Refresh' and 'Help' buttons. Below that are three tabs: 'Stopwords', 'Stemming', and 'Synonyms' (which is selected). The main content area contains the following text: 'Your synonyms are ACTIVE. A synonym has the same or nearly the same meaning as a term that appears in your data and should match documents that contain that term. To customize your domain's synonyms, edit the terms and synonyms in the list below. On each line, specify a term followed by a comma-separated list of its synonyms: term, synonym1, synonym2, Tip: You can select the list to copy and paste it into a text editor.' Below this text is a form with two input fields: 'Add a Term:' with the example 'e.g. cat' and 'Synonyms:' with the example 'e.g. feline, kitten'. An 'Add' button is to the right of the 'Synonyms' field. Below these fields is a large empty box labeled 'Current Synonyms:'. At the bottom of the form are 'Revert' and 'Submit' buttons.

5. Click **Submit** to save your changes.
6. If you are done making configuration changes, click **Run Indexing** on the domain dashboard to rebuild the domain's index.

API

Call [UpdateSynonymOptions](#) (p. 193) to upload a JSON-formatted synonym dictionary to your domain. A synonym dictionary has a single JSON object with one property, *synonyms*. The value of the *synonyms* property is a collection of *string: value* pairs that map each term to one or more synonyms. To map a term to multiple synonyms, specify the synonyms as an array of strings:

```
{ "synonyms": {  
  "term1": [ "synonym1", "synonym2" ],  
  "term2": [ "synonym1" ],  
  "term2": [ "synonym1", "synonym2", "synonym3" ]  
}
```

```
}  
}
```

For example:

```
https://cloudsearch.us-east-1.amazonaws.com  
?Action=UpdateSynonymOptions  
&DomainName=movies  
&Synonyms={"synonyms": {  
  "cat": ["feline", "kitten"],  
  "dog": ["canine", "puppy"]}}  
&Version=2011-02-01  
&X-Amz-Algorithm=AWS4-HMAC-SHA256  
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120402/us-east-1/cloudsearch/aws4_re  
quest  
&X-Amz-Date=2012-04-02T21:56:03.214Z  
&X-Amz-SignedHeaders=host  
&X-Amz-Signature=cf9a49e63e25b0131da11c9dccb4c648ff243c01ea4282d14d88e2c6  
ec414523
```

Uploading Data to an Amazon CloudSearch Domain

To make your data searchable, you must describe it according to the Search Data Format (SDF) and upload the resulting SDF batches to a search domain. Amazon CloudSearch can then generate a search index from your SDF data according to the index fields and text options that you have configured for the domain. As your data changes, you submit SDF updates to add, change, or delete documents from your index. Amazon CloudSearch applies data updates continuously, so your changes become searchable in near real-time.

Amazon CloudSearch ensures that the most recent changes are applied to your domain using the document version numbers specified in the SDF add and delete operations. The operation with the greatest version number always takes precedence. To be applied, the version number in the add or delete operation must be greater than the document's current version number in the index. If the version number in an add or delete operation is less than the document's current version number, the operation is ignored. If an operation specifies the same document version that already exists in the index, the result is undefined—there's no guarantee which one will take precedence.

Important

To successfully upload SDF data to your domain, it has to be valid JSON or XML and conform to the SDF data conventions. For information about creating SDF batches, see [Preparing Your Data for Amazon CloudSearch \(p. 50\)](#).

For information about configuring index fields for a domain, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).

You can submit SDF data to a domain using the `cs-post-sdf` (p. 78) command, from the [Amazon CloudSearch console \(p. 78\)](#), or by [posting it directly \(p. 81\)](#) to the domain's Document endpoint.

Note

You are billed for the total number of document batches uploaded to your search domain, including batches that contain delete operations. For more information about Amazon CloudSearch pricing, see aws.amazon.com/cloudsearch/pricing/.

Command Line Tools

You use the `cs-post-sdf` (p. 157) command to send SDF data to your search domain. The SDF batches can be local or stored in Amazon S3. For information about installing and setting up the Amazon CloudSearch command line tools, see [Amazon CloudSearch Command Line Tool Reference \(p. 136\)](#).

To send data to a domain for indexing

1. If you haven't already, prepare your data according to the SDF schema. For more information about generating SDF, see [Preparing Your Data for Amazon CloudSearch \(p. 50\)](#).
2. Run the `cs-post-sdf` command to upload your SDF data to your domain. You must specify at least one `--source` option to specify the location of the SDF data you want to upload.

```
cs-post-sdf -d mydomain --source data1.sdf
Processing: data1.sdf
Detected source format for data1.sdf as json
Status: success
Added: 5208
Deleted: 0
```

AWS Management Console

In the Amazon CloudSearch console, you can upload data to your domain from the domain dashboard. The console can automatically convert the following types of files to SDF during the upload process:

- Comma Separated Value (.csv)
- Adobe Portable Document Format (.pdf)
- HTML (.htm, .html)
- Microsoft Excel (.xls, .xlsx)
- Microsoft PowerPoint (.ppt, .pptx)
- Microsoft Word (.doc, .docx)
- Text Documents (.txt)
- JSON Documents (.json)
- XML Documents (.xml)

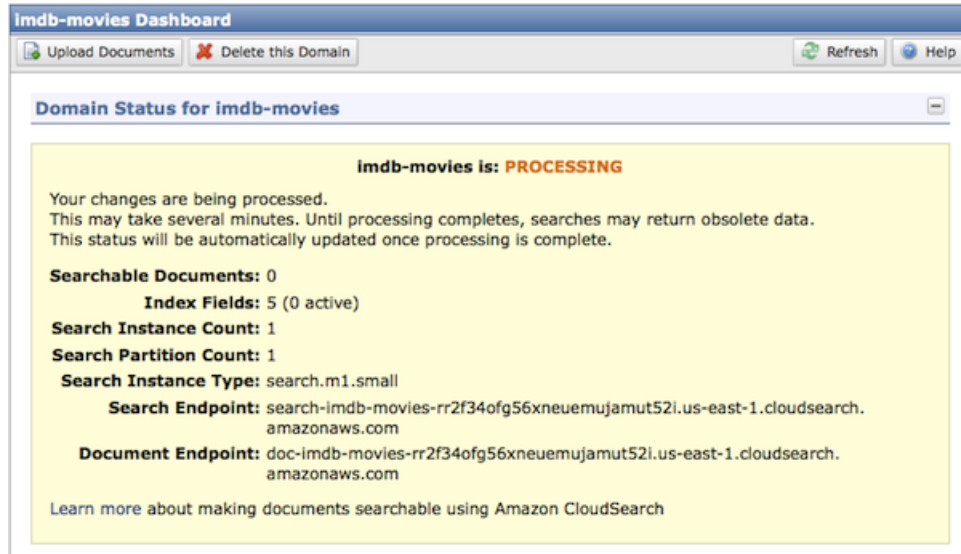
You can also convert and upload items from an DynamoDB table. For more information, see [Uploading DynamoDB Data \(p. 120\)](#).

CSV files are parsed row-by-row and a separate document is generated for each row. All other types of files are treated as a single document. For more information about automatically generating SDF, see [Preparing Your Data for Amazon CloudSearch \(p. 50\)](#).

You can also upload SDF batches through the Amazon CloudSearch console.

To send data to a domain for indexing

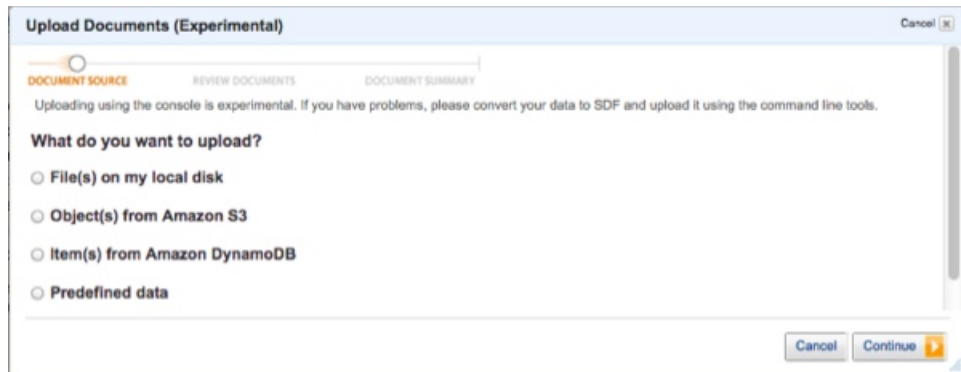
1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain.
3. At the top of the domain dashboard, click **Upload Documents**.



4. Select the location of the data you want to upload to your domain:
- File(s) on my local disk
 - Object(s) from Amazon S3
 - Item(s) from DynamoDB
 - Predefined data

Note

If you upload data in a format other than SDF, it will automatically be converted to SDF during the upload process.



5. If you are uploading local files, click **Browse** to choose the file(s) to upload:



- If you are uploading objects from Amazon S3, select the bucket you want to upload from. To upload the entire contents of the bucket, leave the **Prefix** field empty and click **Add**. To upload selected objects, enter a filter in the **Prefix** field and click **Add**. (You can add multiple prefixes.)

Object(s) from Amazon S3

Select the S3 bucket that contains your sample data. You can specify a prefix to limit which documents will be sampled. You can add multiple S3 buckets up to a maximum of 10000 S3 objects and a total size of 5MB. (Use the command line tools if you need to specify a larger sample data set.)

S3 Bucket:

Prefix:

Bucket/Prefix/Object	Size	Objects	Remove
SupremeCourtDB/scdb	3MB	14	<input type="checkbox"/>
TOTAL	3MB	14	

Objects must be in one of the supported formats. [see list](#)

- If you are uploading items from DynamoDB, select the table you want to upload from. To start reading from a particular item, specify a start key. To limit the read capacity units that can be consumed while reading from the table, enter the maximum percentage of read capacity units.

Item(s) from Amazon DynamoDB

Select the DynamoDB table that contains your sample data. To start reading from a particular item, specify a start key. To limit the read capacity units used while reading from the table, specify the maximum Read Capacity Units %. A maximum of 5 MB of data can be read from the table. If you need to specify a larger data set, use the command line tools.

DynamoDB Table:

Read Capacity Units %:

Start Hash Key:

- If are uploading predefined sample data, choose the data set that you want to use:

Predefined data

Load sample data for

- Once you've selected the data you want to upload, click **Continue**.
- In the **Review Documents** step, review the documents to be uploaded and click **Upload Documents** to continue.

Upload Documents (Experimental) Cancel

DOCUMENT SOURCE REVIEW DOCUMENTS DOCUMENT SUMMARY

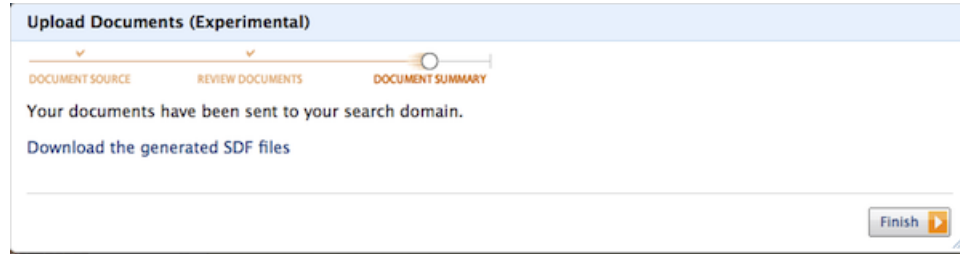
Your upload includes 5208 documents containing the following fields:

- actor
- director
- genre
- title
- year

[Download the generated SDF files](#)

< Back

- In the **Document Summary** step, if SDF batches have been automatically generated from your data, you can click **Download the generated SDF files** to get them. Click **Finish** to return to the domain dashboard.



API

You use the [documents/batch](#) (p. 217) API to post SDF data to your domain to add, update, or remove documents. For example:

```
curl -X POST --upload-file data1.sdf doc.movies-123456789012.us-east-1.cloud
search.amazonaws.com/2011-02-01/documents/batch --header "Content-Type:applica
tion/json"
```

Indexing Document Data with Amazon CloudSearch

When you send document updates to your domain, Amazon CloudSearch automatically updates the domain's search index with the new data in near real time. You don't have to do anything for the updates to be indexed. However, if you change the configuration of your domain's index fields or text options, you must explicitly rebuild your search index for those changes to be visible in search results. Because rebuilding the index can take a significant amount of time if you have a lot of data, you should finish making all of your configuration changes before re-indexing your documents.

When you make changes that require re-indexing, the domain status changes to `NEEDS INDEXING`. While the index is being rebuilt, the domain's status is `PROCESSING`. You can continue to submit search requests while indexing is in process, but the configuration changes won't be visible in search results until indexing completes and the domain's status changes to `ACTIVE`.

Note

Depending on the volume of data, building a full index can take a considerable amount of compute power. Amazon CloudSearch automatically manages the resources needed to build the index in a timely fashion. Most data updates and simple domain configuration changes are built and deployed in minutes. Indexing large volumes of data and applying configuration changes that require rebuilding the full index will take longer to complete.

You can initiate indexing using the `cs-index-documents` (p. 82) command, from the [Amazon CloudSearch console](#) (p. 83), or using the `IndexDocuments` (p. 84) configuration action.

Command Line Tools

You use the `cs-index-documents` (p. 155) command to rebuild your domain's search index.

To explicitly index your domain

- Run the `cs-index-documents` command. For example, to rebuild the index for a domain called *movies*:

```
cs-index-documents --domain-name movies
=====
Indexing documents for domain [movies]
```



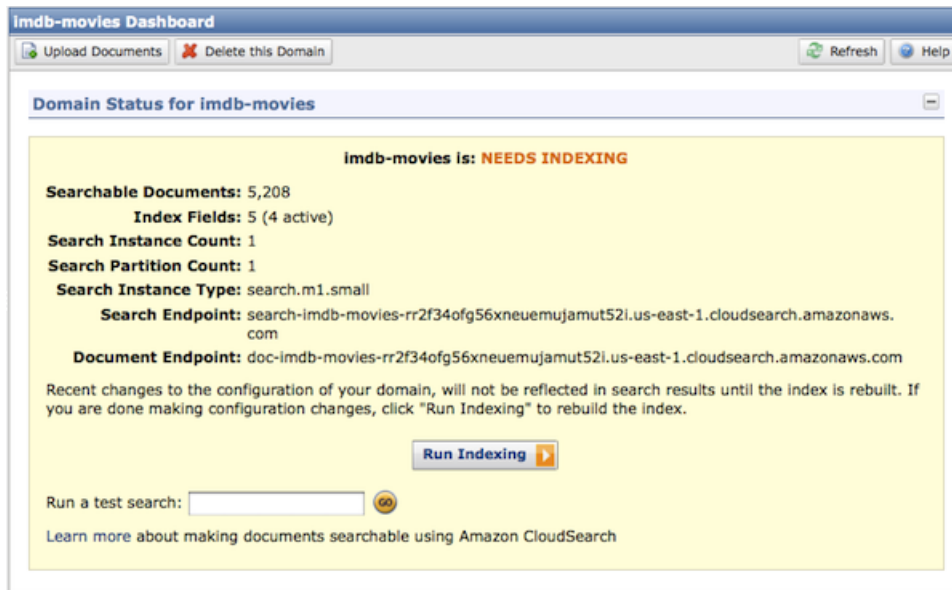
```
Now indexing fields:
=====
actor
director
genre
title
year
=====
```

AWS Management Console

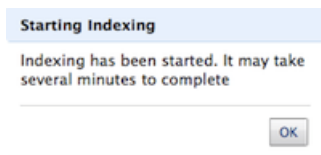
When you make changes that require your domain's index to be rebuilt, the status shown on the domain dashboard changes to **NEEDS INDEXING**. The console also displays a message at the top of the configuration pages prompting you to run indexing when you are done making changes.

To run indexing

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain that needs indexing.
3. On the **Domain Dashboard**, click the **Run Indexing** button.



4. Click **OK** in the **Starting Indexing** dialog box to return to the domain dashboard.



API

You use the [IndexDocuments \(p. 184\)](#) configuration action to initiate indexing. For example:

```
https://cloudsearch.us-east-1.amazonaws.com
?Action=IndexDocuments
&DomainName=movies
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120402/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-04-02T22:41:07.764Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=cf2f7663cc7c80901474f889ab9b1b8e65deea5be1e2c527319bc8e1
6859d7a4
```

Note

Amazon CloudSearch configuration requests are authenticated using your access key ID and secret access key. For more information about signing requests, see [Request Authentication \(p. 25\)](#).

Searching Your Data with Amazon CloudSearch

Topics

- [Submitting Search Requests in Amazon CloudSearch \(p. 86\)](#)
- [Searching Text Fields in Amazon CloudSearch \(p. 87\)](#)
- [Searching Literal Fields in Amazon CloudSearch \(p. 91\)](#)
- [Searching Uint Fields in Amazon CloudSearch \(p. 92\)](#)
- [Constructing Boolean Search Queries in Amazon CloudSearch \(p. 93\)](#)
- [Controlling How Search Results are Returned in Amazon CloudSearch \(p. 94\)](#)
- [Getting and Using Facet Information in Amazon CloudSearch \(p. 96\)](#)
- [Tuning Search Requests in Amazon CloudSearch \(p. 103\)](#)

When you submit a search request, each document in your search domain is examined to see if it matches the query constraints specified in the request. The matching documents (search hits) are then sorted according to the ranking preferences specified in the request. If no ranking preferences are specified, the matching documents are sorted using their `text_relevance` scores. The rank is set to `-text_relevance` to list the documents in descending order with the highest-scoring documents first. For information about how you can specify field weights to customize how `text_relevance` scores are calculated, see [Using Relative Field Weighting to Customize Text Relevance in Amazon CloudSearch \(p. 110\)](#).

Query constraints can be specified in a number of ways. In the simplest case, a text string provided by the user is used as a query constraint to find documents that contain the specified text in the default search field. For example, you could specify the query constraint `q=star+wars` to retrieve a list of the documents that contain the terms "star" and "wars" in any text field:

```
http://search-domainname-domainid.us-east-1.cloudsearch.amazonaws.com/2011-02-01/search?q=star+wars
```

You can specify additional constraints to search specific fields and use Boolean logic to construct more complex queries. When you search text fields, Amazon CloudSearch finds all documents that contain the search terms anywhere within the specified field. If the field being searched is a literal field, the field contents must exactly match the search string to be returned in results. You can search uint fields for a particular value or a range of values.

In your search requests, you can also specify how you want Amazon CloudSearch to rank and return the search hits. Instead of using the default `text_relevance` scores, you can rank hits alphabetically, numerically, or according to your own custom rank expressions.

When you submit a search request, Amazon CloudSearch returns a response that specifies how the results were ranked, the *match expression* that was derived from your query constraints, and a collection of hits that represents the documents that match the query constraints. For example:

```
{
  "rank": "-text_relevance",
  "match-expr": "(label 'star wars')",
  "hits": {
    "found": 7,
    "start": 0,
    "hit": [
      { "id": "tt1185834" },
      { "id": "tt0076759" },
      { "id": "tt0121765" },
      { "id": "tt0080684" },
      { "id": "tt0086190" },
      { "id": "tt0120915" },
      { "id": "tt0121766" } ]
    },
    "info": {
      "rid": "b7c167f6c2da6d93ecb53d18230cbc27146c9356f9c643ec9dec53e707b9af87f27b24b2f4b636a9",
      "time-ms": 4,
      "cpu-time-ms": 0
    }
  }
}
```

- **found**—specifies the total number of documents that matched the query.
- **start**—specifies the offset of the first hit included in the response.
- **id**—specifies the unique document ID of an individual hit.

By default, a search response contains the IDs of the first 10 ranked hits. You can retrieve additional information for each hit by specifying which result enabled fields should be included in the response. You can also control how many hits are returned at a time. When you want to page through a large set of matching documents, you can specify the offset of the first hit that you want to retrieve. For more information, see [Controlling How Search Results are Returned in Amazon CloudSearch \(p. 94\)](#).

Submitting Search Requests in Amazon CloudSearch

You submit search requests to your domain's search endpoint via HTTP GET. To construct a search request, you append the Amazon CloudSearch API version and the name of the resource you are accessing, `2011-02-01/search`, and a query string that specifies the terms and constraints for your search and what you want to get back in the response. The maximum size of a search request is 8190 bytes, including the HTTP method, URI, and protocol version.

For example, the following request performs a simple text search of the `search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com` domain and gets the contents of the title field:

```
http://search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com/2011-02-01/search?q=star+wars&return-fields=title
```

Note

The API version must be specified in all search requests. When there are updates to the Amazon CloudSearch Search API, you access them using a new API version.

The query string in a search request must be URL-encoded. You can use any method you want to send GET requests to your domain's search endpoint—you can enter the request URL directly in a Web browser, use cURL to submit the request, or generate an HTTP call using your favorite HTTP library.

By default, Amazon CloudSearch returns the response in JSON. You can also get the results formatted in XML by specifying the `results-type` parameter, `results-type=xml`.

Note

You can also use the Search Tester in the Amazon CloudSearch console to search your data, browse the results, and view the generated request URLs and JSON and XML responses. For more information, see [Searching with the Search Tester \(p. 16\)](#).

Searching Text Fields in Amazon CloudSearch

Amazon CloudSearch provides two ways to perform free text searches:

- You can use the `q` (p. 87) parameter to search the default search field for one or more terms. By default, this searches all text fields configured for the domain.
- You can use the `bq` (p. 88) parameter to search one or more text fields.

When you search text fields, Amazon CloudSearch finds all documents that contain the search terms anywhere within the specified field, in any order. For example, in the sample movie data, the `title` field is configured as a text field. If you search the title field for "star", you will find all of the movies that contain *star* anywhere in the title field, such as *star*, *star wars*, and *a star is born*. This differs from searching a literal field, where the field value must be identical to the search string to be considered a match.

When searching text fields, you can:

- Use [Boolean operators \(p. 89\)](#) when specifying the terms you are searching for. Amazon CloudSearch supports three operators in text searches: - (NOT), | (OR), and + (AND).
- Use the [wildcard operator \(p. 90\)](#) to perform prefix searches. Amazon CloudSearch only supports the wildcard operator *, which matches zero or more characters at the end of the specified term.
- Use quotes to [search for phrases \(p. 91\)](#).

Searching Text Fields with the Query Parameter in Amazon CloudSearch

The query parameter, `q`, provides an easy way to search the default search field for one or more terms. When you create a domain, the default search field is configured to include all text fields in the index. You can use the [UpdateDefaultSearchField \(p. 185\)](#) configuration action to configure your own default search field.

By default, documents must contain all of the terms you specify to be considered a match. Unlike literal fields, the terms can occur anywhere within the text field, in any order. You can prefix a term with the `-` (NOT) operator to exclude all results that include that term. Similarly, you can separate terms with the `|` (OR) operator if you want to match documents that contain any of the specified terms. For more information, see [Using Boolean Operators in Amazon CloudSearch Text Searches \(p. 89\)](#). To search for a phrase rather than individual terms, enclose the phrase in double quotes. For more information, see [Searching for Phrases in Text Fields in Amazon CloudSearch \(p. 91\)](#).

For example, to search the default search field for *star wars*, specify `q=star+wars` in the query string:

```
https://search-domainname-domainid.us-east-1.cloudsearch.amazonaws.com/2011-02-01/search?q=star+wars
```

The following example shows the default JSON response:

```
{
  "rank": "-text_relevance",
  "match-expr": "(label 'star wars')",
  "hits": {
    "found": 7,
    "start": 0,
    "hit": [
      { "id": "tt1185834" },
      { "id": "tt0076759" },
      { "id": "tt0121765" },
      { "id": "tt0080684" },
      { "id": "tt0086190" },
      { "id": "tt0120915" },
      { "id": "tt0121766" }
    ]
  },
  "info": {
    "rid": "b7c167f6c2da6d93ecb53d18230cbc27146c9356f9c643ec9dec53e707b9af87f27b24b2f4b636a9",
    "time-ms": 4,
    "cpu-time-ms": 0
  }
}
```

Searching Text Fields with the Boolean Query Parameter in Amazon CloudSearch

The Boolean query parameter, `bq`, provides a rich expression language for fine-grained control over document matching. You can search within particular fields and combine expressions with the `and`, `or`, and `not` prefix operators. In addition to searching text fields, you can use the `bq` parameter to search literal and uint fields.

If you don't specify any fields when using the `bq` parameter, the default search field is used, just like with the `q` parameter. For example, the following queries produce the same results:

```
search?bq='star'
search?q=star
```

When constructing queries with `bq`, you must enclose the search terms within single quotes. By default, documents must contain all of the terms you specify to be considered a match. When you search text fields, the terms can occur anywhere within the text field, in any order.

You can prefix a term with the `-` (NOT) operator to exclude all results that include that word. Similarly, you can separate terms with the `|` (OR) operator if you want to match documents that contain any of the specified terms. For more information, see [Using Boolean Operators in Amazon CloudSearch Text Searches](#) (p. 89). To search for a phrase rather than individual terms, enclose the phrase in double quotes. For more information, see [Searching for Phrases in Text Fields in Amazon CloudSearch](#) (p. 91).

To search a particular text field, prefix the search terms with the name of the field you want to search, followed by a colon. For example:

```
search?bq=title:'star'
```

This searches the `title` field of each document and matches all documents whose titles contain the term *star*.

In addition to searching text fields, the `bq` parameter can be used to search specific [literal](#) (p. 91) and [uint](#) (p. 92) fields. To combine matches against multiple fields, you can use the Boolean operators *and*, *or*, and *not*. For more information about constructing Boolean queries, see [Constructing Boolean Search Queries in Amazon CloudSearch](#) (p. 93).

Note

You can only search literal fields that are search-enabled in your domain's configuration. For more information, see [Configuring Index Fields for an Amazon CloudSearch Domain](#) (p. 58).

Using Boolean Operators in Amazon CloudSearch Text Searches

When searching text fields with either the `q` or `bq` parameter, you can use the Boolean operators `+` (AND), `|` (OR), and `-` (NOT). These shortcuts only work for text searches. To create Boolean queries that search `uint` and `literal` fields, you need to use the Boolean query syntax described in [Constructing Boolean Search Queries in Amazon CloudSearch](#) (p. 93).

If you separate search terms with `+` or a space, Amazon CloudSearch matches documents that contain all of the specified search terms—they are ANDed together. You can use the `|` (OR) operator to separate terms when you want to match documents that contain either the preceding term(s) or the following term(s).

To exclude documents that contain a particular term from the search results, prefix the term with the `-` (NOT) operator. For example, to search for all of the documents that don't contain the term *star* in the default search field, you would specify: `search?q=-star`. The NOT operator only applies to individual terms. Searching for `search?q=-star+wars` retrieves all documents that do not contain the term *star*, but do contain the term *wars*.

Note

To retrieve all of the documents in your domain, you can prefix a term that you know doesn't exist in your domain's data with the NOT operator, for example `-1234567`. However, keep in mind that this is a resource intensive operation if you have a large dataset and might be subject to timeouts.

For example, when searching the sample movie data:

- `search?q=star|wars` matches movies that contain either *star* or *wars* in the default search field.
- `search?bq=title:'story funny|underdog'` matches movies that contain both the terms *story* and *funny* or the term *underdog* in the title field.

- `search?bq=title:'red|white|blue'` matches movies that contain either *red*, *white*, or *blue* in the title field.
- `search?bq=actor:'"evans, chris"|"Garity, Troy"'` matches movies that contain either the phrase *evans, chris* or the phrase *Garity, Troy* in the actor field.
- `search?bq='title:-star+war|world'` matches movies whose titles do not contain *star*, but do contain either *war* or *world*.

You can also use the Boolean operators when constructing queries using the full Boolean query syntax. For example, `search?bq=(and director:'Lucas|Spielberg' (not actor:'"Ford, Harrison"'))` matches movies that either Lucas or Spielberg directed, but did not star Harrison Ford. For more information about the Boolean query syntax, see [Constructing Boolean Search Queries in Amazon CloudSearch \(p. 93\)](#).

Using Wildcards in Amazon CloudSearch Text Searches

You can use the * (asterisk) wildcard operator to perform prefix matching. The * operator only applies to individual terms. When you append the * operator to a string, the string is treated as a prefix. Amazon CloudSearch matches results that contain the prefix followed by zero or more characters. Prefix searches are expanded to a maximum of 2,000 indexed terms. If more than 2,000 terms match the prefix, the search results will not include all possible matches.

If you're searching a text field, the matched prefix can occur anywhere within the contents of the field. You can also use the wildcard operator to perform "starts with" searches in literal fields. For more information, see [Using Wildcards in Literal Searches in Amazon CloudSearch \(p. 92\)](#).

For example, the following Boolean query searches the title field for the prefix *star*:

```
search?bq=title:'star*&return-fields=title
```

If you perform this search against the sample movie data, the response will contain movies such as *Stargate*, *Dark Star*, and *Starsky & Hutch*:

```
{ "rank": "-text_relevance",
  "match-expr": "(label title:'star*')",
  "hits": { "found": 34, "start": 0,
    "hit": [
      { "id": "tt1408101", "data": { "title": ["Untitled Star Trek Sequel"] } },
      { "id": "tt0111282", "data": { "title": ["Stargate"] } },
      { "id": "tt0335438", "data": { "title": ["Starsky & Hutch"] } },
      { "id": "tt0477095", "data": { "title": ["Starter for 10"] } },
      { "id": "tt1185834", "data": { "title": ["Star Wars: The Clone Wars"] } },
      { "id": "tt0069945", "data": { "title": ["Dark Star"] } },
      { "id": "tt0088172", "data": { "title": ["Starman"] } },
      { "id": "tt0844760", "data": { "title": ["Starship Troopers 3: Marauder"] } },
      { "id": "tt0092007", "data": { "title": ["Star Trek IV: The Voyage Home"] } },
      { "id": "tt0098382", "data": { "title": ["Star Trek V: The Final Frontier"] } }
    ]
  },
  "info": {
    "rid": "8a0620f6c72ff3e73c2a10e59f186fa89ba1fa67e3b160548fb2c7aa91bce7aebdc0b87198cf138a",
    "time-ms": 3,
  }
}
```



```
    "cpu-time-ms":0  
  }  
}
```

Note

When performing wildcard searches on text fields, keep in mind that Amazon CloudSearch tokenizes the text fields during indexing and performs basic stemming such as removing the trailing *s* from plural terms. Normally, the same text processing is performed on the search query. However, when you use the wildcard operator, no stemming is performed on the prefix. This means that a search for a prefix that ends in *s* won't match the singular version of the term. This can happen for any term that ends in *s*, not just plurals. For example, if you search the `actor` field in the sample movie data for `Gillanders`, there are three matching movies. If you search for `Gillander*`, you get the same three movies. However, if you search for `Gillanders*` there are no matches. This is because the term is stored in the index as `Gillander`, `Gillanders` does not appear in the index. For more information about how Amazon CloudSearch processes text and how it can affect searches, see [Text Processing in Amazon CloudSearch \(p. 247\)](#).

Searching for Phrases in Text Fields in Amazon CloudSearch

You can enclose a phrase in double quotes to match the complete phrase rather than the individual terms in the phrase. You can perform phrase searches with either the `q` or `bq` parameter. For example, the following queries produce the same results:

```
search?q="with love"  
search?bq="'with love' '
```

If you perform this search against the sample movie data, you'll notice that the results for the phrase search contain one less hit than a simple search for the terms *with love*:

```
{  
  "rank": "-text_relevance",  
  "match-expr": "(label \"with love\")",  
  "hits": {  
    "found": 4,  
    "start": 0,  
    "hit": [  
      { "id": "tt0062376" },  
      { "id": "tt0309530" },  
      { "id": "tt1179034" },  
      { "id": "tt0057076" }  
    ]  
  },  
  "info": { "rid": "7508c2e52f5c3c25eca625c994c1351ed8fed385d15bffaf9dd32aae31644e939b8656dcd8c96d09", "time-ms": 2, "cpu-time-ms": 0 }  
}
```

Searching Literal Fields in Amazon CloudSearch

When you search a literal field, Amazon CloudSearch returns only those documents that contain an exact match for the complete search string in the specified field. For example, if the `title` field is configured as a literal field and you search for "star", the value of the title field must be *star* to be considered a match—*star*

wars and *a star is born* will not be included in the search results. This differs from text fields, where the specified search terms can appear anywhere within the field in any order.

Literal fields are often used in conjunction with faceting to enable users to drill down into the results according to the faceted attributes. For more information about faceting, see [Getting and Using Facet Information in Amazon CloudSearch \(p. 96\)](#).

Searching Literal Fields with the Boolean Query Parameter in Amazon CloudSearch

To search literal fields, you must use the Boolean Query parameter, `bq`. To search a literal field, prefix the search string with the name of the literal field you want to search, followed by a colon. The search string must be enclosed in single quotes. For example:

```
search?bq=genre:'sci-fi'
```

This searches the `genre` field of each document and matches all documents whose `genre` field contains the value *sci-fi*. To be a match, the field value must be an exact match for the search string. For example, documents that contain the value *young adult sci-fi* in the `genre` field will not be included in the search results when you search for "sci-fi".

In addition to searching literal fields, the `bq` parameter can be used to search specific [text \(p. 87\)](#) and [uint \(p. 92\)](#) fields. To combine matches against multiple fields, you can use the Boolean operators *and*, *or*, and *not*. For more information about constructing Boolean queries, see [Constructing Boolean Search Queries in Amazon CloudSearch \(p. 93\)](#).

Note

You can only search literal fields that are search-enabled in your domain's configuration. For more information, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).

Using Wildcards in Literal Searches in Amazon CloudSearch

When searching literal fields, you can use the wildcard operator to find values that start with a particular string. For example, the `genre` field in the sample movie data is a literal field. If you search the `genre` field for "fi*", it will match all of the movies in the *film-noir* genre, but not the movies in the *sci-fi* genre. To be a match, the entire string up to the wildcard operator must match exactly.

Searching Uint Fields in Amazon CloudSearch

You can search uint fields for a particular value or a range of values. Uint fields are always search enabled.

Searching Uint Fields with the Boolean Query Parameter in Amazon CloudSearch

To search uint fields, you must use the Boolean Query parameter, `bq`. To search a uint field, prefix the value or range of values you want to search with the name of the uint field, followed by a colon. The integer value or range is *not* enclosed in single quotes.

In addition to searching uint fields, you can use the `bq` parameter to search specific [text \(p. 87\)](#) and [literal \(p. 91\)](#) fields. To combine matches against multiple fields, you can use the Boolean operators *and*,

or, and *not*. For more information about constructing Boolean queries, see [Constructing Boolean Search Queries in Amazon CloudSearch \(p. 93\)](#).

Searching for an Integer Value in Amazon CloudSearch

The syntax for searching a uint field for a particular value is *fieldname:integer*. For example, to search the sample movie data for movies released in 2010, you would specify:

```
search?bq=year:2010
```

Searching for a Range of Values in Amazon CloudSearch

The syntax for searching a uint field for a range of values is *<start>..<end>*. The start and ending values of the range are included. For example, to search the sample data set for movies released from 2008 to 2010, you would specify the range as *2008..2010*:

```
search?bq=year:2008..2010
```

Ranges can be open ended. For example, you could specify *year:2002..* to find all matching movies released from 2002 onward, or *..1970* to find all the movies released through 1970:

```
search?bq=year:2002..  
search?bq=year:..1970
```

Constructing Boolean Search Queries in Amazon CloudSearch

The *bq* parameter enables you to combine matches against fields using the Boolean operators *and*, *or*, and *not*. When constructing Boolean search queries, you use parentheses to control the order of evaluation of the expression. When part of an expression is enclosed in parentheses, that part is evaluated first. The resulting value is used in the evaluation of the remainder of the expression. At a minimum, the entire expression must be enclosed in a single set of parentheses.

For example, to search the title field for matches that either contain the string "star" or do not contain the string "wars":

```
search?bq=(or title:'star' (not title:'wars'))
```

You can use *and*, *or*, and *not* at the field level, and still use the *-* and *|* operators within the match expressions. For example, the following queries produce the same results:

```
search?bq=(or title:'star' title:'-wars')  
search?bq=(or title:'star' (not title:'wars'))
```

For more information about using Boolean operators in match expressions, see [Using Boolean Operators in Amazon CloudSearch Text Searches \(p. 89\)](#).

You can construct Boolean search queries to combine searches against multiple fields. For example:

```
search?bq=(and title:'star' genre:'drama')
```

Note

If you don't get the results you expect from a search request, check the `match-expr` in the response to see how Amazon CloudSearch parsed the match expression specified in the `bq` parameter.

Controlling How Search Results are Returned in Amazon CloudSearch

You can specify query parameters in your search request to:

- [Get results as XML \(p. 94\)](#)
- [Paginate the results \(p. 95\)](#)
- [Retrieve field values \(p. 95\)](#)
- [Sort the results \(p. 96\)](#)

Getting Results as XML in Amazon CloudSearch

By default, Amazon CloudSearch search responses are formatted in JSON. To get results as XML, specify the query parameter `results-type=xml` in your search request:

```
search?q=star+wars&results-type=xml
```

Search responses formatted in XML contain exactly the same information as a JSON response:

```
<results>
  <rank>-text_relevance</rank>
  <match-expr>(label 'star wars')</match-expr>
  <hits found="7" start="0">
    <hit id="tt1185834"/>
    <hit id="tt0076759"/>
    <hit id="tt0121765"/>
    <hit id="tt0080684"/>
    <hit id="tt0086190"/>
    <hit id="tt0120915"/>
    <hit id="tt0121766"/>
  </hits>
  <facets/>
  <info rid="b7c167f6c2da6d93501039ad23f00811361e4acf6ca09ec98ae60af47463dfe4ce2e5565e736aalf" time-ms="3" cpu-time-ms="0"/>
</results>
```

For detailed information about the JSON and XML response formats for search requests, see [Search Response \(p. 234\)](#).

Paginating Results in Amazon CloudSearch

By default, Amazon CloudSearch returns the top ten hits according to the specified ranking. To control the number of hits returned in a result set, you use the `size` parameter. To request the next set of hits beginning from a particular offset, you use the `start` parameter. Note that the result set is zero-based—the first result is at index 0.

For example, `search?q=-star` returns the first 10 hits that don't contain `star` in the default search field, starting at index 0. To get the next set of ten hits, set the `start` parameter to 10:

```
search?q=-star&start=10
```

If you want to retrieve 25 hits at a time, set the `size` parameter to 25. To get the first set of hits, you don't have to set the `start` parameter:

```
search?q=-star&size=25
```

For subsequent requests, use the `start` parameter to retrieve the set of hits you want. For example, to get the third batch of 25 hits specify:

```
search?q=-star&size=25&start=50
```

Retrieving Data from Index Fields in Amazon CloudSearch

By default, searches only return the IDs of the documents that match the search constraints. To include additional information, you can use the `return-fields` parameter to specify which index fields to include in the results.

Integer fields (`uint`) can always be returned in results. However, only text and literal fields that are result enabled in the domain configuration can be returned. You can also specify the default `text_relevance` score as a return field.

You can retrieve up to 2 KB of source data from an index field. All of the source data is indexed, but only the first 2 KB of data can be returned.

Note

Making fields result enabled increases the size of your index, which can increase the cost of running your domain. You should only store document data in the search index by making fields result-enabled when it's difficult or costly to retrieve the data using other means. Since it can take some time to apply document updates across the domain, critical data such as pricing information should be retrieved using the returned document IDs instead of returned from the index.

To retrieve source data for result-enabled fields, you specify the `return-fields` parameter in the query string. You can specify a single return field, or up to 10 fields as a comma-separated list. For example, to include the `actor`, `title`, and default `text_relevance` score in the search results:

```
search?q=star+wars&return-fields=actor,title,text_relevance
```

The specified fields will be included for each hit:

```
{
  "id": "tt1185834",
  "data": {
    "actor": ["Abercrombie, Ian", "Baker, Dee Bradley", "Burton, Corey",
              "Eckstein, Ashley", "Futterman, Nika", "Kane, Tom",
              "Lanter, Matt", "Taber, Catherine", "Taylor, James Arnold",
              "Wood, Matthew"],
    "text_relevance": ["308"],
    "title": ["Star Wars: The Clone Wars"]
  }
}
```

Sorting Results in Amazon CloudSearch

By default, results are sorted according to their `text_relevance` scores, with the highest-scoring documents listed first. You can use the `rank` parameter in your search requests to sort results alphabetically, numerically, or using your own custom rank expressions. When you use a field for ranking, documents without a value in that field are listed last. If you specify a comma separated list of fields or rank expressions, the first field or rank expression is used as the primary sort criteria, the second is used as the secondary sort criteria, and so on.

You can use any result-enabled *text* or *literal* field to sort results alphabetically. For example, `rank=actor` is specified in the following query to sort the results alphabetically by actor:

```
search?q=star+wars&return-fields=title&rank=actor
```

By default, results are listed in an *ascending* order. To sort in descending order, prefix the field name with `-` (minus sign):

```
search?q=star+wars&rank=-actor
```

You can use any `uint` field to sort results numerically. For example, specifying `rank=-year` will sort the results by year with the most recent year listed first:

```
search?q=star+wars&return-fields=title,year&rank=-year
```

Note

If you don't specify the rank option, it is set to `-text_relevance` by default so the highest-scoring documents are listed first.

You can also define custom rank expressions and use them to sort results. For more information about creating and using your own rank expressions, see [Customizing Result Ranking with Amazon CloudSearch \(p. 105\)](#).

Getting and Using Facet Information in Amazon CloudSearch

Topics

- [Getting Facet Information for Text and Literal Fields in Amazon CloudSearch \(p. 97\)](#)
- [Getting Facet Information for Uint Fields in Amazon CloudSearch \(p. 97\)](#)

- [Getting Facet Information for Particular Values in Amazon CloudSearch \(p. 98\)](#)
- [Sorting Facet Information in Amazon CloudSearch \(p. 99\)](#)
- [Using Facet Information in Amazon CloudSearch \(p. 100\)](#)

A facet is an index field that represents a category that you want to use to refine and filter search results. When you submit search requests to Amazon CloudSearch, you can request facet information to find out how many hits share the same value in a particular field. You can display this information along with the search results and use it to enable users to interactively refine their searches. (This is often referred to as faceted navigation or faceted search.)

You can get facet information for any uint field and facet-enabled text and literal fields by specifying the `facet` parameter in your search request.

Amazon CloudSearch also provides search parameters that enable you to control how facet values are returned and sorted. You can select which facets to retrieve, limit the number of facet values returned, and control the sorting of the facet values for each field.

Getting Facet Information for Text and Literal Fields in Amazon CloudSearch

When you request facet information for a text or literal field, Amazon CloudSearch returns facet counts for the top 40 values in the specified field. You can include the `facet-FIELD-top-n` parameter to control the number of facet values that are returned for a particular field.

Note

To get facet information for a text or literal field, the field must be configured to enable faceting. For more information, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).

For example, the following request gets facet counts for the top five most-frequently-occurring values in the `genre` field:

```
search?bq=title:'star'&facet=genre&facet-genre-top-n=5
```

The response includes the returns the facet information after the list of hits.

```
"facets":{
  "genre":{"constraints":[
    {"value":"Sci-Fi","count":20},
    {"value":"Action","count":18},
    {"value":"Adventure","count":16},
    {"value":"Thriller","count":10},
    {"value":"Fantasy","count":5}
  ]}
}
```

Getting Facet Information for Uint Fields in Amazon CloudSearch

When you request facet information for a uint field, Amazon CloudSearch returns the min and max values for the field. For example, when you specify `facet=year`, you get the first and last year that appears in the `year` field:

```
"facets": {"year": {"min": 1974, "max": 2012}}
```

To drill down into particular bins of integers, you use the `facet-FIELD-constraints` parameter. For more information, see [Getting Facet Information for Particular Values in Amazon CloudSearch \(p. 98\)](#).

Getting Facet Information for Particular Values in Amazon CloudSearch

The `facet-FIELD-constraints` parameter controls which facet values are returned for the specified facet. You specify the facet values you want to count as a comma-separated list. The values must be enclosed within single quotes. Note that the facet values are case sensitive:

`facet-genre-constraints='drama'` is not the same as `facet-genre-constraints='Drama'`.

Note

If commas occur in a facet value you want to use as a constraint, the comma must be escaped with a backslash. For example, `facet-actor-constraints='Bai\, Ling', 'Bryant\, Gene'`.

For example, to find out how many documents have *Drama* or *Sci-Fi* in the `genre` field, you'd set `facet-genre-constraints='Drama', 'Sci-Fi'`:

```
search?q=star&facet=genre&facet-genre-constraints='Drama', 'Sci-Fi'
```

In the response, the counts are only shown for the specified constraints:

```
facets": {"genre":
  { "constraints": [
    { "value": "Sci-Fi", "count": 20 },
    { "value": "Drama", "count": 4 }
  ]
}
```

The `facet-FIELD-constraints` parameter can also be used with `uint` fields. You can specify individual values, as well as ranges of values, which enables you to do range-based binning. You can use the `min` and `max` values returned when you don't specify any constraints to calculate the ranges, and then get facet counts for each of those ranges with a subsequent search.

The values and ranges are specified as a comma-separated list. For example, the following request gets facet counts for documents with a year value of 2000, 2001, 2002 through 2004, and all documents with year greater than or equal to 2005:

```
search?q=star&facet=year&facet-year-constraints=2000,2001,2002..2004,2005..
```

By default, the response shows the constraints with the highest counts first:

```
"facets": {
  "year": { "min": 1970, "max": 2012,
    "constraints": [
      { "value": "2005..", "count": 8 },
      { "value": "2002..2004", "count": 2 },
      { "value": "2001", "count": 1 }
    ]
  }
}
```



```
}  
}
```

Sorting Facet Information in Amazon CloudSearch

You can use the `facet-FIELD-sort` parameter to control how the facet information is sorted in the search results. Amazon CloudSearch supports four sorting options:

- `alpha`—sort the facet values alphabetically. The facet values are always sorted ascending order when using the `alpha` option.
- `count`—sort the facet values by their counts. The facet values are always sorted in descending order when using the `count` option.
- `max`—sort the facet values according to the maximum values in the specified field. This option is specified as `max(FIELD)`. By default, the facet values are sorted in ascending order. To sort in descending order, prefix the `max` option with a `-` (minus sign): `-max(FIELD)`.
- `sum`—sort the facet values according to the sum of the values in the specified field (in ascending order). This option is specified as `sum(FIELD)`. By default, the facet values are sorted in ascending order. To sort in descending order, prefix the `sum` option with a `-` (minus sign): `-sum(FIELD)`.

By default, facet information is sorted by facet counts. The `-` (minus) prefix cannot be used to reverse the sort order when using the `alpha` or `count` options.

To sort values for a facet field alphabetically

- Specify `facet-FIELD-sort=alpha`:

```
search?bq=title:'star'&facet=genre&facet-genre-sort=alpha
```

To sort values for a facet field using the value of a uint field or rank expression

- Specify `facet-FIELD-sort=max(FIELD)`. When you use the `max` option, the score used for sorting is the maximum value in the specified field across all matching documents with that facet value. By default, the values are sorted in ascending order. You can prefix the `max` option with a `-` (minus sign) to reverse the order.

For example, you could use the default `text_relevance` score to sort the facet values. In the following request, the facet value that has the matching document with the highest `text_relevance` score is listed first:

```
search?bq=title:'star'&facet=genre&facet-genre-sort=-max(text_relevance)
```

The maximum `text_relevance` score for each facet value is displayed in the facet information:

```
"facets":  
  {"genre":  
    {"constraints": [  
      {"value": "Action", "count": 18, "score": 288},  
      {"value": "Adventure", "count": 16, "score": 288},  
      {"value": "Sci-Fi", "count": 20, "score": 288},  
      {"value": "Animation", "count": 1, "score": 282},  
      {"value": "Comedy", "count": 4, "score": 282},
```

```
{
  { "value": "Thriller", "count": 10, "score": 282 },
  { "value": "Biography", "count": 1, "score": 276 },
  { "value": "Drama", "count": 3, "score": 276 },
  { "value": "Romance", "count": 1, "score": 276 },
  { "value": "Mystery", "count": 3, "score": 274 },
  { "value": "Music", "count": 1, "score": 272 },
  { "value": "Fantasy", "count": 5, "score": 271 },
  { "value": "Family", "count": 3, "score": 270 }
}
```

To sum the values in a field and use the resulting score to sort the facet values

- Specify `facet-FIELD-sort=sum(FIELD)`. When you use the `sum` option, the score used for sorting is the sum of the values in the specified field for all matching documents with that facet value. By default, the values are listed in ascending order. For example:

```
search?bq='state'&facet=chief&facet-chief-sort=sum(majvotes)
```

The sum is displayed in the facet information as the score for the facet value:

```
facets": {
  "chief": {
    "constraints": [
      { "value": "Roberts", "count": 116, "score": 869 },
      ...
      { "value": "Warren", "count": 712, "score": 4932 }
    ]
  }
}
```

Note

You can prefix the `sum` option with a `-` (minus sign) to list the values in descending order.

Using Facet Information in Amazon CloudSearch

You can display facet information to enable users to more easily browse search results and zero in on the information they are interested in. For example, if a user is trying to find one of the Star Trek movies, but can't remember the full title, he might start by searching for "star". If you want to display top facets for *actor* and *genre*, you would specify those facets in the query, along with the number of facet values you want to retrieve for each facet:

```
search?q=star&facet=actor,genre&facet-actor-top-n=10&facet-genre-top-n=5&size=5&results-type=xml
```

This gives you the following information in the search response:

```
<results xmlns="http://cloudsearch.amazonaws.com/
2011-02-01/results">
```

```
<rank>-text_relevance</rank>
<match-expr>(label 'star')</match-expr>
<hits found="26" start="0">
  <hit id="tt1408101"/>
  <hit id="tt0069945"/>
  <hit id="tt1185834"/>
  <hit id="tt0092007"/>
  <hit id="tt0098382"/>
</hits>
<facets>
  <facet name="actor">
    <constraint value="Doohan, James" count="7"/>
    <constraint value="Koenig, Walter" count="7"/>
    <constraint value="Nimoy, Leonard" count="7"/>
    <constraint value="Kelley, DeForest" count="6"/>
    <constraint value="Nichols, Nichelle" count="6"/>
    <constraint value="Shatner, William" count="6"/>
    <constraint value="Takei, George" count="6"/>
    <constraint value="Daniels, Anthony" count="5"/>
    <constraint value="Burton, LeVar" count="4"/>
    <constraint value="Dorn, Michael" count="4"/>
  </facet>
  <facet name="genre">
    <constraint value="Sci-Fi" count="20"/>
    <constraint value="Action" count="18"/>
    <constraint value="Adventure" count="17"/>
    <constraint value="Thriller" count="10"/>
    <constraint value="Fantasy" count="5"/>
  </facet>
</facets>
<info rid="3c5a461d28b76874a756e4d419a38646955da47864afeeef172add882f
712bb0b7c9e486627e07e2" time-ms="3" cpu-time-ms="0"/>
</results>
```

Using the document ids, you can retrieve the data you want to display for each hit from a separate system. By displaying the facet information, you can provide a way for the user to zero in on the movie he's looking for. For example, he might click "William Shatner" in the list of actors to see the subset of movies that William Shatner appeared in. To retrieve the subset, you can use the `bq` search parameter to perform a fielded search against the actor field and find the matches that contain *star* in any text field and *William Shatner* in the actor field.

Note

In this example, both the actor and genre fields have configured as facets. If you want to try out these queries with the sample imdb-movie data, you'll need to modify your movie domain's indexing options to configure the actor field as a facet. For more information, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).

```
search?bq=(and 'star' actor:'William Shatner')&facet=actor,genre
&facet-actor-top-n=10&facet-genre-top-n=5&size=5
&results-type=xml
```

This retrieves the subset of hits along with the actor and genre facet information:

```
<results>
  <rank>-text_relevance</rank>
  <match-expr>(and 'star' actor:'William Shatner')</match-expr>
```

```
<hits found="6" start="0">
  <hit id="tt0092007"/>
  <hit id="tt0098382"/>
  <hit id="tt0088170"/>
  <hit id="tt0079945"/>
  <hit id="tt0084726"/>
</hits>
<facets>
  <facet name="actor">
    <constraint value="Doohan, James" count="6"/>
    <constraint value="Kelley, DeForest" count="6"/>
    <constraint value="Koenig, Walter" count="6"/>
    <constraint value="Nichols, Nichelle" count="6"/>
    <constraint value="Nimoy, Leonard" count="6"/>
    <constraint value="Shatner, William" count="6"/>
    <constraint value="Takei, George" count="6"/>
    <constraint value="Butrick, Merritt" count="2"/>
    <constraint value="Lenard, Mark" count="2"/>
    <constraint value="Adamson, Joseph" count="1"/>
  </facet>
  <facet name="genre">
    <constraint value="Sci-Fi" count="6"/>
    <constraint value="Action" count="5"/>
    <constraint value="Adventure" count="5"/>
    <constraint value="Thriller" count="4"/>
    <constraint value="Mystery" count="2"/>
  </facet>
</facets>
<info
rid="ccd66a5219f938d2d27598352059d8c34094e7b0695b7c51dc91631555cb382dc17ef8064dbc9fdd"
  time-ms="3" cpu-time-ms="0"/>
</results>
```

At this point, the user might remember that the movie he's trying to find also had Joseph Adamson in it and click on Joseph Adamson in the actor list. Again, you would use his selection to further refine the query:

```
search?bq=(and 'star' actor:'William Shatner' actor:'Adamson, Joseph')
&return-fields=title&facet=actor,genre&facet-actor-top-n=10
&facet-genre-top-n=5&size=5&results-type=xml
```

Now, there's just a single match that you can display to the user *Star Trek IV: The Voyage Home*:

```
<results>
  <rank>-text_relevance</rank>
  <match-expr>(and 'star' actor:'William Shatner' actor:'Adamson,
Joseph')</match-expr>
  <hits found="1" start="0">
    <hit id="tt0092007">
      <d name="title">Star Trek IV: The Voyage Home</d>
    </hit>
  </hits>
</facets>
...
```

```
</facets>  
</results>
```

Tuning Search Requests in Amazon CloudSearch

One of the things that can impact the performance and cost of running your search domain is how resource intensive your search requests are to process. In general, searches that return a large volume of hits and complex Boolean queries are more resource intensive than simple text queries that match a small percentage of the documents in your search domain.

If you are experiencing slow response times, frequently encountering internal server errors (typically 507 or 509 errors), or seeing the number of instance hours your search domain is consuming increase without a substantial increase in the volume of data you're searching, fine-tuning your search requests can help reduce the processing overhead. This section reviews what to look for and steps you can take to tune your search requests.

Analyzing Query Latency

Before you can start tuning your requests, you need to analyze your current search performance. Log your search requests and the corresponding response times so you can see which requests take the longest to process. Slow searches can disproportionately affect overall performance by tying up your search domain's resources. Optimizing the slowest search requests will speed up *all* of your searches.

Reducing the Number of Hits

Query latency is directly proportional to the number of matching documents. Searches that match the most documents are generally the slowest.

Eliminating two types of searches that commonly result in a huge number of matching documents can significantly improve overall performance:

- Negative queries that match every document in your corpus. While this can be a convenient way to list all of the documents in your domain, it's a very resource intensive query. If you have a lot of documents, not only can it cause other requests to time out, it's likely to time out itself.
- Wildcard searches with only one or two characters specified. If you're using this type of search to provide instant results as the user types, wait until the user has entered at least two characters before you start submitting requests and displaying the possible matches.

To reduce the number of documents that match your requests, you can also:

- Eliminate irrelevant words from your corpus so they aren't using for matching. The easiest way to do this is to add them to your domain's stopwords list. Alternatively, you can preprocess your SDF data to strip out irrelevant words. Eliminating irrelevant words also has the benefit of reducing the size of your index.
- Explicitly filter the results based on the value of a particular field. In some cases, you might find that filtering on text or literal fields is actually faster than filtering on uint fields.

If you still have requests that match a lot of documents, you can reduce latency by minimizing the amount of processing that needs to be done on the result set:

- Minimize the facet information that you're requesting. Generating the facet counts adds to the time it takes to process the request and increases the likelihood that other requests will time out. If you do request facet information, keep in mind that the more facets you specify, the longer it will take to process the request.
- Skip specifying your own ranking options. The additional processing required to rank the results increases the likelihood that requests will time out. If you must customize how the results are ranked, using a field value for ranking is generally faster than using a rank expression.

Also keep in mind that returning a large amount of data in the search results can increase the transport time and affect query latency. Minimizing the number of return fields you use can improve performance as well as reduce the size of your index.

Simplifying Boolean Queries

The more clauses there are in a Boolean query, the longer it takes to process the query. This is especially true if the query contains a large number of OR clauses.

If you have complex Boolean queries that don't perform well, you need to find a way to reduce the number of clauses. In some cases, you might simply be able to set a limit or reformulate the query. In others, you might need to modify your domain configuration to accommodate simpler queries.

Customizing Result Ranking with Amazon CloudSearch

Topics

- [Configuring Rank Expressions in Amazon CloudSearch \(p. 106\)](#)
- [Using Relative Field Weighting to Customize Text Relevance in Amazon CloudSearch \(p. 110\)](#)
- [Comparing Rank Expressions in Amazon CloudSearch \(p. 111\)](#)
- [Ranking Search Results in Amazon CloudSearch \(p. 113\)](#)
- [Constraining Search Results in Amazon CloudSearch \(p. 113\)](#)

By default, search results are ranked according to their relevance to the search request. A document's default `text_relevance` score takes into account the proximity of the search terms and the frequency of those terms within the document compared to how common the term is across all documents in the domain. To change how search results are ranked, you can:

- Use any text or literal field to [sort results \(p. 96\)](#) alphabetically.
- Use any uint field to [sort results \(p. 96\)](#) numerically.
- Use a custom rank expression to rank results.

To define a rank expression, you construct a numeric expression using uint fields, other rank expressions, a document's `text_relevance` score, and numeric operators and functions. To use a rank expression to customize result ranking, you use the `rank` parameter in your search requests to specify the rank expression you want to use to order the results.

You can also use rank expressions in your search requests to set thresholds for search results through the `t-FIELD` parameter.

Rank expressions configured as part of your domain configuration can be referenced in any search request to rank the results, set thresholds for the results, and be returned in the search results. To test and tune rank expressions, you can define and use them directly within search requests. To facilitate tuning, the Amazon CloudSearch console enables you to easily compare how results are ranked using different rank expressions.

Configuring Rank Expressions in Amazon CloudSearch

A rank expression is a numeric expression specified using an expression syntax that's based on JavaScript expressions. You can construct rank expressions using:

- Uint fields
- Other rank expressions
- The `text_relevance` score that's computed for each matching document
- A specialized function called `cs.text_relevance` that enables you to control the relative importance of text fields by specifying field weights that are used when computing `text_relevance` scores for the rank expression. (For more information, see [Using Relative Field Weighting to Customize Text Relevance in Amazon CloudSearch \(p. 110\)](#).)
- Integer, floating point, hex and octal literals
- Arithmetic operators: `+` `-` `*` `/` `%`
- Bitwise operators: `|` `&` `^` `~` `<<` `>>` `>>>`
- Boolean operators (including the ternary operator): `&&` `||` `!` `?:`
- Comparison operators: `<` `<=` `==` `>=` `>`
- Common mathematic functions: `abs` `ceil` `erf` `exp` `floor` `lgamma` `ln` `log2` `log10` `max` `min` `sqrt` `pow`
- Trigonometric library functions: `acosh` `acos` `asinh` `asin` `atanh` `atan` `cosh` `cos` `sinh` `sin` `tanh` `tan`
- Miscellaneous functions: `rand`, `time`, `min`, `max`

JavaScript order of precedence rules apply for operators. Shortcut evaluation is used for logical operators—the second argument is only evaluated if the value of the expression cannot be determined after evaluating the first argument. For example, in the expression `a || b`, `b` is only evaluated if `a` is not true.

Rank expressions always return an integer value from 0 to the maximum unsigned 32-bit integer value. If the expression is invalid or evaluates to a negative value, it returns 0. If the expression evaluates to a value greater than the maximum, it returns the maximum value. Intermediate results are calculated as double-precision floating point values and the return value is rounded to the nearest integer.

Rank expression names must begin with a letter and be at least 3 and no more than 64 characters long. The following characters are allowed: a-z (lower-case letters), 0-9, and `_` (underscore). The names "body", "docid", and "text_relevance" are reserved names and cannot be specified as rank expression names.

For example, if you define a uint field named *popularity* for your domain, you could use that field in conjunction with the default `text_relevance` score to construct a custom rank expression. The following expression bases 30% of a document's rank score on the popularity field, and 70% of its rank score on its default `text_relevance` score. (The default `text_relevance` score is in the range 0-1000, the popularity field is assumed to have values in the range 0-10000, and the expression returns a value in the range 0-1000.)

```
((0.3*popularity)/10.0)+(0.7*text_relevance)
```

For more information about using rank expressions to sort search results, see [Sorting Results in Amazon CloudSearch \(p. 96\)](#).

In addition to specifying how you want to rank results, the Amazon CloudSearch Search API also enables you to specify threshold constraints. A threshold constraint can be based on the value of a uint field, or

on the value of a rank expression. For example, if your documents have an `available_on` field that specifies a date as an epoch uint value, you could define a rank expression to exclude documents whose `available_on` value is later than the current time:

```
(time() > available_on)?1:0
```

For more information about using rank expressions to constrain search results, see [Constraining Search Results in Amazon CloudSearch \(p. 113\)](#)

Defining Amazon CloudSearch Rank Expressions in Search Requests

You can define a query time rank expression by specifying the `rank-RANKNAME` parameter. You can use the rank expression to rank results for that request or set a threshold for the search results. The rank expression can also be specified as a return field or used in the definition of another rank expression. Existing rank expressions can be overridden by specifying the name of the expression you want to override as the `RANKNAME`.

When you define a rank expression within a search request, it is not stored as part of your domain configuration. If you want to use the rank expression in other requests, you must define the rank expression in each request or add the rank expression to your domain configuration, as described in [Defining Amazon CloudSearch Rank Expressions in a Domain Configuration \(p. 107\)](#).

Defining a rank expression in a search request enables you to quickly test and tune rank expressions. Once you have finished tuning a rank expression, you should configure it for use in all requests by adding it to your domain configuration. Defining the rank expression in every request increases the request overhead and can result in slower response times and potentially increase the cost of running your domain.

You can define and use multiple rank expressions in a search request. The definition of a rank expression can reference other rank expressions defined within the request, as well as expressions configured as part of the domain configuration. For example, the following request creates two rank expressions that are used to rank the results and returns one of them in the search results:

```
search?q=terminator&rank-expression1=sin(text_relevance)&rank-expression2=cos(text_relevance)&rank=expression1,expression2&return-fields=title,text_relevance,expression2
```

Defining Amazon CloudSearch Rank Expressions in a Domain Configuration

To configure a rank expression for use in all requests, you add the rank expression to your domain configuration. You can configure rank expressions using the `cs-configure-ranking (p. 108)` command, from the [Amazon CloudSearch console \(p. 108\)](#), or using the `DefineRankExpression (p. 109)` configuration action.

When you add a rank expression to your domain configuration, it takes some time for the change to be processed and the new rank expression to become active. To quickly test changes to a rank expression, you can define and use the expression directly in a search request, as described in [Defining Amazon CloudSearch Rank Expressions in Search Requests \(p. 107\)](#). Once you have finished testing and tuning a rank expression, you should add it to your domain configuration. Adding a rank expression to the domain configuration reduces the overhead of specifying it in every request, and helps maximize response times and minimize costs.

Command Line Tools

You use the `cs-configure-ranking` (p. 146) command to define rank expressions for a domain.

To configure a rank expression

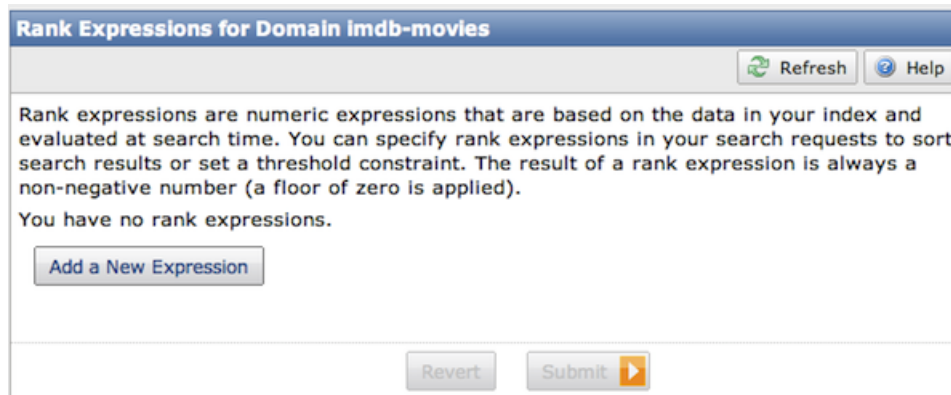
- Run the `cs-configure-ranking` command to define a new rank expression. You specify a name for the expression with the `--name` option, and the numeric expression that you want to evaluate with the `--expression` option.

```
cs-configure-ranking --name popularhits --expression '((0.3*popularity)/10.0)+(0.7*text_relevance)'
```

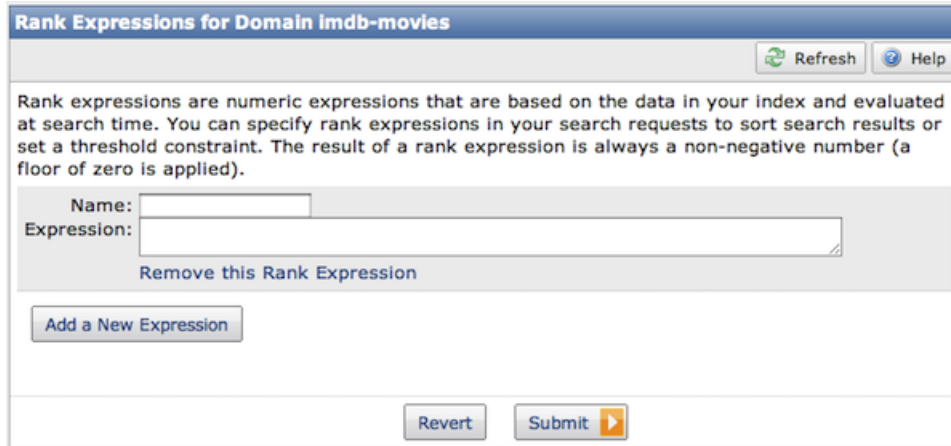
AWS Management Console

To configure a rank expression

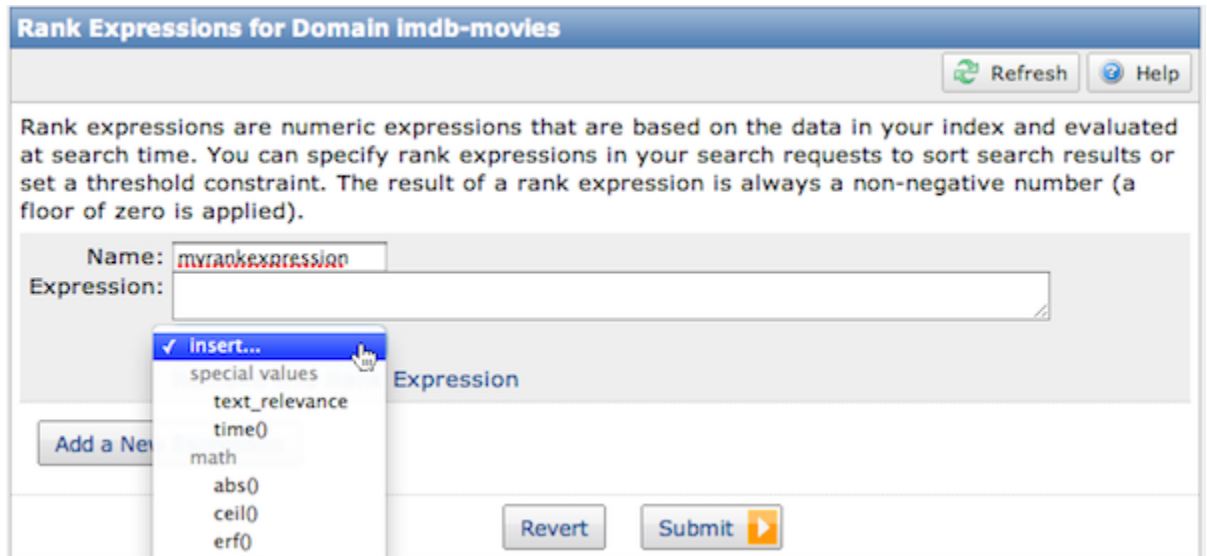
1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Rank Expressions** link.
3. In the **Rank Expressions** pane, click the **Add a New Rank Expression** button. The button is below the list of expressions configured for the domain.



4. Enter a name for the new expression in the **Name** field.



5. Enter the numerical expression you want to evaluate at search time in the **Expression** field. You can use the **insert...** menu to insert special values and mathematic and trigonometric functions.



6. Click **Add a New Expression** to configure additional rank expressions.
7. Click **Submit** to save your changes.

API

You use the [DefineRankExpression](#) (p. 169) configuration action to specify rank expressions.

The name you specify in the `RankExpression.RankName` option is how you reference the expression in your search requests. You specify the numeric expression that you want to evaluate for each search result in the `RankExpression.RankExpression` option.

For example:

```
https://cloudsearch.us-east-1.amazonaws.com  
?Action=DefineRankExpression
```

```
&DomainName=movies
&RankExpression.RankExpression=((0.3*year)/10.0)+((0.7*text_relevance))
&RankExpression.RankName=popularhits
&Version=2011-02-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20120403/us-east-1/cloudsearch/aws4_re
quest
&X-Amz-Date=2012-04-03T00:16:02.684Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=30205ede7907cf8a3fc41172fc63e323136a083b0967f96196bdea53f60d3cf3
```

Using Relative Field Weighting to Customize Text Relevance in Amazon CloudSearch

You can use the `cs.text_relevance` function in a rank expression to calculate `text_relevance` scores using custom field weights. This enables you to control how much matches in particular text or literal fields affect a document's `text_relevance` score. By default all fields are considered equally important. The `cs.text_relevance` function enables you to assign different weights to different fields so you can boost the `text_relevance` score of documents with matches in key fields such as the title field, and minimize the impact of matches in less important fields. When you use the `cs.text_relevance` function to specify field weights within a rank expression, those weights are only applied when evaluating that rank expression.

The `cs.text_relevance` function takes a JSON object that can contain two members:

- `weights`—a JSON object that defines weights for one or more source fields. Note that you specify weights for the *source fields* defined in your SDF data. By default, a source field is mapped to the index field with the same name, but source fields can also be explicitly mapped to any index field. For more information, see [Adding Sources for an Amazon CloudSearch Index Field \(p. 59\)](#).
- `default_weight`—the default weight to use for fields for which no weight is specified. (If the `default_weight` is not specified, the default weight for all fields is 1.0.)

Both the `weights` and `default_weight` members are optional. The specified weights must be in the range 0.0 to 10.0, inclusive. Weights must be specified as a numeric value, you cannot use mathematical functions or other rank expressions to define a field weight. Keep in mind that a document's `text_relevance` score is a value from 0 to 1000 (inclusive). If the `text_relevance` score calculated for a document is greater than 1000, the document's score is set to 1000. If you specify a large default weight, it increases the likelihood that `text_relevance` scores will exceed 1000 and be clipped.

For example, if you want matches within the title field to be ranked higher than matches within the description field, you could create a rank expression that sets the weight of the title field to 1.5 and the weight of the description field to 0.5:

```
cs.text_relevance({"weights": {"title": 1.5, "description": 0.5}})
```

Note that the keys can be enclosed in single or double quotes, or omitted entirely. To simplify this example, you can specify it as:

```
cs.text_relevance({weights: {title: 1.5, description: 0.5}})
```

Because the `default_weight` is not specified, the weight of all other fields defaults to 1.0. If you want all fields other than the title field to be treated the same, you could set the weight of the title field to 1.5, and set the `default_weight` to 0.5:

```
cs.text_relevance({weights: {title: 1.5}, default_weight: 0.5})
```

In your rank expressions, you can use the `cs.text_relevance` function in conjunction with `uint` fields, other rank expressions, a document's default `text_relevance` score, and the standard numeric operators and functions. For example, you could create a custom rank expression that's based on popularity (which could be defined as a `uint` field, or as another rank expression) and boosts the importance of the title field by setting its weight to 4.0:

```
((0.3*popularity)/10.0)+(0.7*cs.text_relevance({weights: {title:4.0}}))
```

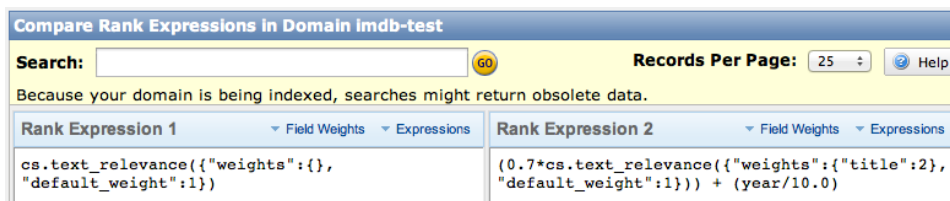
The `cs.text_relevance` function can only be used in the definition of a rank expression. To rank or threshold results using the specified weights, you specify the name of the rank expression. To rank or threshold results based on the default `text_relevance` score, you specify `text_relevance`.

Comparing Rank Expressions in Amazon CloudSearch

The Amazon CloudSearch console enables you to easily compare rank expressions to see how changes to the expression and field weights affect how search results are sorted.

To compare rank expressions

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, select a domain, and then click the domain's **Compare Rank Expressions** link.
3. In the **Compare Rank Expressions** pane, specify the rank expressions you want to compare. In each Rank Expression editor, you can add a new rank expression or select an existing expression from the **Expressions** menu. New rank expressions are validated when you submit a search request. If errors are detected, the rank expression is highlighted in red and a description of the problem is displayed.



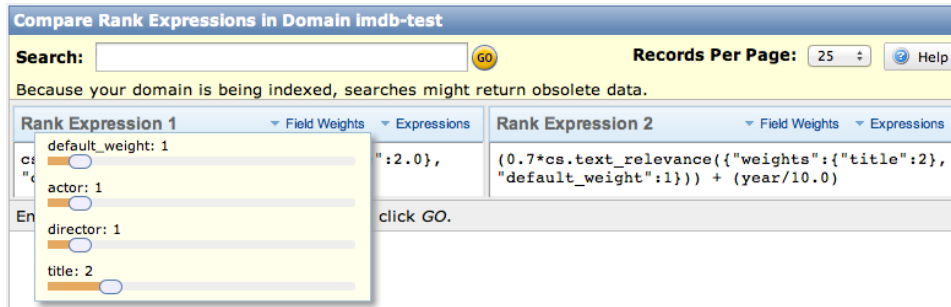
4. Specify the field weights to use for each rank expression by adjusting the sliders in the **Field Weights** menu. You can also edit the field weights directly in the rank expression. Field weights must be in the range 0.0 to 10.0, inclusive. By default, the weight for all fields is set to 1.0. Setting individual field weights enables you to control how much matches in particular text or literal fields affect a document's `text_relevance` score. You can also change the default weight.

Note

Adjusting field weights only affects result ranking if the rank expression uses the `cs.text_relevance` function. When you adjust the weights, the `cs.text_relevance` function is automatically inserted in the rank expression. You can modify the expression to change

Amazon CloudSearch Developer Guide Comparing Rank Expressions

how the customized text_relevance score contributes to a document's overall ranking. For more information, see [Using Relative Field Weighting to Customize Text Relevance in Amazon CloudSearch](#) (p. 110).



- Enter the terms you want to search for in the **Search** field and click **GO**. The results for the search are ranked using the specified rank expressions and weights. The results are refreshed whenever you make changes to the rank expressions or weights.

The search results for the two rank expressions are shown side-by-side. (If the rank expression is empty, the results are sorted according to the default text_relevance score.) Four icons are used to highlight the differences:

 **Green Up Arrow**

The document is ranked higher in the search results using the second rank expression.

 **Red Down Arrow**

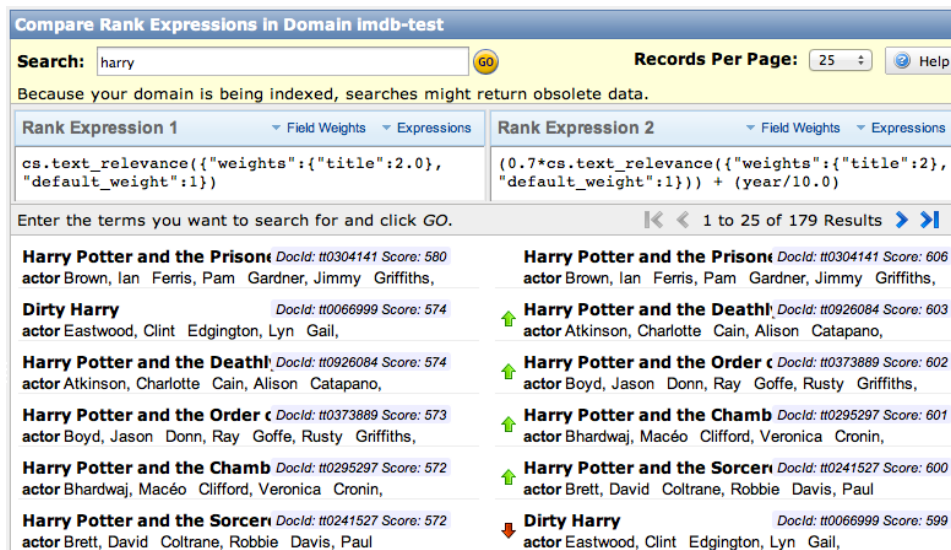
The document is ranked lower in the search results using the second rank expression.

 **Yellow Plus**

The document is included in the search results using the second rank expression, but was omitted from the search results using the first rank expression.

 **Red Minus**

The document was omitted from the search results using the second rank expression, but was included in the search results using the first rank expression.



Note

You can save rank expressions to your domain configuration directly from the **Compare Rank Expressions** pane. To save either rank expression, select **Save Expression** from the **Expressions** menu, enter a name for the expression, and click **OK**.

Ranking Search Results in Amazon CloudSearch

To use a rank expression to control how search results are ordered, you use the `rank` option in your search requests to specify the name of the rank expression. You must negate the rank expression name by prepending `-` (minus) if you want to sort the results in descending order. For example:

```
search?q=star+wars&return-fields=title&rank=-popularhits
```

If you do not specify the `rank` option, the results are ranked using the documents' default `text_relevance` scores with the highest-scoring documents listed first. (This is equivalent to specifying `rank=-text_relevance`.)

Constraining Search Results in Amazon CloudSearch

To use a rank expression as a threshold constraint for the search results, you use the `threshold` option in your search requests. The threshold option is specified as `t-RANKNAME`. When using the threshold option, you can specify an exact value, or a range of values. Ranges are specified as a pair of nonnegative integers separated by two dots, for example `50..100`. To specify an open-ended upper or lower limit, you can omit one value. For example, you could specify `50..` to set the threshold to 50 or greater.

For example, to only include results for which the `is_available` rank expression evaluates to 1 (true), you could use the `is_available` rank expression as a threshold:

```
search?q=star+wars&return-fields=title&&rank=-custom_relevance&t-is_available=1
```

In this example, the results are ranked using a rank expression called `custom_relevance`.

You can also set thresholds based on the default `text_relevance` score. A document's `text_relevance` score is in the range 0-1000. To limit the results to documents that have a `text_relevance` score of at least 500, you could specify:

```
search?q=star+wars&return-fields=title&t-text_relevance=500..
```


Searching and Ranking Results by Geographic Location in Amazon CloudSearch

Topics

- [Representing Locations in Amazon CloudSearch \(p. 114\)](#)
- [Searching Within an Area in Amazon CloudSearch \(p. 115\)](#)
- [Sorting Results by Distance in Amazon CloudSearch \(p. 116\)](#)

Although Amazon CloudSearch does not have a native type to support latitude and longitude, you can implement geographically-based searching and sorting by representing latitude and longitude as integers.

Note

Looking for sample code? The [CloudSearchGeoSpatial](#) sample application shows how to use Amazon CloudSearch to implement location-based search in an HTML/Javascript application.

Representing Locations in Amazon CloudSearch

To associate a location with a search document, you can represent the latitude and longitude in linear units and store those values as uint fields. This requires translating latitude and longitude from a spherical coordinate system to a Cartesian coordinate system. (For more information, see [Expressing Latitude and Longitude as Linear Units](#) in the Geographic Coordinate System Wikipedia article.)

To convert a location's latitude and longitude from degrees to meters, you first need to determine the number of meters per degree. Then, it's simply a matter of multiplying the location's latitude or longitude value in degrees by the number of meters per degree.

Converting Latitudes

A location's latitude in meters is the number of meters per degree of latitude multiplied by the location's latitude in degrees. To calculate the number of meters per degree of latitude, you use the following formula:


```
meters per degree of latitude = 2 rearth/360
```

The radius of the earth varies, but using the mean of the equatorial and polar radii (6,367,444 meters) provides a reasonable estimate for a degree of latitude. Using this value, the number of meters per degree of latitude is $(2\pi * 6,367,444)/360 = 111,133$ meters (rounded off).

To represent a location's latitude in meters, you simply multiply the latitude in degrees by 111,133. For example, New York City is at 40° N, 74° W. Since the latitude of New York City is 40°, the latitude in meters is $40 * 111,133 = 4,445,320$ meters.

Converting Longitudes

The number of meters per degree of longitude varies from the equator to the poles, so you need to take the location's latitude into account to calculate the number of meters per degree of longitude. To do this, you factor in the cosine of the latitude:

```
meters per degree of longitude = (meters per degree of latitude) * cos(latitude)
```

For example, the number of meters per degree of longitude at New York's latitude of 40° is $111,133 * \cos(40) = 85,133$ meters.

To represent a location's longitude in meters, you multiply the longitude in degrees by the number of meters per degree of longitude. The longitude of New York City in degrees is 74° W, so the longitude in meters is $74 * 85,133 = 6,299,842$ meters.

Note

In the Western hemisphere (West of the Prime Meridian), longitude values are specified in the range 0 to -180 degrees—74° W is -74°. Similarly, Southern latitudes are represented as negative values. In Amazon CloudSearch, numeric values must be represented as unsigned integers. You can either use the absolute value of the longitude and latitude (if you know the context of the location data), or add 180 degrees to all latitudes and longitudes to represent them as positive values from 0 to 360 degrees.

Searching Within an Area in Amazon CloudSearch

You can use integer range searching to find matches within a rectangular area. First, determine the top, left, bottom, and right coordinates of the rectangle and represent them as unsigned integers, as described in [Representing Locations in Amazon CloudSearch \(p. 114\)](#). Then, use the `bq` parameter in your search to specify these coordinates as a range for latitude and longitude.

Note

For example:

```
bq=(and 'toyota dealers' longitude:left..right latitude:bottom..top)
```

In this example, *top*, *left*, *bottom*, and *right* are the linear representations of the boundary of the rectangle.

Note

If your locations are constrained to a particular locale and you choose to work with absolute values, keep in mind that you must specify the smaller value first when performing a range

search. Alternatively, you can add 180 degrees to all latitudes and longitudes to represent them as positive values from 0 to 360 degrees.

Sorting Results by Distance in Amazon CloudSearch

You can use [query time rank expressions](#) (p. 107) to pass in coordinates from the user's location and rank results using a distance function such as:

```
sqrt(pow(abs(user-lat - latitude), 2) + pow(abs(user-lon - longitude), 2))
```

For example, if your user is in New York City, you can define and use the following rank expression in your search request to sort the results by distance. (Note that a current bug requires that you take the absolute value of the differences.)

```
rank-geo=sqrt(pow(abs(4524691 - latitude),2) + pow(abs(6299768 - longitude),2))
```

To sort by distance from the user's location, add `rank=geo` to your search request. (The results are ranked in ascending order to get the closest results ranked first.)

Note

This rank expression provides a fast approximation for ranking by distance, but the further from the equator the location, the less accurate it becomes. If your application requires a more accurate distance calculation for ranking, you can use a rank expression based on the [great-circle distance](#). However, keep in mind that there's a trade off between performance and accuracy—the more complex your rank expression, the longer it will take to rank your results. For more information, watch the AWS Webcast [Building Location-Based Search With Amazon CloudSearch](#) .

Searching DynamoDB Data with Amazon CloudSearch

Topics

- [Configuring an Amazon CloudSearch Domain to Search DynamoDB Data \(p. 117\)](#)
- [Uploading Data to Amazon CloudSearch from DynamoDB \(p. 119\)](#)
- [Synchronizing a Search Domain with an DynamoDB Table \(p. 121\)](#)

You can specify an DynamoDB table as a source when configuring indexing options or uploading data to a search domain through the console or command line tools. This enables you to quickly set up a search domain to experiment with searching data stored in DynamoDB database tables.

To keep your search domain in sync with changes to the table, you can send updates to both your table and your search domain, or you can periodically load the entire table into a new search domain.

Configuring an Amazon CloudSearch Domain to Search DynamoDB Data

The easiest way to configure a search domain to search DynamoDB data is to use the Amazon CloudSearch console. The console's configuration wizard analyzes your table data and suggests indexing options based on the attributes in the table. You can modify the suggested configuration to control which table attributes are indexed and how they are mapped to index fields.

You can also use the command line tools to generate SDF batches from your table and automatically configure your domain, or you can configure indexing options manually. For general information about configuring indexing options, see [Configuring Index Fields for an Amazon CloudSearch Domain \(p. 58\)](#).

When you automatically configure a search domain from an DynamoDB table, the first 200 unique attributes are mapped to index fields. (You can configure a maximum of 200 fields for a search domain.) When Amazon CloudSearch detects an attribute that has a small number of distinct values, the field is facet-enabled in the suggested configuration.

Important

When you use an DynamoDB table to configure a domain, the data is not automatically uploaded to the domain for indexing. You must upload the data for indexing as a separate step after you configure the domain.

AWS Management Console

You can use the Amazon CloudSearch console to analyze data from an DynamoDB table to configure a search domain. A maximum of 5 MB is read from the table regardless of the table size. By default, Amazon CloudSearch reads from the beginning of the table. You can specify a start key to begin reading from a particular item.

To configure a search domain using an DynamoDB table

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain, and then click the domain's **Indexing Options** link.
3. At the top of the **Indexing Options** pane, click the **configuration wizard** link.

Name	Status	Type	Search	Facet	Result	Default Value	S
	new	text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		[

[Add Index Field](#)

4. In the **Choose Source** step, select **Analyze sample item(s) from DynamoDB**.

Analyze sample item(s) from Amazon DynamoDB

Select the DynamoDB table that contains your sample data. To start reading from a particular item, specify a start key. To limit the read capacity units used while reading from the table, specify the maximum Read Capacity Units %. A maximum of 5 MB of data can be read from the table. If you need to specify a larger data set, use the command line tools. This data is used for configuration only. These documents will not be indexed. Once the domain is created, you can upload documents for indexing.

DynamoDB Table: Select a table...
Read Capacity Units %: 20
Start Hash Key:

5. From the **DynamoDB Table** list, select the DynamoDB table that you want to analyze.
 - To limit the read capacity units that can be consumed while reading from the table, enter the maximum percentage of read capacity units you want to use.
 - To start reading from a particular item, specify a **Start Hash Key**. If the table uses a hash and range type primary key, specify both the hash attribute and the range attribute for the item.
6. When you finish specifying the table options, click **Continue**.
7. In the **Review Configuration** step, review the suggested configuration. You can edit these fields and add additional fields.
8. When you finish, click **Apply Configuration**.

9. In the **Apply Configuration** step, you can choose to run indexing when you exit the configuration wizard. If you haven't uploaded data to your domain yet, clear the **Run Indexing Now** checkbox to exit without indexing. If you are done making configuration changes and are ready to index your data with the new configuration, make sure **Run Indexing Now** is selected. When you are ready to apply the changes, click **Finish**.

You can also use an DynamoDB table to configure indexing options when you first create a domain. In the **Configure Index** step, select **Analyze sample item(s) from DynamoDB** and select the table to analyze.

Command Line Tools

You can use the [cs-generate-sdf](#) (p. 158) and [cs-configure-from-sdf](#) (p. 151) commands to configure a domain based on the data in an DynamoDB table.

To configure a search domain using an DynamoDB table

1. Run the `cs-generate-sdf` command and specify the `--source` and `--output` options. The source is the name of an DynamoDB table. The output is the local directory or Amazon S3 bucket where you want to save the generated SDF batches.

```
cs-generate-sdf --source ddb://myDDBTable --output c:\myddbdata\SDF
```

Note

You can make changes to the SDF data before you use it to configure your domain. For more information about mapping your data to index fields, see [Preparing Your Data for Amazon CloudSearch](#) (p. 50). For information about customizing your domain configuration, see [Configuring Index Fields for an Amazon CloudSearch Domain](#) (p. 58).

2. Run the `cs-configure-from-sdf` command and specify the `--domain` and `--source` options. The domain is the name of the search domain you are configuring. The source specifies the SDF batch (or batches) to use to configure the domain.

```
cs-configure-from-sdf --domain ddb-cs-search --source c:\myddbdata\SDF\*
```

Uploading Data to Amazon CloudSearch from DynamoDB

You can upload DynamoDB data to a search domain through the Amazon CloudSearch console or with the Amazon CloudSearch command line tools. When you upload data from an DynamoDB table, Amazon CloudSearch converts it to SDF batches so it can be indexed. You select which attributes you want to upload from the table by mapping those attributes to index fields in your domain's indexing options. Any attributes that aren't mapped to index fields are ignored. For more information, see [Configuring an Amazon CloudSearch Domain to Search DynamoDB Data](#) (p. 117).

You can upload data from more than one DynamoDB table to the same Amazon CloudSearch domain. If an item with the same key appears in more than one uploaded table, the item with the highest version number overwrites all previous versions.

When converting table data to SDF data, Amazon CloudSearch generates an SDF document for each item it reads from the table, and represents each item attribute as a document field. The unique ID for

each document is either read from the docid item attribute (if it exists) or assigned an alphanumeric value based on the primary key. Similarly, the document version number is either read from the version attribute or generated using a timestamp.

When Amazon CloudSearch generates SDF documents for table items:

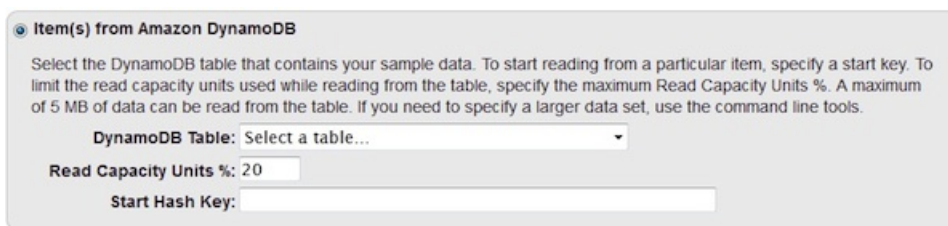
- Sets of strings and sets of numbers are represented as multi-value fields. If an DynamoDB set contains more than 100 values, only the first 100 values are added to the multi-value field.
- DynamoDB binary attributes are ignored.
- Attribute names are modified to conform to the Amazon CloudSearch naming conventions for field names:
 - All uppercase letters are converted to lowercase.
 - If the DynamoDB attribute name does not begin with a letter, the field name is prefixed with `f_`.
 - Any characters other than a-z, 0-9, and `_` (underscore) are replaced by an underscore. If this transformation results in a duplicate field name, a number is appended to make the field name unique. For example, the attribute names `hât`, `h-t`, `hát` would be mapped to `h_t`, `h_t1`, and `h_t2` respectively.
 - If the DynamoDB attribute name exceeds 64 characters, the first 56 characters of the attribute name are concatenated with the 8-character MD5 hash of the full attribute name to form the field name.
 - If the attribute name is `body`, it is mapped to the field name `f_body`.
 - If the attribute name is `text_relevance` it is mapped to the field name `f_text_relevance`.
- Number attributes are mapped to Amazon CloudSearch uint fields and the values are transformed to 32-bit unsigned integers:
 - If a number attribute contains a decimal value, only the integral part of the value is stored. Everything to the right of the decimal point is dropped.
 - If the value is larger than can be stored as an unsigned integer, the value is truncated.
 - Negative integers are treated as unsigned positive integers.

AWS Management Console

You can use the Amazon CloudSearch console to upload up to 5 MB of data from an DynamoDB table to a search domain. To upload a larger amount of data from an DynamoDB table, use the [command line tools](#) (p. 121).

To upload DynamoDB data using the console

1. Sign in to the AWS Management Console and open the [Amazon CloudSearch console](#).
2. In the **Navigation** pane, click the name of the domain.
3. At the top of the **Domain Dashboard**, click **Upload Documents**.
4. In the **Document Source** step, select **Item(s) from DynamoDB**.



Item(s) from Amazon DynamoDB

Select the DynamoDB table that contains your sample data. To start reading from a particular item, specify a start key. To limit the read capacity units used while reading from the table, specify the maximum Read Capacity Units %. A maximum of 5 MB of data can be read from the table. If you need to specify a larger data set, use the command line tools.

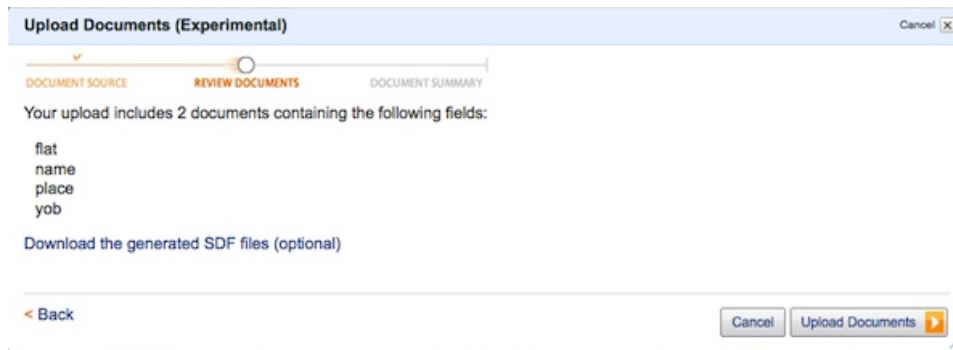
DynamoDB Table:

Read Capacity Units %:

Start Hash Key:

5. In the DynamoDB Table list, select the DynamoDB table that contains your data.

- To limit the read capacity units that can be consumed while reading from the table, enter the maximum percentage of read capacity units.
 - To start reading from a particular item, specify a **Start Hash Key**. If the table uses a hash and range type primary key, specify both the hash attribute and the range attribute for the item.
6. When you finish specifying the table options, click **Continue**.
 7. In the **Review Documents** step, review the items that will be uploaded. (You can also save the generated SDF batches by clicking **Download the generated SDF files**.) When you finish, click **Upload Documents**.



8. In the **Document Summary** step, click **Finish** to exit the upload documents wizard.

Command Line Tools

You can use the [cs-generate-sdf](#) (p. 158) command to generate and upload SDF batches from items in an DynamoDB table. (To upload existing SDF batches, use [cs-post-sdf](#) (p. 157).)

To upload DynamoDB data using the command line tools

- Run the `cs-generate-sdf` command and specify the `--source` and `--domain` options. The source is the name of the DynamoDB table that contains your data. The domain is the name of the search domain you want to use to search the data.

```
cs-generate-sdf --domain ddb-cs-search --source ddb://myDDBTable
```

Note

You can save the generated SDF batches to your local file system or an Amazon S3 bucket by specifying the `--output` option instead of the `--domain` option. This enables you to review and modify the SDF data before uploading it with the [cs-post-sdf](#) (p. 157) command.

Synchronizing a Search Domain with an DynamoDB Table

To keep your search domain in sync with updates to your DynamoDB table, you can either programmatically track and apply updates to your domain, or periodically create a new domain and upload the entire table again. If you have a large amount of data, it's best to track and apply updates programmatically.

Programmatically Synchronizing Updates

To synchronize changes and additions to your DynamoDB table, you can create a separate update table to track the changes to the table you are searching and periodically upload the contents of the update table to the corresponding search domain.

To remove documents from the search domain, you must generate and upload SDF batches that contain a delete operation for each deleted document. One option is to use a separate DynamoDB table to track deleted items, periodically process the table to generate a batch of delete operations, and upload the batch to your search domain.

To make sure that you don't lose any changes that are made during the initial data upload, you must begin collecting tracking changes before the initial data upload. While you might update some Amazon CloudSearch documents with identical data, you ensure that no changes are lost and your search domain contains an up-to-date version of every document.

Important

For a document to be updated or deleted, the specified version number must be greater than the last version number Amazon CloudSearch received. If you do not explicitly set the version attribute when uploading data from an DynamoDB table, Amazon CloudSearch uses a timestamp to generate the version. (For more information about versioning, see [Document Versions in Amazon CloudSearch](#) (p. 53).)

How often you synchronize updates depends on the volume of changes and your update latency tolerance. One approach is to accumulate changes over a fixed time period and at the end of the time period upload the changes and delete the period's tracking tables.

For example, to synchronize changes and additions once a day, at the beginning of each day you could create a table called `updates_YYYY_MM_DD` to collect the daily updates. At the end of the day, you upload the `updates_YYYY_MM_DD` table to your search domain. (If the update table is larger than 5 MB, you must use the command line tools.) After the upload is complete, you can delete the update table and create a new one for the next day.

Switching to a New Search Domain

If you don't want to track and apply individual updates to your table, you can periodically load the entire table into a new search domain and then switch your query traffic over to the new domain.

To switch to a new search domain

1. Create a new search domain and clone the configuration from your existing domain. For more information, see [Cloning an Existing Domain's Indexing Options](#) (p. 64).
2. Upload the entire DynamoDB table to the new domain. (If the table is larger than 5 MB, you must use the command line tools to upload it.) For more information, see [Uploading Data to Amazon CloudSearch from DynamoDB](#) (p. 119).
3. After the new domain is active, update the DNS entry that directs query traffic to the old search domain to point to the new domain. For example, if you use [Amazon Route 53](#), you can simply update the recordset with your new search service endpoint.
4. Delete the old domain.

Using the Amazon CloudSearch Console

Topics

- [Amazon CloudSearch Dashboard](#) (p. 123)
- [Domain Dashboard](#) (p. 124)
- [Search Domains](#) (p. 125)
- [Processing Status](#) (p. 125)
- [Needs Indexing Status](#) (p. 125)
- [Run a Test Search](#) (p. 126)
- [Rank Comparison](#) (p. 126)
- [Upload Data](#) (p. 128)
- [Search Count Metrics](#) (p. 130)
- [Top Documents Metrics](#) (p. 130)
- [Top Searches Metrics](#) (p. 130)
- [Access Policies](#) (p. 131)
- [Indexing Options](#) (p. 131)
- [Rank Expressions](#) (p. 132)
- [Stems](#) (p. 133)
- [Stopwords](#) (p. 134)
- [Synonyms](#) (p. 135)

This section provides information to help you navigate and interact with the Amazon CloudSearch console to create and manage search domains.

Amazon CloudSearch Dashboard

The Amazon CloudSearch Dashboard provides a summary of your account and displays overall information about the Amazon CloudSearch service. From the Amazon CloudSearch Dashboard, you can:

- View the status of the Amazon CloudSearch service
- View the status of your search domains

- Access the dashboard for a particular domain
- Access the Amazon CloudSearch documentation and other resources

The Amazon CloudSearch Dashboard has three sections: *My Search Domains*, *Service Health*, and *Related Links*.

My Search Domains

My Search Domains displays a summary of each [search domain](#) associated with your account.

- To update the table with the latest information, click the **Refresh** button at the top of the page.
- To view the [domain dashboard \(p. 124\)](#) for a particular domain, click the name of the domain.

The following information is displayed in the My Search Domains table:

- **Domain**—The name of each search domain.
- **Status**—The status of each search domain:
 - **LOADING**—The domain has just been created and is still being initialized. You must wait until the domain status changes to PROCESSING, NEEDS INDEXING, or ACTIVE before you can start uploading documents.
 - **ACTIVE**—The domain is running and all configured fields have been indexed.
 - **NEEDS INDEXING**—You have made changes to the domain configuration that require rebuilding the index. If you search the domain, these changes won't be reflected in the results. When you are done making changes, click **Run Indexing** to rebuild your index.
 - **PROCESSING**—Configuration changes are being applied to your domain. If you search the domain, the most recent configuration changes might not be reflected in the results.
 - **BEING DELETED**—You chose to delete the domain and its contents the domain and all of its resources are in the process of being removed. When deletion is complete, the domain will be removed from this list.
- **Searchable Documents**—The number of documents indexed for this domain.
- **Search Instances**—The number and type of search instances serving this domain.

Service Health

This section provides information about the health of the Amazon CloudSearch service.

Related Links

This section provides links to more information about the Amazon CloudSearch service.

Domain Dashboard

The Domain Dashboard provides information about a single [search domain \(p. 125\)](#). From the Domain Dashboard, you can:

- View the status of a particular search domain
- Upload documents to the domain
- Search the domain
- Access the domain configuration pages

- Delete the domain

Domain Status

Domain Status displays a summary of the domain's configuration and status. A domain can be in one of the following states:

- **LOADING**—The domain has just been created and is still being initialized. You must wait until the domain status changes to **PROCESSING**, **NEEDS INDEXING**, or **ACTIVE** before you can start uploading documents.
- **ACTIVE**—The domain is running and all configured fields have been indexed.
- **NEEDS INDEXING**—You have made changes to the domain configuration that require rebuilding the index. If you search the domain, these changes won't be reflected in the results. When you are done making changes to the domain configuration, click **Run Indexing** to rebuild your index.
- **PROCESSING**—Configuration changes are being applied to your domain. If you search the domain, the most recent configuration changes might not be reflected in the results.
- **BEING DELETED**—You chose to delete the domain and the domain and all of its resources are in the process of being removed. When deletion is complete, the domain will be removed from your list of domains.

Search Domains

A search domain provides the engine that enables you to index and search your data. You configure the search domain with the data you want to search, descriptions of your data, and options that control how it is indexed.

You create a search domain for each different collection of data you want to make searchable. Each domain has a unique endpoint you use to submit search requests.

To create a search domain, click **Create a New Domain** in the Navigation pane.

Processing Status

The domain status *Processing* indicates that the domain's index is being rebuilt. It can take some time to rebuild the entire index—how long depends on the number and complexity of the documents being indexed.

Your domain's search service is accessible during indexing, however the most recent configuration changes will not be reflected in search results until the process is complete. To see which fields are still being processed, go to the domain's Indexing Options page.

Note that document updates are processed continuously—the index is updated to reflect additions and deletions in near real time. The index is normally only completely rebuilt when you explicitly invoke index documents.

Needs Indexing Status

The domain status *Needs Indexing* indicates that the domain index needs to be rebuilt. When you make configuration changes to your domain, such as adding index fields, your domain's index needs to be rebuilt before those changes will be reflected in search results.

Building your domain's full index takes some time, so you should complete all of your configuration changes before running index documents.

To start indexing, click **Run Indexing** at the top of the domain dashboard. The Amazon CloudSearch dashboard also shows any domains that need to be reindexed.

Run a Test Search

Run a Test Search enables you to submit a search request to the specified domain's search endpoint. The domain must have at least one indexed document to be able to respond to search requests.

To search the domain

1. To constrain the search to a particular text or literal field, select the field you want to search. By default, **All Text Fields** will be searched.
2. Enter the terms you want to search for in the **Search** field. By default, a document must contain all of the specified search terms to be returned in the results. To exclude documents that contain a particular term, prefix the term with the - (NOT) operator. For example, `wars -star` retrieves all documents that contain the term `wars`, but do not contain the term `star`. To find documents that contain either of two terms, use the | (OR) operator to separate the terms. For example, `star | wars` retrieves documents that contain either the term `star` or the term `wars`.
3. Click **GO**.

The search results are automatically sorted by text relevance. To change the sort order, choose the index field or rank expression you want to sort by from the **Sort by** menu. You can use any uint field to sort numerically, and any result-enabled text or literal field to sort alphabetically.

Facets are automatically calculated for all facet-enabled literal fields and used to display filtered results. To filter the search results, select one or more facet values in the **Filter Search Results** column. To clear a filter, click the fieldname.

To view the actual search request and the raw search results returned by the search service, select the result format you want to view: *JSON* or *XML*.

Rank Comparison

The Rank Comparison page enables you to compare how search results are ranked using two different rank expressions. You can change the rank expressions and adjust field weights for text fields to see how those changes affect the search results. When you are satisfied with the results, you can save the rank expression to your domain configuration.

Comparing Rank Expressions

Comparing rank expressions enables you to see how changes to the expression and field weights affect how search results are sorted.

To compare rank expressions

1. In each Rank Expression editor, enter a new rank expression or select an existing expression from the Expressions menu. New rank expressions are validated when you submit a search request. If errors are detected, the rank expression is highlighted in red and a description of the problem is displayed.

Amazon CloudSearch Developer Guide Comparing Rank Expressions





- Specify the field weights to use for each rank expression by adjusting the sliders in the **Field Weights** menu. You can also edit the field weights directly in the rank expression. Text field weights must be in the range 0.0 to 10.0, inclusive. By default, the weight for all fields is set to 1.0. Setting individual field weights enables you to control how much matches in particular text fields affect a document's `text_relevance` score. You can also change the default weight.

Note

Adjusting field weights only affects result ranking if the rank expression uses the `cs.text_relevance` function. When you adjust the weights, the `cs.text_relevance` function is automatically inserted in the rank expression. You can modify the expression to change how the customized `text_relevance` score contributes to a document's overall ranking. For more information, see [Using Relative Field Weighting to Customize Text Relevance in Amazon CloudSearch](#) (p. 110) in the Amazon CloudSearch Developer Guide.

- Enter the terms you want to search for in the **Search** field and click **GO**. The results for the search are ranked using the specified rank expressions and weights. The results are refreshed whenever you make changes to the rank expressions or weights.

The search results for the two rank expressions are shown side-by-side. (If the rank expression is empty, the results are sorted according to the default text_relevance score.) Four icons are used to highlight the differences:

-  Green Up Arrow—The document is ranked higher in the search results using the second rank expression.
-  Red Down Arrow—The document is ranked lower in the search results using the second rank expression.
-  Yellow Plus—The document is included in the search results using the second rank expression, but was omitted from the search results using the first rank expression.
-  Red Minus—The document was omitted from the search results using the second rank expression, but was included in the search results using the first rank expression.

Saving Rank Expressions

When you are finished tuning a rank expression, you can save it in your domain configuration. You can use the rank parameter in your search requests to reference saved rank expressions to control how the results are sorted.

To save a rank expression

1. Select **Save Expression** from the Expressions menu.
2. Enter a name for the expression and click **OK**.

Upload Data

When you upload data to your search domain, Amazon CloudSearch generates a search index according to the index fields and text options configured for the domain. For your data to be indexed, it must be described according to the Search Data Format (SDF). For more information, see [Uploading Data to an Amazon CloudSearch Domain \(p. 77\)](#).

You can generate and upload SDF batches yourself, or Amazon CloudSearch can automatically generate SDF batches from source data stored locally, in Amazon S3, or in DynamoDB. When you upload local or S3 data, SDF can be generated from:

- Comma Separated Value files (.csv)
- Adobe Portable Document Format files (.pdf)
- HTML files (.htm, .html)
- Microsoft Excel files (.xls, .xlsx)
- Microsoft PowerPoint files (.ppt, .pptx)
- Microsoft Word files (.doc, .docx)
- Text files (.txt)
- JSON files (.json)
- XML files (.xml)

To use the console to upload up to 5 MB of data from your local filesystem, Amazon S3, or DynamoDB

1. In the **Interactive Upload** section, click **Start Upload**.
2. In the **Upload** wizard select the type of data you want to upload and specify the local files, Amazon S3 objects, DynamoDB table items, or predefined data you want to upload to your search domain.

Important

Uploading data from an DynamoDB table consumes provisioned throughput from the DynamoDB table.

To use `cs-generate-sdf` to generate and upload SDF batches

1. Install the Amazon CloudSearch command line tools. For instructions, see [Installing the Command Line Tools for Amazon CloudSearch \(p. 137\)](#).
2. Run the `cs-generate-sdf` (p. 158) command and use the `--source` option to specify the local files, Amazon S3 objects, or DynamoDB table items you want to upload to your search domain. Use the `--output` option to save the generated SDF instead of uploading it to your domain. For more information, see [Generating SDF from Your Source Data in Amazon CloudSearch \(Experimental\) \(p. 55\)](#)

Important

Uploading data from an DynamoDB table consumes provisioned throughput from the DynamoDB table.

To create a data pipeline to upload data from DynamoDB using an Amazon EMR cluster

1. In the **Bulk Upload** section, click **Create Upload Pipeline**.
2. In the **Data Pipeline** console, create a new pipeline that uses the **Export DynamoDB to CloudSearch** template.

Important

Uploading data from an DynamoDB table consumes provisioned throughput from the DynamoDB table and configuring a data pipeline to upload DynamoDB data will result in additional AWS charges.

To use `cs-post-sdf` to upload SDF batches to your search domain

1. Install the Amazon CloudSearch command line tools. For instructions, see [Installing the Command Line Tools for Amazon CloudSearch \(p. 137\)](#)
2. Run the `cs-post-sdf` (p. 157) command and use the `--source` option to specify the SDF batches you want to upload to your search domain.

To use the document service API to upload SDF batches to your search domain

- Post SDF data to your search domain's Document Service endpoint using the [documents/batch \(p. 217\)](#) resource. For example:

```
curl -X POST --upload-file sdfdata.sdf
doc.movies-123456789012.us-east-1.cloudsearch.amazonaws.com/2011-02-01/documents/batch
--header "Content-Type:application/json"
```


Search Count Metrics

The Search Count metrics page shows the number of searches over time for a particular domain. Two metrics are displayed:

- **Total Searches**—The total number of searches over the specified time period.
- **Searches with No Results**—The number of searches over the specified time period for which no matching documents were found.

By default, this page displays metrics for the previous month. Dates are based on GMT. Search count metrics are retained indefinitely.

To view metrics for a different time period, hover over the date range to display the date menu and enter the new dates in the **From** and **To** fields. You can also choose dates from the calendar.

To change the granularity of the metrics, select **Hour**, **Day**, or **Month**.

To view details for a particular day, week, or month, hover over the data point on the graph.

To download the search count data for offline analysis, click **Download CSV**.

Top Documents Metrics

The Top Documents metrics page shows the documents that were most frequently returned in search results on a particular day.

By default, this page displays metrics for the current date. Dates are based on GMT. Top Documents metrics are updated hourly and retained for 13 months.

To view metrics for a different day, hover over the date to display the date menu and select or enter the new date.

To download the top documents data for offline analysis, click **Download CSV**.

Top Searches Metrics

The Top Searches metrics page shows the most frequent searches on a particular day. Two metrics are displayed:

- **Top Searches**—The most frequent searches submitted on the specified day.
- **Top Searches without Results**—The most frequent searches on the specified day for which no matching documents were found.

By default, this page displays metrics for the current date. Dates are based on GMT. Top Searches metrics are updated hourly and retained for 13 months.

To view metrics for a different day, hover over the date to display the date menu and select or enter the new date.

To download the top searches data for offline analysis, click **Download CSV**.

Access Policies

Access policy rules control which IP addresses can access a domain's search and document services. Deny rules take precedence over allow rules.

IP address authorization is only used to control access to a domain's search and document endpoints. The Amazon CloudSearch configuration service uses standard AWS authentication.

The access policy rules defined here do not affect access to the Amazon CloudSearch services through the console. The console uses your AWS account name and password for authentication.

To add an access policy rule

1. Click **Add a New Rule**.
2. Select the type of rule you want to add: *Allow* or *Deny*.
3. Specify whether you want the rule to apply to all of the domain's services, or only control access to the domain's search or document endpoint: *All Services*, *Search*, *Document*.
4. Enter an IP address. This can either be an individual address that you want to authorize or block, or the base address for an address range specified in CIDR notation.
5. If you are specifying an address range, enter the network mask. This value indicates the number of leftmost bits used to identify the network. For example, if you specify the base address 10.24.34.0 and a network mask of 24, the range of IP addresses affected by the rule would be 10.24.34.0 - 10.24.34.255.
6. To specify additional addresses or ranges as part of the rule, click the **+** button.
7. Click **Submit** to save your changes.

You can also use the following shortcuts to configure access policies:

- Click **Recommended rules** to allow search requests from any IP address, but only allow document updates from your IP address.
- Click **Allow only my IP address access to all services** to only allow search and document requests from your IP address.
- Click **Allow everyone access to all services** to allow search and document requests from any IP address. Keep in mind that this means *anyone* can upload documents to your domain.
- Click **Deny everyone access to all services** to block search and document requests from all IP addresses. You will only be able to interact with your domain through the console.

You must click **Submit** to save any changes you make to the access policy rules. To restore the previously configured rules, click **Revert**.

Indexing Options

Indexing Options define the fields included in a domain's index and control how they can be used. You can manually configure each of the fields for your domain, or upload sample data so Amazon CloudSearch can configure fields automatically.

Amazon CloudSearch supports three types of index fields:

- **text**—a text field contains arbitrary alphanumeric data. A text field is always searchable. The value of a text field can be returned in search results or the field can be used as a facet, but not both.

- **literal**—a literal field contains an identifier or other data that you want to be able to match exactly. By default, literal fields are searchable. The value of a literal field can be returned in search results or the field can be used as a facet, but not both.
- **uint**—a uint field contains an unsigned integer value. Faceting is always enabled for uint fields and the value of a uint field can always be returned in results. Uint fields can also be used in rank expressions.

In the definition of a field you can specify:

- Whether the field can be searched
- Whether facets can be calculated for the field to enable filtering
- Whether the contents of the field can be returned in the search results
- A default value for the field
- One or more data sources for the field

To configure a new field

1. Click **Add Index Field**.
2. Enter a name for the field. Field names must begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). The names *body*, *docid*, and *text_relevance* are reserved and cannot be used as field names.
3. Select the type of the new field: *text*, *literal*, or *uint*.
4. Choose the search, facet, and result options you want to enable for the field. Text fields are always searchable. Text and literal fields can be used as facets or returned in search results, but not both. Faceting is always enabled for uint fields and they can always be returned in results.
5. Specify a default value for the field (optional). This value is used when no value is specified for the field in the domain's document data.
6. Add one or more sources for the field (optional). You can add sources to combine the contents of multiple fields into a single field, strip common title words from a field so you can use it for sorting, or map values from the source to a different set of values in your new index field.
7. To configure additional fields, click **Add Index Field** and repeat.
8. When you are done configuring fields, click **Submit** to save your changes. To restore the previous field configurations, click **Revert**.

Rank Expressions

You can define numeric expressions and reference them with the rank parameter in your search requests to control how the results are sorted. Rank expressions can also be used to set thresholds for search results. The rank expression syntax is based on JavaScript expressions.

To create a rank expression

1. Click **Add a New Expression**.
2. Enter a name for the expression. Rank expression names must begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). The names *body*, *docid*, and *text_relevance* are reserved and cannot be used as expression names.
3. Enter the numeric expression you want to evaluate to determine how to sort or constrain search results. A rank expression can reference uint fields and other rank expressions by name. You can also use the default *text_relevance* score in your rank expressions.
4. To configure additional rank expressions, click **Add a New Expression** and repeat.
5. When you are done configuring rank expressions, click **Submit** to save your changes. To restore the previous rank expressions, click **Revert**.

Rank expressions support:

- Integer, floating point, hex and octal literals
- Shortcut evaluation of logical operators such that an expression `a || b` evaluates to the value `a` if `a` is `true` without evaluating `b` at all
- JavaScript order of precedence for operators
- Arithmetic operators: `+` `-` `*` `/` `%`
- Bitwise operators: `|` `&` `^` `~` `<<` `>>` `>>>`
- Boolean operators (including the ternary operator): `&&` `||` `!` `?:`
- Comparison operators: `<` `<=` `=` `>=` `>`
- Common mathematic functions: `abs` `ceil` `erf` `exp` `floor` `lgamma` `ln` `log2` `log10` `max` `min` `sqrt` `pow`
- Trigonometric library functions: `acosh` `acos` `asinh` `asin` `atanh` `atan` `cosh` `cos` `sinh` `sin` `tanh` `tan`
- Random generation of a number between 0 and 1: `rand`
- Current time in epoch: `time`
- `min` `max` functions that operate on a variable argument list

When computing the value of a rank expression, intermediate results are double-precision floating point values. A rank expression's return value is automatically converted from floating point to a 32-bit unsigned integer by rounding to the nearest integer. This conversion imposes a natural floor (0) and ceiling (4294967295). Mathematical errors such as dividing by 0 will fail the computation and return a value of 0.

Example 1: `((0.3 * popularity) / 10.0) + ((0.7 * text_relevance))`

This expression defines a custom relevance score where 30% of a document's score is based on a uint field named *popularity*, and 70% of its score is based on its default `text_relevance` score. This expression returns a value in the range 0-1000. The *popularity* field is assumed to have values in the range 0-10000. The default `text_relevance` score is in the range 0-1000. If this expression is called *combined_relevance*, you could use it by adding the query parameter `rank=combined_relevance` to your search request.

Example 2: `(time() > available_on)?1:0`

This expression defines a threshold constraint that could be used to rule out documents whose `available_on` field is later than the current time, where `available_on` is an epoch uint value defined for each document. If this expression is called *is_available*, you could use it by adding the query parameter `t-is_available=1` to your search request.

Stems

Stems map variations of a term to a common stem to enable matching on all the variants. You can define multiple terms that map to the same stem. For example:

```
ran, run
```

```
running, run
```

You can add term and stem pairs to the list, edit the list directly, or copy and paste the list into a text editor to make changes. Each line in the stems list should contain a comma-separated *term*, *stem* pair.

To add stems

1. Enter the term that you want to map to a stem in the **Term** field.
2. Enter the stem you want to map the term to in the **Stem** field.
3. Click **Add Stem**. You can also click at the beginning or end of an entry in the stem list, press the Enter key, and type a comma-separated *term, stem* pair.
4. When you're done adding stems, click **Submit** to save your changes. To revert to the previous list, click **Revert**.

To delete stems

1. Select the *term, stem* pair(s) you want to remove and press the Delete key.
2. When you're done deleting stems, click **Submit** to save your changes. To revert to the previous list, click **Revert**.

To replace the entire list of stems

1. Copy the list of stems you want to define. The list must contain one *term, stem* pair per line.
2. Click **select** to select the existing stem list.
3. Use Paste to replace the existing list.
4. Click **Submit** to save your changes. To revert to the previous list, click **Revert**.

Stopwords

You can configure stopwords to specify words that frequently occur in your data but should be ignored to avoid returning massive numbers of matches. The default stopwords for English are: *a, an, and, are, as, at, be, but, by, for, in, is, it, of, on, or, the, to, was*.

You can add individual stopwords to the list, edit the list directly, or copy and paste the list into a text editor to make changes. Each line in the stopwords list should contain a single term.

To add stopwords

1. Enter a term in the **New Stopword** field and click **Add Stopword**. You can also click at the beginning or end of a term in the stopwords list, press the Enter key, and type the new term.
2. When you're done adding stopwords, click **Submit** to save your changes. To revert to the previous list, click **Revert**.

To delete stopwords

1. Select the term(s) you want to remove and press the Delete key.
2. When you're done deleting stopwords, click **Submit** to save your changes. To revert to the previous list, click **Revert**.

To replace the entire stopwords list

1. Copy the list of stopwords you want to use. The list must contain one term per line.
2. Click **select** to select the existing stopwords list.
3. Use Paste to replace the existing list.
4. Click **Submit** to save your changes. To revert to the previous list, click **Revert**.

Synonyms

You can configure synonyms for terms that appear in your document data. When you configure a synonym, if a user searches for the synonym rather than the indexed term, the results will include documents that contain the indexed term. For example, you might want to configure synonyms so that a search for "Rocky Four" or "Rocky 4" will match the movie titled "Rocky IV". To do that, you would configure *4* and *four* as synonyms of the indexed term *IV*.

A term can have multiple synonyms. Each entry in the synonym list specifies a term followed by a comma-separated list of its synonyms. For example, the following entry defines synonyms for the term *horse*:

```
horse, colt, filly, foal
```

You can add term and synonym mappings to the list, edit the list directly, or copy and paste the list into a text editor to make changes.

To add synonyms

1. Enter a term in the **New Term** field. The term is a word that appears in your data.
2. Specify synonyms for the term as a comma-separated list in the **Synonyms** field. Each synonym is a word that doesn't necessarily appear in your data, but should match documents that contain the specified term.
3. Click **Add Synonyms**. You can also click at the beginning or end of an entry in the synonym list, press the Enter key, and type a term followed by a comma-separated list of synonyms.
4. When you're done adding synonyms, click **Submit** to save your changes. To revert to the previous list, click **Revert**.

To delete synonyms

1. Select the *term, synonym, ...* mapping you want to remove and press the Delete key.
2. When you're done deleting synonyms, click **Submit** to save your changes. To revert to the previous list, click **Revert**.

To replace the entire synonym list

1. Copy the list of synonyms you want to define. The list must contain one *term, synonym[, synonym...]* mapping per line.
2. Click **select** to select the existing synonym list.
3. Use Paste to replace the existing list.
4. Click **Submit** to save your changes. To revert to the previous list, click **Revert**.

Amazon CloudSearch Command Line Tool Reference

Topics

- [Using the Command Line Tools for Amazon CloudSearch \(p. 136\)](#)
- [Common Options for the Amazon CloudSearch Command Line Tools \(p. 139\)](#)
- [cs-configure-access-policies \(p. 140\)](#)
- [cs-configure-fields \(p. 143\)](#)
- [cs-configure-ranking \(p. 146\)](#)
- [cs-configure-text-options \(p. 148\)](#)
- [cs-create-domain \(p. 150\)](#)
- [cs-configure-from-sdf \(p. 151\)](#)
- [cs-delete-domain \(p. 153\)](#)
- [cs-describe-domain \(p. 154\)](#)
- [cs-index-documents \(p. 155\)](#)
- [cs-post-sdf \(p. 157\)](#)
- [Experimental Tools for Amazon CloudSearch \(p. 158\)](#)

This section provides detailed information about the Amazon CloudSearch command line tools. You can also access the reference information for each tool from the command line by specifying the `--help` option. For example, `cs-generate-sdf --help`.

The Amazon CloudSearch command line tools wrap the configuration and document service APIs to provide a simple way to set up and manage your search domains. Both the command line tools and sample IMDB movie data set are available from the [Amazon CloudSearch developer tools page](#).

Using the Command Line Tools for Amazon CloudSearch

This section describes:

- [Prerequisites for Installing the Amazon CloudSearch Command Line Tools \(p. 137\)](#)

- [Installing the Command Line Tools for Amazon CloudSearch \(p. 137\)](#)
- [Running the Amazon CloudSearch Commands \(p. 139\)](#)

Prerequisites for Installing the Amazon CloudSearch Command Line Tools

To use the Amazon CloudSearch command line tools, you need:

- A basic familiarity with working in a Linux/UNIX or Windows environment.
- A Java 6-compatible Java Runtime Environment (JRE).
- A `JAVA_HOME` environment variable that points to your Java runtime. This environment variable should be set to the full path of the directory that contains the `bin` directory that contains the `java` (Linux/UNIX) or `java.exe` (Windows) executable.
- Your AWS access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about getting credentials, see [How Do I Get Security Credentials?](#) in the *AWS General Reference*.

Installing the Command Line Tools for Amazon CloudSearch

To install the Amazon CloudSearch command line tools

1. To download the command line tools for Windows, go to <https://aws.amazon.com/developertools/4320728073503020> and click the **Download** button.
2. To download the command line tools for Mac OS/Linux, go to <https://aws.amazon.com/developertools/9054800585729911> and click the **Download** button.
3. Unpack the `.zip` or `.tar.gz` file. On Windows, we recommend unzipping the tools in the `C:\CloudSearch` directory.
4. Set the `CS_HOME` environment variable to point to the directory where you unpacked the tools.

On Linux and UNIX, enter following command:

```
export CS_HOME=install_directory_path
```

On Windows, enter the following command:

```
set CS_HOME=install_directory_path
```

Note

These examples temporarily set the `CS_HOME` and `PATH` variables for the duration of your terminal session. You can also set them permanently. On Linux and MacOSX, add the `export` commands to your shell startup file (`.profile`, `.bashrc`, `.tcshrc`, or `.zshrc`) in your home directory. On Windows, you can do this through the Control Panel: Control Panel > System and Security > System > Advanced > Environment Variables.

5. Add the `CS_HOME` environment variable to your `PATH`.

On Linux and UNIX, enter following command:

```
export PATH=$PATH:$CS_HOME/bin
```

On Windows, enter the following command:

```
set PATH=%PATH%;%CS_HOME%\bin
```

6. Make sure you have the Java 6 (or later) JRE installed and the `JAVA_HOME` environment variable is set to the full path of the directory that contains the bin directory in which the Java executable resides. For information about checking your Java installation, go to java.com.

Note

On Mac OS X, `JAVA_HOME` should be set using the `/usr/libexec/java_home` command. For example: `export JAVA_HOME=$(/usr/libexec/java_home)`. For more information, see [QA1170](#) on developer.apple.com.

7. Configure the command line tools to use your AWS identifiers. The Amazon CloudSearch command line tools look for your AWS identifiers in a text file on your local system in the location specified by the `AWS_CREDENTIAL_FILE` environment variable. If you have not already configured an AWS credential file:

- a. Use a text editor to create a two-line text file that specifies your AWS identifiers. The first line sets the `accessKey` property and the second line sets the `secretKey` property. For example:

```
accessKey=AKIAIOSFODNN7EXAMPLE  
secretKey=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
```

- b. Save the file using any name you want (for example, `account-key`).
- c. Limit the file permissions to only the file owner. (For example, use `chmod 600` on the file if you are using Linux/UNIX).
- d. Set the `AWS_CREDENTIAL_FILE` environment variable.

On Linux and UNIX, enter following command:

```
export AWS_CREDENTIAL_FILE=credential_file_path
```

On Windows, enter the following command:

```
set AWS_CREDENTIAL_FILE=credential_file_path
```

8. Set the `CS_ENDPOINT` environment variable to specify the Amazon CloudSearch configuration service endpoint for the AWS Region where you want to create and configure search domains. See [Amazon CloudSearch Regions and Endpoints](#) for a list of supported regions.

On Linux and UNIX, enter following command:

```
export CS_ENDPOINT=cloudsearch.region.amazonaws.com
```

On Windows, enter the following command:

```
set CS_ENDPOINT=cloudsearch.region.amazonaws.com
```


Note

If `CS_ENDPOINT` is not set, the Amazon CloudSearch command line tools default to the configuration service endpoint in the US East (Northern Virginia) Region, `cloudsearch.us-east-1.amazonaws.com`. You can also explicitly set the endpoint by specifying the `--endpoint` option when you run Amazon CloudSearch commands.

9. To verify that the Amazon CloudSearch tools are configured correctly, run the `cs-describe-domain` command. (Since you haven't configured any domains yet, the Domain Summary will be empty.)

```
cs-describe-domain
```

If you get an error, check the following:

- If the system cannot find the specified path, your `JAVA_HOME` environment variable needs to be set to the location where you have the JRE installed. For example, `C:\Program Files\Java\jre6`.
- If `cs-describe-domain` is not recognized as a command, check your `PATH` and make sure it contains the `bin` directory for the command line tools, for example `/Users/username/CloudSearch/tools/bin`.
- If you get an `InvalidClientId` error, your AWS credentials are not configured correctly. Make sure that you've configured the `AWS_CREDENTIAL_FILE` environment variable and that your credential file contains a valid AWS access key ID and secret access key.

Running the Amazon CloudSearch Commands

All of the Amazon CloudSearch commands require an AWS access key ID and secret access key. The easiest way to specify your keys is to set up an AWS credential file and set the `AWS_CREDENTIAL_FILE` environment variable as described in the installation instructions.

You can also explicitly specify your keys with each request, either by using the `--aws-credential-file` option to specify the location of your credential file, or by specifying both the `--access-key` and `--secret-key` options.

For most commands, you must also specify the name of your search domain with the `-d` or `--domain-name` option. (The one exception is that you can invoke the `cs-describe-domain` command without specifying the `--domain-name` option to list information about all of your Amazon CloudSearch domains.)

Common Options for the Amazon CloudSearch Command Line Tools

This section lists the options that can be specified for all of the Amazon CloudSearch command line tools.

Option	Description
<code>-a</code> , <code>--access-key</code> <i>STRING</i>	Your AWS access key. Used in conjunction with <code>--secret-key</code> . Required if you do not use an AWS credential file. Example: <code>-a AKIAIOSFODNN7EXAMPLE</code>

Option	Description
<code>-c, --aws-credential-file FILE</code>	The path to the file that contains your AWS access key and secret access key. Required if you have not set the <code>AWS_CREDENTIAL_FILE</code> environment variable or explicitly set your credentials with <code>--access-key</code> and <code>--secret-key</code> . Example: <code>-c keyfile.txt</code>
<code>-d, --domain-name STRING</code>	The name of the domain that you are querying or configuring. Required for most commands, optional for <code>cs-describe-domain</code> . Example: <code>-d imdb-movies</code>
<code>-e, --endpoint URL</code>	The endpoint for the Amazon CloudSearch Configuration Service. Optional. Default: The <code>CS_ENDPOINT</code> environment variable or <code>cloudsearch.us-east-1.amazonaws.com</code> if the environment variable is not set. Example: <code>-e cloudsearch.eu-west-1.amazonaws.com</code>
<code>-h, --help</code>	Displays usage information for the command. Optional.
<code>-k, --secret-key STRING</code>	Your AWS secret key. Used in conjunction with <code>--access-key</code> . Required if you do not use an AWS credential file. Example: <code>-k wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY</code>
<code>-ve, --verbose</code>	Display verbose output, including the API request and response. Optional.
<code>-v, --version</code>	Display the version number of the command line tools. Optional.

cs-configure-access-policies

NAME
<code>cs-configure-access-policies</code> - Configure access to an Amazon CloudSearch domain.
SYNOPSIS
<code>cs-configure-access-policies --service doc search all</code> <code> [--allow IP CIDR all] [--deny IP CIDR all]</code> <code> [--update] [--policy-file FILE]</code> <code> [--delete IP CIDR] [--force]</code> <code> [--retrieve]</code> COMMON_OPTIONS
DESCRIPTION
Defines access policies for a domain's document and search endpoints.

When a domain is first created, it is configured to deny all access. To access the document or search services through the Amazon CloudSearch Command Line Tools or APIs, you must authorize one or more IP addresses.

This command provides two ways for you to update your domain's access policies:

- `--update` Add or remove specific permissions from your domain's access policies. Changes are automatically merged with the domain's existing policy document.
- `--policy-file` Upload a policy document to your domain. The uploaded file overwrites the domain's existing policy document.

When using the `--update` option, you can specify multiple `--allow` or `--deny` options to allow or block multiple IP addresses or address ranges. You must specify one or more `--service` options to indicate which service endpoints you want to apply the access policies to.

Address ranges are specified using Classless Inter-Domain Routing (CIDR) notation with the base IP address followed by a / and a network mask that indicates the number of leftmost bits used to identify the network. If you don't specify a network mask it defaults to 32, which authorizes or blocks only the specified IP address.

When using the `--policy-file` option, the uploaded policy document replaces the domain's existing policy document. The specified file must be a valid AWS Identity and Access Management (IAM) policy document. (You can use the `--retrieve` mode to get the domain's current policy document.) For information about the IAM Access Policy Language, see <http://docs.aws.amazon.com/IAM/latest/UserGuide/AccessPolicyLanguage.html>.

COMMON OPTIONS

- `-a, --access-key STRING` Your AWS access key ID. Used in conjunction with `--secret-key`. Required if you do not use an AWS credential file.
- `-c, --aws-credential-file FILE` The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the `AWS_CREDENTIAL_FILE` environment variable or explicitly set your credentials with `--access-key` and `--secret-key`.
- `-d, --domain-name STRING` The name of the domain that you are querying or configuring. Required.
- `-e, --endpoint URL` The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the `CS_ENDPOINT` environment variable or `cloudsearch.us-east-1.amazonaws.com` if the environment variable is not set. Optional.
- `-h, --help` Display this help message. Optional.

<code>-k, --secret-key STRING</code>	Your AWS secret access key. Used in conjunction with <code>--access-key</code> . Required if you do not use an AWS credential file.
<code>-ve, --verbose</code>	Display verbose log messages. Optional.
<code>-v, --version</code>	Display the version number of the command line tools. Optional.
UPDATE ACCESS POLICY OPTIONS	
<code>-al, --allow IP CIDR all</code>	Add access privileges for a specific IP address or CIDR block. Specify <code>all</code> to allow access from any IP address. Multiple <code>--allow</code> options can be specified to authorize multiple addresses or address ranges. Used in conjunction with the <code>--update</code> option. Optional.
<code>-del, --delete IP CIDR</code>	Delete the allow or deny rule configured for the specified IP address or CIDR block. Used in conjunction with the <code>--update</code> option. Optional.
<code>-de, --deny IP CIDR all</code>	Deny access privileges for a specific IP address or CIDR block. Specify <code>all</code> to block access from all IP addresses. Multiple <code>--deny</code> options can be specified to block multiple addresses or address ranges. Used in conjunction with the <code>--update</code> option. Optional.
<code>-se, --service doc search all</code>	Specify the service to apply the policy changes to: <code>doc</code> , <code>search</code> , or <code>all</code> . All allow, deny, and delete options will be applied to the specified service. Multiple <code>--service</code> options can be specified to apply the same policies to multiple services. Required when using the <code>--update</code> option.
<code>-u, --update</code>	Update the policy with the specified allow, deny and delete options. When using <code>--update</code> , you must also specify at least one <code>--allow</code> , <code>--deny</code> , or <code>--delete</code> option. You must also specify at least one of the domain's endpoints with the <code>--service</code> option. Optional.
POLICY FILE OPTIONS	
<code>-pf, --policy-file FILE</code>	Replace the domain's existing policy document with the specified JSON policy document. Can be specified as a path to a local file or an S3 URI. Optional.
<code>-r, --retrieve</code>	Retrieve the domain's existing policy document.

Optional.

MISCELLANEOUS OPTIONS

`-f, --force` Apply changes to the domain's access policies without confirmation. Can be used in conjunction with either the `--update` or `--policy-file` option. Optional.

EXAMPLES

Authorize addresses in the range 192.0.2.0 to 192.0.2.255 to access all services:

```
cs-configure-access-policies -d mydomain --update --allow 192.0.2.0/24
                             --service all
                             COMMON_OPTIONS
```

Block a particular IP address from accessing the search service:

```
cs-configure-access-policies -d mydomain --update --deny 192.0.2.0
                             --service search
                             COMMON_OPTIONS
```

Allow access to all services from any IP address:

```
cs-configure-access-policies -d mydomain --update --allow all --service all
                             COMMON_OPTIONS
```

Upload a policy document and overwrite the domain's access policies without having to confirm the change:

```
cs-configure-access-policies -d mydomain --policy-file c:\mypolicydoc.json
                             --force
                             COMMON_OPTIONS
```

cs-configure-fields

NAME

`cs-configure-fields` - Define index fields for a domain.

SYNOPSIS

```
cs-configure-fields --name STRING --type text|literal|uint
                   [--option search|nosearch|facet|nofacet|result|noresult]
                   [--source STRING] [--default-value NUM]
                   [--text-processor STRING] [--delete]
                   COMMON_OPTIONS
```

DESCRIPTION

Defines the fields that will be included in a domain's index and specifies which fields can be searched, included in search results, or used as facets.

You can also use this command to delete fields from the domain.

The --option values you can specify for a field depend on the field type:

- text Text fields are always searchable. You can specify the facet, nofacet, result, or noresult options for a text field. A text field can be used as a facet or returned in search results, but not both. By default, text fields are not facet or result enabled.

- literal You can specify the search, nosearch, facet, nofacet, result, or noresult options for a literal field. A literal field can be used as a facet or returned in search results, but not both. By default, literal fields are not searchable, facet-enabled, or result enabled.

- uint Uint fields can always be used as facets and returned in results. No --option values are valid for a uint field.

For more information about configuring indexing options, see the Amazon CloudSearch Developer Guide.

COMMON OPTIONS

- a, --access-key STRING Your AWS access key ID. Used in conjunction with --secret-key. Required if you do not use an AWS credential file.

- c, --aws-credential-file FILE The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the AWS_CREDENTIAL_FILE environment variable or explicitly set your credentials with --access-key and --secret-key.

- d, --domain-name STRING The name of the domain that you are querying or configuring. Required.

- e, --endpoint URL The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the CS_ENDPOINT environment variable or cloudsearch.us-east-1.amazonaws.com if the environment variable is not set. Optional.

- h, --help Display this help message. Optional.

- k, --secret-key STRING Your AWS secret access key. Used in conjunction with --access-key. Required if you do not use an AWS credential file.

- ve, --verbose Display verbose log messages. Optional.

- v, --version Display the version number of the command line tools. Optional.

INDEXING OPTIONS

<code>-dval, --default-value NUM</code>	The default value for a uint field. This value will be added to any document that does not contain at least one value for the field. Optional.
<code>-del, --delete</code>	Delete the field specified by the <code>--name</code> and <code>--type</code> options. Optional.
<code>-fn, --name STRING</code>	The name of the field you are configuring or deleting. Field names must begin with a letter and can contain the following characters: a-z (lower-case letters), 0-9, and <code>_</code> (underscore). Field names must be at least 3 and no more than 28 characters. Required.
<code>-fo, --option OPTION</code>	Configures an option for the field specified by the <code>--name</code> and <code>--type</code> options. Valid values: <code>search</code> , <code>nosearch</code> , <code>facet</code> , <code>nofacet</code> , <code>result</code> , <code>noresult</code> . Text and literal fields cannot have both the <code>facet</code> and <code>result</code> options enabled. By default, text and uint fields are always searchable and uint fields are always facet-enabled. Optional.
<code>-fs, --source FIELD</code>	A source field for a compound field. The value of a compound field is the concatenation of the values of all its sources. Optional.
<code>-tp, --text-processor STRING</code>	The text processor to apply to a text field. Valid values: <code>cs_text_no_stemming</code> . The <code>cs_text_no_stemming</code> processor turns off stemming for a text field. Optional.
<code>--type text literal uint</code>	The type of the field that you are configuring or deleting. Valid values: <code>text</code> , <code>literal</code> , <code>uint</code> . Required.

EXAMPLES

Configure index fields:

```
cs-configure-fields -d mydomain --name title --type text --option result
COMMON_OPTIONS

cs-configure-fields -d mydomain --name people --type text --source actor
--source director --text-processor cs_text_no_stemming
COMMON_OPTIONS

cs-configure-fields -d mydomain --name category --type literal
--options facet
COMMON_OPTIONS

cs-configure-fields --name value --type uint --default-value 100
```

COMMON_OPTIONS

Delete an index field:

```
cs-configure-fields -d mydomain --name obsolete_field --type index-uint
--delete
COMMON_OPTIONS
```

cs-configure-ranking

NAME

cs-configure-ranking - Configure a custom rank expression for a domain.

SYNOPSIS

```
cs-configure-ranking --name STRING --expression EXPRESSION [--delete]
COMMON_OPTIONS
```

DESCRIPTION

Enables you to specify a rank expression to control how search results are ranked. A rank expression is a numeric expression that can reference uint fields and other rank expressions by name. You can also reference a document's default text_relevance score in a rank expression.

A document's text_relevance score is a value from 0 to 1000 (inclusive). To calculate the relevance score, Amazon CloudSearch takes into account how many times the search terms appear (term frequency) and how close the search terms are to each other (proximity).

All of the usual arithmetic, bitwise, boolean, and comparison operators and most common math C library functions can be used in rank expressions. Intermediate results are calculated as double-precision floating point values and the return value is rounded to the nearest integer. If the expression is invalid or evaluates to a negative value, it returns 0.

To use a rank expression to sort search results, you specify &rank=RANKEXPRESSION in your search requests.

For more information about constructing and using rank expressions, see the Amazon CloudSearch API Reference and the Amazon CloudSearch Developer Guide.

COMMON OPTIONS

-a, --access-key STRING	Your AWS access key ID. Used in conjunction with --secret-key. Required if you do not use an AWS credential file.
-c, --aws-credential-file FILE	The path to the file that contains your AWS access key ID and secret access key.


```
Required if you have not set the
AWS_CREDENTIAL_FILE environment variable
or explicitly set your credentials with
--access-key and --secret-key.

-d, --domain-name STRING      The name of the domain that you are
                              querying or configuring. Required.

-e, --endpoint URL           The endpoint for the Amazon CloudSearch
                              Configuration Service. Defaults to the
                              CS_ENDPOINT environment variable or
                              cloudsearch.us-east-1.amazonaws.com
                              if the environment variable is not set.
                              Optional.

-h, --help                   Display this help message. Optional.

-k, --secret-key STRING      Your AWS secret access key. Used in
                              conjunction with --access-key. Required
                              if you do not use an AWS credential file.

-ve, --verbose               Display verbose log messages. Optional.

-v, --version                 Display the version number of the command
                              line tools. Optional.

RANKING OPTIONS

--delete                      Delete the rank expression specified in
                              the --name option. Optional.

--name STRING                 The name of the rank expression you are
                              configuring or deleting. Required.

-ex, --expression EXPRESSION The rank expression to be computed when
                              processing a search request. A rank
                              expression is a numeric expression that
                              can reference uint fields and other rank
                              expressions by name, as well as a document's
                              default text_relevance score. Optional.

EXAMPLES

cs-configure-ranking -d mydomain --name myrankexp
                    --expression ((0.3*myuintfield)/10.0)+((0.7*text_relevance))
                    COMMON_OPTIONS

cs-configure-ranking -d mydomain --name myrankexp
                    --expression text_relevance+myotherrankexp/100000
                    COMMON_OPTIONS
```

cs-configure-text-options

NAME

cs-configure-text-options - Specify domain-specific stopwords, synonyms, and stems.

SYNOPSIS

```
cs-configure-text-options [--stopwords FILE|S3_URI]
                          [--synonyms FILE|S3_URI]
                          [--stems FILE|S3_URI]
                          [--print-stopwords]
                          [--print-synonyms]
                          [--print-stems]
                          COMMON_OPTIONS
```

DESCRIPTION

Amazon CloudSearch gives you control over how your content is indexed by enabling you to specify the following language-specific text options:

- stopwords Words that should typically be ignored both during indexing and at search time because they are either insignificant or so common that including them would result in a massive number of matches. The default stopwords for English are: a, an, and, are, as, at, be, but, by, for, in, is, it, of, on, or, the, to, was.
- synonyms Words that have the same or nearly the same meaning as terms that appear in your corpus. When a user searches for a synonym rather than the indexed term, the results will include documents that contain the indexed term. No synonyms are defined by default.
- stems Define mappings between related words and a common stem. This enables matching on variants of a word. No stems are defined by default.

COMMON OPTIONS

- a, --access-key STRING Your AWS access key ID. Used in conjunction with --secret-key. Required if you do not use an AWS credential file.
- c, --aws-credential-file FILE The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the AWS_CREDENTIAL_FILE environment variable or explicitly set your credentials with --access-key and --secret-key.
- d, --domain-name STRING The name of the domain that you are querying or configuring. Required.

<code>-e, --endpoint URL</code>	The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the <code>CS_ENDPOINT</code> environment variable or <code>cloudsearch.us-east-1.amazonaws.com</code> if the environment variable is not set. Optional.
<code>-h, --help</code>	Display this help message. Optional.
<code>-k, --secret-key STRING</code>	Your AWS secret access key. Used in conjunction with <code>--access-key</code> . Required if you do not use an AWS credential file.
<code>-ve, --verbose</code>	Display verbose log messages. Optional.
<code>-v, --version</code>	Display the version number of the command line tools. Optional.
TEXT OPTIONS	
<code>--stems FILE S3_URI</code>	The path or S3 URI for a stemming dictionary file. Optional. The stemming dictionary file should contain one comma-separated term, stem pair per line. For example: mice, mouse people, person running, run
<code>--stopwords FILE S3_URI</code>	The path or S3 URI for a stopwords dictionary file. Optional. The stopwords dictionary file should contain one stopword per line. For example: the or and The stopwords dictionary must explicitly list each word you want to ignore. Wildcards and regular expressions are not supported.
<code>--synonyms FILE S3_URI</code>	The path or S3 URI for a synonyms dictionary file. Optional. Each line in the file should specify a term followed by a comma-separated list of its synonyms. For example: cat, feline, kitten dog, canine, puppy horse, equine, colt, filly Note: During processing, the synonyms dictionary is converted to JSON and the

```
resulting JSON document cannot exceed
102400 bytes.

-psw, --print-stopwords      List the domain's stopwords. Optional.
-psm, --print-stems          List the domain's stems. Optional.
-psn, --print-synonyms      List the domain's synonyms. Optional.
```

EXAMPLES

```
cs-configure-text-options -d mydomain --stems /home/mystems.txt
                           --stopwords /home/mystopwords.txt
                           --synonyms /home/mysynonyms.txt
                           COMMON_OPTIONS

cs-configure-text-options -d mydomain --print-stopwords COMMON_OPTIONS
```

cs-create-domain

NAME

cs-create-domain - Create a new Amazon CloudSearch domain.

SYNOPSIS

```
cs-create-domain --domain-name STRING [--wait]
                 COMMON_OPTIONS
```

DESCRIPTION

Creates a search domain with the name specified by the `--domain-name` option. Domain names must begin with a letter or number and can contain the following characters: a-z, 0-9, and -. Uppercase letters and underscores are not allowed. Domain names must be at least 3 and no more than 28 characters.

By default, this command returns immediately. If you specify the `--wait` option, `cs-create-domain` will return once the domain is created.

COMMON OPTIONS

```
-a, --access-key STRING      Your AWS access key ID. Used in
                              conjunction with --secret-key.
                              Required if you do not use an AWS
                              credential file.

-c, --aws-credential-file FILE The path to the file that contains your
                              AWS access key ID and secret access key.
                              Required if you have not set the
                              AWS_CREDENTIAL_FILE environment variable
                              or explicitly set your credentials with
                              --access-key and --secret-key.
```

```
-d, --domain-name STRING      The name of the domain that you are
                               querying or configuring. Required.

-e, --endpoint URL           The endpoint for the Amazon CloudSearch
                               Configuration Service. Defaults to the
                               CS_ENDPOINT environment variable or
                               cloudsearch.us-east-1.amazonaws.com
                               if the environment variable is not set.
                               Optional.

-h, --help                   Display this help message. Optional.

-k, --secret-key STRING      Your AWS secret access key. Used in
                               conjunction with --access-key. Required
                               if you do not use an AWS credential file.

-ve, --verbose               Display verbose log messages. Optional.

-v, --version                 Display the version number of the command
                               line tools. Optional.

DOMAIN OPTIONS

-w, --wait                   Wait for domain creation to complete
                               before returning. Optional.

EXAMPLES

cs-create-domain -d mydomain --wait COMMON_OPTIONS

cs-create-domain -d myeu-domain --endpoint cloudsearch.eu-west-1.amazonaws.com
```

cs-configure-from-sdf

NAME

cs-configure-from-sdf - Define index fields for a domain based on the contents of one or more SDF batches.

SYNOPSIS

```
cs-configure-from-sdf --source FILE|S3_URI [--replace] [--force] COMMON_OPTIONS
```

DESCRIPTION

Scans SDF batches specified with the `--source` option and configures index fields for all of the document fields. Prompts for confirmation before making any changes unless you specify the `--force` option.

By default, fields that have already been configured are left as-is. You can use the `--replace` option to overwrite the existing configuration.

COMMON OPTIONS

- a, --access-key STRING Your AWS access key ID. Used in conjunction with --secret-key. Required if you do not use an AWS credential file.
- c, --aws-credential-file FILE The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the AWS_CREDENTIAL_FILE environment variable or explicitly set your credentials with --access-key and --secret-key.
- d, --domain-name STRING The name of the domain that you are querying or configuring. Required.
- e, --endpoint URL The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the CS_ENDPOINT environment variable or cloudsearch.us-east-1.amazonaws.com if the environment variable is not set. Optional.
- h, --help Display this help message. Optional.
- k, --secret-key STRING Your AWS secret access key. Used in conjunction with --access-key. Required if you do not use an AWS credential file.
- ve, --verbose Display verbose log messages. Optional.
- v, --version Display the version number of the command line tools. Optional.

FIELD OPTIONS

- f, --force Apply changes to the domain's configuration without confirmation. Optional.
- re, --replace Upload configuration information for all identified fields and overwrite the configuration of any fields that were already defined. (Prompts for confirmation unless you also specify --force.) Optional.
- s, --source FILE|S3 URI The path to a file or an S3 URI that contains the data you want to scan. You can specify multiple files or S3 URIs. For example: --source SDF1.json SDF2.json. Required.

EXAMPLES

cs-configure-from-sdf -d mydomain --source s3://mybucket/mydata COMMON_OPTIONS

cs-delete-domain

NAME

cs-delete-domain - Permanently delete the specified domain and all of its data.

SYNOPSIS

```
cs-delete-domain --domain-name STRING [--force] COMMON_OPTIONS
```

DESCRIPTION

Deletes the search domain specified by the --domain-name option.

COMMON OPTIONS

-a, --access-key STRING	Your AWS access key ID. Used in conjunction with --secret-key. Required if you do not use an AWS credential file.
-c, --aws-credential-file FILE	The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the AWS_CREDENTIAL_FILE environment variable or explicitly set your credentials with --access-key and --secret-key.
-d, --domain-name STRING	The name of the domain that you are querying or configuring. Required.
-e, --endpoint URL	The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the CS_ENDPOINT environment variable or cloudsearch.us-east-1.amazonaws.com if the environment variable is not set. Optional.
-h, --help	Display this help message. Optional.
-k, --secret-key STRING	Your AWS secret access key. Used in conjunction with --access-key. Required if you do not use an AWS credential file.
-ve, --verbose	Display verbose log messages. Optional.
-v, --version	Display the version number of the command line tools. Optional.

DELETE DOMAIN OPTIONS

-f, --force	Delete the domain without prompting for confirmation. Optional.
-------------	---

EXAMPLES

Delete a domain without prompting for confirmation:

```
cs-delete-domain -d mydomain --force COMMON_OPTIONS
```

cs-describe-domain

NAME

`cs-describe-domain` - Display information about a domain, including its status and endpoints.

SYNOPSIS

```
cs-describe-domain [--show-all] COMMON_OPTIONS
```

DESCRIPTION

Display information about your configured domains. If the `--domain-name` option is specified, `cs-describe-domain` only shows information for the specified domain.

This command returns a table that contains the following information about the domain(s):

Domain Name	The name of the domain.
Document Service Endpoint	The endpoint through which you can submit document updates.
Search Endpoint	The endpoint through which you can submit search requests.
Searchable Documents	The number of documents that have been indexed.
Index Fields	The name and type of each configured index field. Only shown when <code>--show-all</code> is specified.
Ranking Fields	The name and type of each ranking field. Only shown when <code>--show-all</code> is specified.
SearchPartitionCount	The number of partitions being used to hold the search index.
SearchInstanceCount	The number of search instances being used to process search requests.
SearchInstanceType	The Amazon EC2 instance type being used to process search requests.

The domain status also indicates whether or not the index needs to be

rebuilt to process configuration changes.

COMMON OPTIONS

- | | |
|---|---|
| <code>-a, --access-key STRING</code> | Your AWS access key ID. Used in conjunction with <code>--secret-key</code> . Required if you do not use an AWS credential file. |
| <code>-c, --aws-credential-file FILE</code> | The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the <code>AWS_CREDENTIAL_FILE</code> environment variable or explicitly set your credentials with <code>--access-key</code> and <code>--secret-key</code> . |
| <code>-d, --domain-name STRING</code> | The name of the domain that you are querying or configuring. Required. |
| <code>-e, --endpoint URL</code> | The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the <code>CS_ENDPOINT</code> environment variable or <code>cloudsearch.us-east-1.amazonaws.com</code> if the environment variable is not set. Optional. |
| <code>-h, --help</code> | Display this help message. Optional. |
| <code>-k, --secret-key STRING</code> | Your AWS secret access key. Used in conjunction with <code>--access-key</code> . Required if you do not use an AWS credential file. |
| <code>-ve, --verbose</code> | Display verbose log messages. Optional. |
| <code>-v, --version</code> | Display the version number of the command line tools. Optional. |

DESCRIBE DOMAIN OPTIONS

- | | |
|-------------------------------|--|
| <code>-all, --show-all</code> | Display all available information for the domain, including configured fields. Optional. |
|-------------------------------|--|

EXAMPLES

Get information about a particular domain:

```
cs-describe-domain -d mydomain --show-all COMMON_OPTIONS
```

cs-index-documents

NAME

cs-index-documents - Index a domain's documents.

SYNOPSIS

cs-index-documents COMMON_OPTIONS

DESCRIPTION

Builds and deploys a complete index for the domain specified by the --domain-name option.

COMMON OPTIONS

- | | |
|--------------------------------|---|
| -a, --access-key STRING | Your AWS access key ID. Used in conjunction with --secret-key. Required if you do not use an AWS credential file. |
| -c, --aws-credential-file FILE | The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the AWS_CREDENTIAL_FILE environment variable or explicitly set your credentials with --access-key and --secret-key. |
| -d, --domain-name STRING | The name of the domain that you are indexing. Required. |
| -e, --endpoint URL | The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the CS_ENDPOINT environment variable or cloudsearch.us-east-1.amazonaws.com if the environment variable is not set. Optional. |
| -h, --help | Display this help message. Optional. |
| -k, --secret-key STRING | Your AWS secret access key. Used in conjunction with --access-key. Required if you do not use an AWS credential file. |
| -ve, --verbose | Display verbose log messages. Optional. |
| -v, --version | Display the version number of the command line tools. Optional. |

INDEX DOCUMENTS OPTIONS

No specific options.

EXAMPLES

cs-index-documents -d mydomain COMMON_OPTIONS

cs-post-sdf

NAME

cs-post-sdf - Upload the SDF documents that you want to index and search.

SYNOPSIS

```
cs-post-sdf --source FILE|S3_URI COMMON_OPTIONS
```

DESCRIPTION

Update the contents of the domain specified by the `--domain-name` option with the documents specified by the `--source` option. The source documents must be specified in the SDF format, which can be generated from most types of files using the `cs-generate-sdf` command.

COMMON OPTIONS

<code>-a, --access-key STRING</code>	Your AWS access key ID. Used in conjunction with <code>--secret-key</code> . Required if you do not use an AWS credential file.
<code>-c, --aws-credential-file FILE</code>	The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the <code>AWS_CREDENTIAL_FILE</code> environment variable or explicitly set your credentials with <code>--access-key</code> and <code>--secret-key</code> .
<code>-d, --domain-name STRING</code>	The name of the domain that you are querying or configuring. Required.
<code>-e, --endpoint URL</code>	The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the <code>CS_ENDPOINT</code> environment variable or <code>cloudsearch.us-east-1.amazonaws.com</code> if the environment variable is not set. Optional.
<code>-h, --help</code>	Display this help message. Optional.
<code>-k, --secret-key STRING</code>	Your AWS secret access key. Used in conjunction with <code>--access-key</code> . Required if you do not use an AWS credential file.
<code>-ve, --verbose</code>	Display verbose log messages. Optional.
<code>-v, --version</code>	Display the version number of the command line tools. Optional.

UPDATE DOCUMENTS OPTIONS

<code>-s, --source FILE S3_URI</code>	The path to a file or an S3 URI that contains
---------------------------------------	---

the SDF data you want to upload. You can specify multiple files or S3 URIs. For example: `--source SDF1.json SDF2.json`. Required.

EXAMPLES

```
cs-post-sdf -d movies --source s3://mybucket/mydata COMMON_OPTIONS
```

SEE ALSO

```
cs-generate-sdf
```

Experimental Tools for Amazon CloudSearch

The following tools are provided on an experimental basis. Please let us know if you would like to see them fully-supported and enhanced in future releases:

- [cs-generate-sdf](#) (p. 158)

cs-generate-sdf

NAME

```
cs-generate-sdf - Experimental tool for analyzing the data you want to index and automatically generating SDF batches for indexing.
```

SYNOPSIS

```
cs-generate-sdf --source PATH|S3_URI|DDB_TABLE [--output PATH|S3_URI]
  [--modified-after yyyy-MM-ddTHH:mm:ssZ]
  [--exclude-metadata] [--exclude-content]
  [--single-doc-per-csv] [--multivalued FIELDS]
  [--sdf-format json|xml] [--docid-prefix STRING]
  [--doc-version NUM] [--batch-size MB]
  [--batch-docs NUM]
  [--num-rows NUM] [--dynamodb-rcu-percent NUM]
  [--start-hash-key STRING] [--start-range-key STRING]
COMMON_OPTIONS
```

DESCRIPTION

Analyze your data and generate SDF (Search Data Format) batches that can be submitted to Amazon CloudSearch for indexing. If you specify the `--domain` option, the generated SDF is automatically uploaded to your search domain. Alternatively, you can specify the `--output` option to save the SDF batches to your local file system or an S3 bucket and submit them yourself using the `cs-post-sdf` command.

If you are processing local data or data in S3, the `cs-generate-sdf` command can generate SDF batches from the following content types:

```
text/csv
text/html
text/plain
```

```
application/json
application/msword
application/pdf
application/vnd.ms-excel
application/vnd.ms-powerpoint
application/vnd.openxmlformats-officedocument.presentationml.presentation
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
application/vnd.openxmlformats-officedocument.wordprocessingml.document
application/xhtml+xml
application/xml
```

For most file types, including JSON and XML, `cs-generate-sdf` adds a single add document operation to the SDF batch for each source file. If metadata is available for the file, the metadata is mapped to corresponding document fields--the fields generated from the document metadata vary depending on the file type. The contents of the source file are parsed into a single text field. If the file contains more than 1 MB of data, the data mapped to the text field is truncated so the document does not exceed 1 MB.

CSV files are handled differently. When processing CSV files, `cs-generate-sdf` uses the contents of the first row to define the document fields, and creates a separate document for each following row. If there is a column header called `docid`, the values in that column are used as the document IDs. If necessary, the `docid` values are normalized to conform to the allowed character set: a-z (lower-case letters), 0-9, and `_` (underscore). If there is no `docid` column, a unique ID is generated for each document based on the filename and row number. Similarly, if there is a column called `version`, the values in that column are used as the versions for the document updates. Version numbers must be specified as 32-bit unsigned integers. If there is no `version` column, version numbers are generated automatically based on a timestamp.

If you are processing multiple types of files, CSV files are parsed row-by-row, and non-CSV files are treated as individual documents.

You can specify the `--single-doc-per-csv` option to override the default behavior and treat each CSV file as a single document. Specifying the `--single-doc-per-csv` option has no effect on non-CSV files.

Note: Currently, only CSV files are parsed to automatically extract custom field data and generate multiple documents. When processing XML and JSON files, each file is treated as a separate document and the contents of the file are used to populate a single text field.

If you are extracting data from a DynamoDB table, an SDF document is generated for each row that's read from the table. You can use the `--num-rows`, `--start-hash-key`, and `--start-range` key options to generate SDF from selected rows in the table. The `--dynamodb-rcu-percent` option enables you to limit the percentage of DynamoDB read capacity units consumed when extracting data from the table.

COMMON OPTIONS

<code>-a, --access-key STRING</code>	Your AWS access key ID. Used in conjunction with <code>--secret-key</code> . Required if you do not use an AWS credential file.
--------------------------------------	---

-c, --aws-credential-file FILE	The path to the file that contains your AWS access key ID and secret access key. Required if you have not set the AWS_CREDENTIAL_FILE environment variable or explicitly set your credentials with --access-key and --secret-key.
-d, --domain-name STRING	The name of the domain that you are querying or configuring. Required.
-e, --endpoint URL	The endpoint for the Amazon CloudSearch Configuration Service. Defaults to the CS_ENDPOINT environment variable or cloudsearch.us-east-1.amazonaws.com if the environment variable is not set. Optional.
-h, --help	Display this help message. Optional.
-k, --secret-key STRING	Your AWS secret access key. Used in conjunction with --access-key. Required if you do not use an AWS credential file.
-ve, --verbose	Display verbose log messages. Optional.
-v, --version	Display the version number of the command line tools. Optional.
BASIC SDF OPTIONS	
-o, --output PATH S3_URI	The local directory or S3 bucket where you want to save the generated SDF batches. Required if you do not specify the --domain option to upload the generated SDF batches to a search domain. Optional.
-s, --source PATH S3_URI DDB_TABLE	The local directory, file, S3 bucket, or a DynamoDB table that contains the data that you want to create SDF batches from. You can process data from multiple locations by specifying multiple --source options. Accepts Apache-ant style wildcards such as */** for files and S3 prefixes. Required.
ADVANCED SDF OPTIONS	
-bd, --batch-docs NUM	The maximum number of documents in a batch. Optional.
-bs, --batch-size MB	The maximum batch size in MB. Defaults to 5MB. Optional.

<code>-sdpc, --single-doc-per-csv</code>	Treat the CSV file as a single document. If this option is specified, the contents of the CSV file will be treated as a single text field. This option is not valid if the <code>-mv</code> option is specified and it has no effect on non-CSV files. Optional.
<code>-mv, --multivalued FIELDS</code>	Treat the specified fields as multi-valued fields when processing the CSV file. Specify multiple fields as a space-separated list. If no fields are specified, all fields other than <code>docid</code> and <code>version</code> are processed as multi-valued fields. This option is not valid if the <code>-sdpc</code> option is specified and it has no effect on non-CSV files. Optional.
<code>-dp, --docid-prefix STRING</code>	The prefix to prepend to the document ID while processing CSV data. If not specified, the filename is used as the <code>--docid-prefix</code> . The <code>docid</code> column is used as the document ID if it is included in the CSV data; otherwise, the row number is used as the document ID. Optional.
<code>-dv, --doc-version NUM</code>	The version number to use for all of the generated SDF documents. Defaults to 1. Optional.
<code>-ec, --exclude-content</code>	Do not include the content of the source files in the generated SDF documents, only process the metadata. Optional.
<code>-em, --exclude-metadata</code>	Do not include the metadata of the source files in the generated SDF documents, only process the content. Optional.
<code>-format, --sdf-format json xml</code>	The format of the generated SDF documents: json or xml. Defaults to json. Optional.

`-m, --modified-after TIMESTAMP` Only process files or S3 objects modified after the specified date and time. Specified in RFC 822 time zone format (yyyy-MM-dd'T'HH:mm:ssZ). For example, 2012-12-12T01:00:00GMT. Optional.

DynamoDB SOURCE OPTIONS

`-n, --num-rows` The maximum number of rows to read from the DynamoDB table. Optional. By default, the entire table is read.

`-drp, --dynamodb-rcu-percent` The maximum percentage of configured read capacity units to use while reading from the DynamoDB table. Optional. By default, the maximum number of read capacity units is set to 20% the table's configured read capacity units.

`-shk, --start-hash-key` The hash attribute of the item in the DynamoDB table where you want to begin reading. If the table has a hash and range type primary key, the `--start-range-key` option must also be specified. Optional. By default, the table is read starting with the first item.

`-srk, --start-range-key` The range attribute of the item in the DynamoDB table where you want to begin reading. Required if `--start-hash-key` is specified and the DynamoDB table has a hash and range type primary key. Not used if the table has a hash type primary key. Optional.

EXAMPLES

Generate an SDF batch from a plain text file:

```
cs-generate-sdf --source c:\myAmazingDataSet\data1.txt
                --output c:\myAmazingDataSet\SDF\batch
                COMMON_OPTIONS
```

Generate a single document for each CSV file:

```
cs-generate-sdf --source c:\myAmazingDataSet\*.csv -sdpc
                --output c:\myAmazingDataSet\SDF\batch
                COMMON_OPTIONS
```

Generate an SDF batch from multiple documents:

```
cs-generate-sdf --source c:\myAmazingDataSet\data1.xml
                --source c:\myAmazingDataSet\data2.xml
                --source c:\myAmazingDataSet\data3.xml
```



```
--output c:\myAmazingDataSet\SDF\batch  
COMMON_OPTIONS
```

Generate SDF batches from all HTML documents in a directory:

```
cs-generate-sdf --source c:\myAmazingDataSet\*.html  
--output c:\myAmazingDataSet\SDF\batch  
COMMON_OPTIONS
```

Generate SDF batches from all Word or PDF documents in a directory:

```
cs-generate-sdf --source c:\myAmazingDataSet\*.doc  
--source c:\myAmazingDataSet\*.pdf  
--output c:\myAmazingDataSet\SDF\batch  
COMMON_OPTIONS
```

Generate SDF batches from all recognized file types:

```
cs-generate-sdf --source c:\myAmazingDataSet\  
--output c:\myAmazingDataSet\SDF\batch  
COMMON_OPTIONS
```

Generate SDF batches and upload them to your search domain:

```
cs-generate-sdf -d mydomain --source c:\myAmazingDataSet\  
COMMON_OPTIONS
```

Generate SDF batches from a DynamoDB table and upload them to your search domain:

```
cs-generate-sdf -d mydomain --source ddb://myDDBTable  
COMMON_OPTIONS
```

Generate SDF batches from a DynamoDB table and save them to your local file system:

```
cs-generate-sdf --source ddb://myDDBTable  
--output c:\myAmazingDataSet\SDF\batch  
COMMON_OPTIONS
```

Generate SDF batches from selected rows in a DynamoDB table using no more than 10% of read capacity units configured for the DynamoDB table:

```
cs-generate-sdf --source ddb://myDDBTable  
--output c:\myAmazingDataSet\SDF\batch  
--num-rows 100  
--dynamodb-rcu-percent 10  
--start-hash-key "Amazon DynamoDB"  
--start-range-key "DynamoDB Thread 100"  
COMMON_OPTIONS
```

SEE ALSO

cs-post-sdf

Amazon CloudSearch Configuration API Reference

Topics

- [Actions \(p. 165\)](#)
- [Data Types \(p. 194\)](#)
- [Common Parameters \(p. 213\)](#)
- [Common Errors \(p. 215\)](#)

You use the Amazon CloudSearch Configuration API to create, configure, and manage search domains. The configuration service is accessed through a region-specific endpoint, `cloudsearch.region.amazonaws.com`. For example, `cloudsearch.us-east-1.amazonaws.com`. For a current list of supported regions and endpoints, see [Regions and Endpoints](#).

You submit Amazon CloudSearch configuration requests using the AWS Query protocol. AWS Query requests are HTTP or HTTPS requests submitted via HTTP GET or POST with a Query parameter named Action. The API version must be specified in all requests. The current Amazon CloudSearch API version is 2011-02-01.

Requests submitted to the Configuration API are authenticated using your AWS access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about getting credentials, see [How Do I Get Security Credentials?](#) in the *AWS General Reference*.

You must include authorization parameters and a digital signature in every request. Amazon CloudSearch supports AWS Signature Version 4. For detailed signing instructions, see [Signature V4 Signing Process](#) in the *AWS General Reference*.

The other APIs you use to interact with Amazon CloudSearch are:

- [Amazon CloudSearch Document Service API Reference \(p. 217\)](#)—Submit the data you want to search.
- [Amazon CloudSearch Search API Reference \(p. 227\)](#)—Search your domain.

Actions

The following actions are supported:

- [CreateDomain](#) (p. 166)
- [DefineIndexField](#) (p. 167)
- [DefineRankExpression](#) (p. 169)
- [DeleteDomain](#) (p. 171)
- [DeleteIndexField](#) (p. 172)
- [DeleteRankExpression](#) (p. 174)
- [DescribeDefaultSearchField](#) (p. 176)
- [DescribeDomains](#) (p. 177)
- [DescribeIndexFields](#) (p. 178)
- [DescribeRankExpressions](#) (p. 179)
- [DescribeServiceAccessPolicies](#) (p. 180)
- [DescribeStemmingOptions](#) (p. 181)
- [DescribeStopwordOptions](#) (p. 182)
- [DescribeSynonymOptions](#) (p. 183)
- [IndexDocuments](#) (p. 184)
- [UpdateDefaultSearchField](#) (p. 185)
- [UpdateServiceAccessPolicies](#) (p. 187)
- [UpdateStemmingOptions](#) (p. 189)
- [UpdateStopwordOptions](#) (p. 191)
- [UpdateSynonymOptions](#) (p. 193)

CreateDomain

Description

Creates a new search domain.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `CreateDomainResult`.

DomainStatus

The current status of the search domain.

Type: [DomainStatus \(p. 200\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

DefineIndexField

Description

Configures an `IndexField` for the search domain. Used to create new fields and modify existing ones. If the field exists, the new configuration replaces the old one. You can configure a maximum of 200 index fields.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

IndexField

Defines a field in the index, including its name, type, and the source of its data. The `IndexFieldType` indicates which of the options will be present. It is invalid to specify options for a type other than the `IndexFieldType`.

Type: [IndexField \(p. 202\)](#)

Required: Yes

Response Elements

The following element is returned in a structure named `DefineIndexFieldResult`.

IndexField

The value of an `IndexField` and its current status.

Type: [IndexFieldStatus \(p. 203\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DefineRankExpression

Description

Configures a `RankExpression` for the search domain. Used to create new rank expressions and modify existing ones. If the expression exists, the new configuration replaces the old one. You can configure a maximum of 50 rank expressions.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

RankExpression

A named expression that can be evaluated at search time and used for ranking or thresholding in a search query.

Type: [NamedRankExpression \(p. 204\)](#)

Required: Yes

Response Elements

The following element is returned in a structure named `DefineRankExpressionResult`.

RankExpression

The value of a `RankExpression` and its current status.

Type: [RankExpressionStatus \(p. 206\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DeleteDomain

Description

Permanently deletes a search domain and all of its data.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DeleteDomainResult`.

DomainStatus

The current status of the search domain.

Type: [DomainStatus \(p. 200\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

DeleteIndexField

Description

Removes an `IndexField` from the search domain.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

IndexFieldName

A string that represents the name of an index field. Field names must begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). Uppercase letters and hyphens are not allowed. The names "body", "docid", and "text_relevance" are reserved and cannot be specified as field or rank expression names.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Response Elements

The following element is returned in a structure named `DeleteIndexFieldResult`.

IndexField

The value of an `IndexField` and its current status.

Type: [IndexFieldStatus \(p. 203\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DeleteRankExpression

Description

Removes a `RankExpression` from the search domain.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

RankName

The name of the `RankExpression` to delete.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Response Elements

The following element is returned in a structure named `DeleteRankExpressionResult`.

RankExpression

The value of a `RankExpression` and its current status.

Type: [RankExpressionStatus \(p. 206\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeDefaultSearchField

Description

Gets the default search field configured for the search domain.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeDefaultSearchFieldResult`.

DefaultSearchField

The name of the `IndexField` to use for search requests issued with the `q` parameter. The default is the empty string, which automatically searches all text fields.

Type: [DefaultSearchFieldStatus \(p. 196\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeDomains

Description

Gets information about the search domains owned by this account. Can be limited to specific domains. Shows all domains by default.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainNames.member.N

Limits the DescribeDomains response to the specified search domains.

Type: String list

Required: No

Response Elements

The following element is returned in a structure named `DescribeDomainsResult`.

DomainStatusList

The current status of all of your search domains.

Type: [DomainStatus \(p. 200\)](#) list

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

DescribeIndexFields

Description

Gets information about the index fields configured for the search domain. Can be limited to specific fields by name. Shows all fields by default.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

FieldNames.member.N

Limits the `DescribeIndexFields` response to the specified fields.

Type: String list

Required: No

Response Elements

The following element is returned in a structure named `DescribeIndexFieldsResult`.

IndexFields

The index fields configured for the domain.

Type: [IndexFieldStatus \(p. 203\)](#) list

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeRankExpressions

Description

Gets the rank expressions configured for the search domain. Can be limited to specific rank expressions by name. Shows all rank expressions by default.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

RankNames.member.N

Limits the `DescribeRankExpressions` response to the specified fields.

Type: String list

Required: No

Response Elements

The following element is returned in a structure named `DescribeRankExpressionsResult`.

RankExpressions

The rank expressions configured for the domain.

Type: [RankExpressionStatus \(p. 206\)](#) list

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeServiceAccessPolicies

Description

Gets information about the resource-based policies that control access to the domain's document and search services.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeServiceAccessPoliciesResult`.

AccessPolicies

A `PolicyDocument` that specifies access policies for the search domain's services, and the current status of those policies.

Type: [AccessPoliciesStatus \(p. 195\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeStemmingOptions

Description

Gets the stemming dictionary configured for the search domain.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeStemmingOptionsResult`.

Stems

The stemming options configured for this search domain and the current status of those options.

Type: [StemmingOptionsStatus \(p. 209\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeStopwordOptions

Description

Gets the stopwords configured for the search domain.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeStopwordOptionsResult`.

Stopwords

The stopwords options configured for this search domain and the current status of those options.

Type: [StopwordOptionsStatus \(p. 210\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

DescribeSynonymOptions

Description

Gets the synonym dictionary configured for the search domain.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `DescribeSynonymOptionsResult`.

Synonyms

The synonym options configured for this search domain and the current status of those options.

Type: [SynonymOptionsStatus \(p. 210\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

IndexDocuments

Description

Tells the search domain to start indexing its documents using the latest text processing options and `IndexFields`. This operation must be invoked to make options whose [OptionStatus \(p. 205\)](#) has `OptionState` of `RequiresIndexDocuments` visible in search results.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `IndexDocumentsResult`.

FieldNames

The names of the fields that are currently being processed due to an `IndexDocuments` action.

Type: String list

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

UpdateDefaultSearchField

Description

Configures the default search field for the search domain. The default search field is the text field that is searched when a search request does not specify which fields to search. By default, it is configured to include the contents of all of the domain's text fields.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DefaultSearchField

The text field to search if the search request does not specify which field to search. The default search field is used when search terms are specified with the `q` parameter, or if a match expression specified with the `bq` parameter does not constrain the search to a particular field. The default is an empty string, which automatically searches all text fields.

Type: String

Required: Yes

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `UpdateDefaultSearchFieldResult`.

DefaultSearchField

The value of the `DefaultSearchField` configured for this search domain and its current status.

Type: [DefaultSearchFieldStatus \(p. 196\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

UpdateServiceAccessPolicies

Description

Configures the policies that control access to the domain's document and search services. The maximum size of an access policy document is 100 KB.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

AccessPolicies

An IAM access policy as described in [The Access Policy Language](#) in *Using AWS Identity and Access Management*. The maximum size of an access policy document is 100 KB.

```
Example: { "Statement": [ { "Effect": "Allow", "Action": "*", "Resource": "arn:aws:cs:us-east-1:1234567890:search/movies", "Condition": { "IpAddress": { "aws:SourceIp": [ "203.0.113.1/32" ] } } }, { "Effect": "Allow", "Action": "*", "Resource": "arn:aws:cs:us-east-1:1234567890:documents/movies", "Condition": { "IpAddress": { "aws:SourceIp": [ "203.0.113.1/32" ] } } } ] }
```

Type: String

Required: Yes

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Response Elements

The following element is returned in a structure named `UpdateServiceAccessPoliciesResult`.

AccessPolicies

A `PolicyDocument` that specifies access policies for the search domain's services, and the current status of those policies.

Type: [AccessPoliciesStatus \(p. 195\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

UpdateStemmingOptions

Description

Configures a stemming dictionary for the search domain. The stemming dictionary is used during indexing and when processing search requests. The maximum size of the stemming dictionary is 500 KB.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Stems

Maps terms to their stems, serialized as a JSON document. The document has a single object with one property "stems" whose value is an object mapping terms to their stems. The maximum size of a stemming document is 500 KB. Example: { "stems": { "people": "person", "walking": "walk" } }

Type: String

Required: Yes

Response Elements

The following element is returned in a structure named `UpdateStemmingOptionsResult`.

Stems

The stemming options configured for this search domain and the current status of those options.

Type: [StemmingOptionsStatus \(p. 209\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

UpdateStopwordOptions

Description

Configures stopwords for the search domain. Stopwords are used during indexing and when processing search requests. The maximum size of the stopwords dictionary is 10 KB.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Stopwords

Lists stopwords serialized as a JSON document. The document has a single object with one property "stopwords" whose value is an array of strings. The maximum size of a stopwords document is 10 KB. Example: { "stopwords": ["a", "an", "the", "of"] }

Type: String

Required: Yes

Response Elements

The following element is returned in a structure named `UpdateStopwordOptionsResult`.

Stopwords

The stopwords options configured for this search domain and the current status of those options.

Type: [StopwordOptionsStatus \(p. 210\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

UpdateSynonymOptions

Description

Configures a synonym dictionary for the search domain. The synonym dictionary is used during indexing to configure mappings for terms that occur in text fields. The maximum size of the synonym dictionary is 100 KB.

Request Parameters

For information about the common parameters that all actions use, see [Common Parameters \(p. 213\)](#).

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

Synonyms

Maps terms to their synonyms, serialized as a JSON document. The document has a single object with one property "synonyms" whose value is an object mapping terms to their synonyms. Each synonym is a simple string or an array of strings. The maximum size of a stopwords document is 100 KB. Example: { "synonyms": { "cat": ["feline", "kitten"], "puppy": "dog" } }

Type: String

Required: Yes

Response Elements

The following element is returned in a structure named `UpdateSynonymOptionsResult`.

Synonyms

The synonym options configured for this search domain and the current status of those options.

Type: [SynonymOptionsStatus \(p. 210\)](#)

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 215\)](#).

Base

An error occurred while processing the request.

HTTP Status Code: 400

Internal

An internal error occurred while processing the request. If this problem persists, report an issue from the [Service Health Dashboard](#).

HTTP Status Code: 500

InvalidType

The request was rejected because it specified an invalid type definition.

HTTP Status Code: 409

LimitExceeded

The request was rejected because a resource limit has already been met.

HTTP Status Code: 409

ResourceNotFound

The request was rejected because it attempted to reference a resource that does not exist.

HTTP Status Code: 409

Data Types

The Amazon CloudSearch Configuration Service API contains several data types that various actions use. This section describes each data type in detail.

Note

The order of each element in the response is not guaranteed. Applications should not assume a particular order.

The following data types are supported:

- [AccessPoliciesStatus](#) (p. 195)
- [CreateDomainResult](#) (p. 195)
- [DefaultSearchFieldStatus](#) (p. 196)
- [DefineIndexFieldResult](#) (p. 196)
- [DefineRankExpressionResult](#) (p. 197)
- [DeleteDomainResult](#) (p. 197)
- [DeleteIndexFieldResult](#) (p. 197)
- [DeleteRankExpressionResult](#) (p. 197)
- [DescribeDefaultSearchFieldResult](#) (p. 198)
- [DescribeDomainsResult](#) (p. 198)
- [DescribeIndexFieldsResult](#) (p. 198)
- [DescribeRankExpressionsResult](#) (p. 199)
- [DescribeServiceAccessPoliciesResult](#) (p. 199)
- [DescribeStemmingOptionsResult](#) (p. 199)
- [DescribeStopwordOptionsResult](#) (p. 200)
- [DescribeSynonymOptionsResult](#) (p. 200)
- [DomainStatus](#) (p. 200)
- [IndexDocumentsResult](#) (p. 202)
- [IndexField](#) (p. 202)
- [IndexFieldStatus](#) (p. 203)
- [LiteralOptions](#) (p. 204)
- [NamedRankExpression](#) (p. 204)
- [OptionStatus](#) (p. 205)
- [RankExpressionStatus](#) (p. 206)
- [ServiceEndpoint](#) (p. 207)
- [SourceAttribute](#) (p. 207)

- [SourceData](#) (p. 208)
- [SourceDataMap](#) (p. 208)
- [SourceDataTrimTitle](#) (p. 209)
- [StemmingOptionsStatus](#) (p. 209)
- [StopwordOptionsStatus](#) (p. 210)
- [SynonymOptionsStatus](#) (p. 210)
- [TextOptions](#) (p. 211)
- [UIntOptions](#) (p. 212)
- [UpdateDefaultSearchFieldResult](#) (p. 212)
- [UpdateServiceAccessPoliciesResult](#) (p. 212)
- [UpdateStemmingOptionsResult](#) (p. 213)
- [UpdateStopwordOptionsResult](#) (p. 213)
- [UpdateSynonymOptionsResult](#) (p. 213)

AccessPoliciesStatus

Description

A `PolicyDocument` that specifies access policies for the search domain's services, and the current status of those policies.

Contents

Options

An IAM access policy as described in [The Access Policy Language in Using AWS Identity and Access Management](#). The maximum size of an access policy document is 100 KB.

Example:

```
{ "Statement": [ { "Effect": "Allow", "Action": "*", "Resource": "arn:aws:cs:us-east-1:1234567890:search/movies", "Condition": { "IpAddress": { "aws:SourceIp": [ "203.0.113.1/32" ] } } }, { "Effect": "Allow", "Action": "*", "Resource": "arn:aws:cs:us-east-1:1234567890:documents/movies", "Condition": { "IpAddress": { "aws:SourceIp": [ "203.0.113.1/32" ] } } } ] }
```

Type: String

Required: Yes

Status

The status of an option, including when it was last updated and whether it is actively in use for searches.

Type: [OptionStatus](#) (p. 205)

Required: Yes

CreateDomainResult

Description

A response message that contains the status of a newly created domain.

Contents

DomainStatus

The current status of the search domain.

Type: [DomainStatus](#) (p. 200)

Required: No

DefaultSearchFieldStatus

Description

The value of the `DefaultSearchField` configured for this search domain and its current status.

Contents

Options

The name of the `IndexField` to use as the default search field. The default is an empty string, which automatically searches all text fields.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

Status

The status of an option, including when it was last updated and whether it is actively in use for searches.

Type: [OptionStatus](#) (p. 205)

Required: Yes

DefineIndexFieldResult

Description

A response message that contains the status of an updated index field.

Contents

IndexField

The value of an `IndexField` and its current status.

Type: [IndexFieldStatus](#) (p. 203)

Required: Yes

DefineRankExpressionResult

Description

A response message that contains the status of an updated `RankExpression`.

Contents

RankExpression

The value of a `RankExpression` and its current status.

Type: [RankExpressionStatus](#) (p. 206)

Required: Yes

DeleteDomainResult

Description

A response message that contains the status of a newly deleted domain, or no status if the domain has already been completely deleted.

Contents

DomainStatus

The current status of the search domain.

Type: [DomainStatus](#) (p. 200)

Required: No

DeleteIndexFieldResult

Description

A response message that contains the status of a deleted index field.

Contents

IndexField

The value of an `IndexField` and its current status.

Type: [IndexFieldStatus](#) (p. 203)

Required: Yes

DeleteRankExpressionResult

Description

A response message that contains the status of a deleted `RankExpression`.

Contents

RankExpression

The value of a `RankExpression` and its current status.

Type: [RankExpressionStatus](#) (p. 206)

Required: Yes

DescribeDefaultSearchFieldResult

Description

A response message that contains the default search field for a search domain.

Contents

DefaultSearchField

The name of the `IndexField` to use for search requests issued with the `q` parameter. The default is the empty string, which automatically searches all text fields.

Type: [DefaultSearchFieldStatus](#) (p. 196)

Required: Yes

DescribeDomainsResult

Description

A response message that contains the status of one or more domains.

Contents

DomainStatusList

The current status of all of your search domains.

Type: [DomainStatus](#) (p. 200) list

Required: Yes

DescribeIndexFieldsResult

Description

A response message that contains the index fields for a search domain.

Contents

IndexFields

The index fields configured for the domain.

Type: [IndexFieldStatus](#) (p. 203) list

Required: Yes

DescribeRankExpressionsResult

Description

A response message that contains the rank expressions for a search domain.

Contents

RankExpressions

The rank expressions configured for the domain.

Type: [RankExpressionStatus](#) (p. 206) list

Required: Yes

DescribeServiceAccessPoliciesResult

Description

A response message that contains the access policies for a domain.

Contents

AccessPolicies

A `PolicyDocument` that specifies access policies for the search domain's services, and the current status of those policies.

Type: [AccessPoliciesStatus](#) (p. 195)

Required: Yes

DescribeStemmingOptionsResult

Description

A response message that contains the stemming options for a search domain.

Contents

Stems

The stemming options configured for this search domain and the current status of those options.

Type: [StemmingOptionsStatus](#) (p. 209)

Required: Yes

DescribeStopwordOptionsResult

Description

A response message that contains the stopword options for a search domain.

Contents

Stopwords

The stopword options configured for this search domain and the current status of those options.

Type: [StopwordOptionsStatus](#) (p. 210)

Required: Yes

DescribeSynonymOptionsResult

Description

A response message that contains the synonym options for a search domain.

Contents

Synonyms

The synonym options configured for this search domain and the current status of those options.

Type: [SynonymOptionsStatus](#) (p. 210)

Required: Yes

DomainStatus

Description

The current status of the search domain.

Contents

Created

True if the search domain is created. It can take several minutes to initialize a domain when [CreateDomain](#) (p. 166) is called. Newly created search domains are returned from [DescribeDomains](#) (p. 177) with a false value for Created until domain creation is complete.

Type: Boolean

Required: No

Deleted

True if the search domain has been deleted. The system must clean up resources dedicated to the search domain when [DeleteDomain](#) (p. 171) is called. Newly deleted search domains are returned from [DescribeDomains](#) (p. 177) with a true value for IsDeleted for several minutes until resource cleanup is complete.

Type: Boolean

Required: No

DocService

The service endpoint for updating documents in a search domain.

Type: [ServiceEndpoint](#) (p. 207)

Required: No

DomainId

An internally generated unique identifier for a domain.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

DomainName

A string that represents the name of a domain. Domain names must be unique across the domains owned by an account within an AWS region. Domain names must start with a letter or number and can contain the following characters: a-z (lowercase), 0-9, and - (hyphen). Uppercase letters and underscores are not allowed.

Type: String

Length constraints: Minimum length of 3. Maximum length of 28.

Required: Yes

NumSearchableDocs

The number of documents that have been submitted to the domain and indexed.

Type: Integer

Required: No

Processing

True if processing is being done to activate the current domain configuration.

Type: Boolean

Required: No

RequiresIndexDocuments

True if [IndexDocuments](#) (p. 184) needs to be called to activate the current domain configuration.

Type: Boolean

Required: Yes

SearchInstanceCount

The number of search instances that are available to process search requests.

Type: Integer

Required: No

SearchInstanceType

The instance type (such as search.m1.small) that is being used to process search requests.

Type: String

Required: No

SearchPartitionCount

The number of partitions across which the search index is spread.

Type: Integer

Required: No

SearchService

The service endpoint for requesting search results from a search domain.

Type: [ServiceEndpoint](#) (p. 207)

Required: No

IndexDocumentsResult

Description

The result of an `IndexDocuments` action.

Contents

FieldNames

The names of the fields that are currently being processed due to an `IndexDocuments` action.

Type: String list

Required: No

IndexField

Description

Defines a field in the index, including its name, type, and the source of its data. The `IndexFieldType` indicates which of the options will be present. It is invalid to specify options for a type other than the `IndexFieldType`.

Contents

IndexFieldName

The name of a field in the search index. Field names must begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and _ (underscore). Uppercase letters and hyphens are not allowed. The names "body", "docid", and "text_relevance" are reserved and cannot be specified as field or rank expression names.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

IndexFieldType

The type of field. Based on this type, exactly one of the [UIntOptions](#) (p. 212), [LiteralOptions](#) (p. 204) or [TextOptions](#) (p. 211) must be present.

Type: String

Valid Values: `uint` | `literal` | `text`

Required: Yes

LiteralOptions

Options for literal field. Present if `IndexFieldType` specifies the field is of type `literal`.

Type: [LiteralOptions](#) (p. 204)

Required: No

SourceAttributes

An optional list of source attributes that provide data for this index field. If not specified, the data is pulled from a source attribute with the same name as this `IndexField`. When one or more source attributes are specified, an optional data transformation can be applied to the source data when populating the index field. You can configure a maximum of 20 sources for an `IndexField`.

Type: [SourceAttribute](#) (p. 207) list

Required: No

TextOptions

Options for text field. Present if `IndexFieldType` specifies the field is of type `text`.

Type: [TextOptions](#) (p. 211)

Required: No

UIntOptions

Options for an unsigned integer field. Present if `IndexFieldType` specifies the field is of type `unsigned integer`.

Type: [UIntOptions](#) (p. 212)

Required: No

IndexFieldStatus

Description

The value of an `IndexField` and its current status.

Contents

Options

Defines a field in the index, including its name, type, and the source of its data. The `IndexFieldType` indicates which of the options will be present. It is invalid to specify options for a type other than the `IndexFieldType`.

Type: [IndexField](#) (p. 202)

Required: Yes

Status

The status of an option, including when it was last updated and whether it is actively in use for searches.

Type: [OptionStatus](#) (p. 205)

Required: Yes

LiteralOptions

Description

Options that define a literal field in the search index.

Contents

DefaultValue

The default value for a literal field. Optional.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

FacetEnabled

Specifies whether facets are enabled for this field. Default: False.

Type: Boolean

Required: No

ResultEnabled

Specifies whether values of this field can be returned in search results and used for ranking. Default: False.

Type: Boolean

Required: No

SearchEnabled

Specifies whether search is enabled for this field. Default: False.

Type: Boolean

Required: No

NamedRankExpression

Description

A named expression that can be evaluated at search time and used for ranking or thresholding in a search query.

Contents

RankExpression

The expression to evaluate for ranking or thresholding while processing a search request. The `RankExpression` syntax is based on JavaScript expressions and supports:

- Integer, floating point, hex and octal literals
- Shortcut evaluation of logical operators such that an expression `a || b` evaluates to the value `a`, if `a` is true, without evaluating `b` at all
- JavaScript order of precedence for operators
- Arithmetic operators: `+` `-` `*` `/` `%`

- Boolean operators (including the ternary operator)
- Bitwise operators
- Comparison operators
- Common mathematic functions: `abs` `ceil` `erf` `exp` `floor` `lgamma` `ln` `log2` `log10` `max` `min` `sqrt` `pow`
- Trigonometric library functions: `acosh` `acos` `asinh` `asin` `atanh` `atan` `cosh` `cos` `sinh` `sin` `tanh` `tan`
- Random generation of a number between 0 and 1: `rand`
- Current time in epoch: `time`
- The `min` `max` functions that operate on a variable argument list

Intermediate results are calculated as double precision floating point values. The final return value of a `RankExpression` is automatically converted from floating point to a 32-bit unsigned integer by rounding to the nearest integer, with a natural floor of 0 and a ceiling of `max(uint32_t)`, 4294967295. Mathematical errors such as dividing by 0 will fail during evaluation and return a value of 0.

The source data for a `RankExpression` can be the name of an `IndexField` of type `uint`, another `RankExpression` or the reserved name `text_relevance`. The `text_relevance` source is defined to return an integer from 0 to 1000 (inclusive) to indicate how relevant a document is to the search request, taking into account repetition of search terms in the document and proximity of search terms to each other in each matching `IndexField` in the document.

For more information about using rank expressions to customize ranking, see the Amazon CloudSearch Developer Guide.

Type: String

Length constraints: Minimum length of 1. Maximum length of 10240.

Required: Yes

RankName

The name of a rank expression. Rank expression names must begin with a letter and can contain the following characters: a-z (lowercase), 0-9, and `_` (underscore). Uppercase letters and hyphens are not allowed. The names "body", "docid", and "text_relevance" are reserved and cannot be specified as field or rank expression names.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

OptionStatus

Description

The status of an option, including when it was last updated and whether it is actively in use for searches.

Contents

CreationDate

A timestamp for when this option was created.

Type: DateTime

Required: Yes

PendingDeletion

Indicates that the option will be deleted once processing is complete.

Type: Boolean

Required: No

State

The state of processing a change to an option. Possible values:

- `RequiresIndexDocuments`: the option's latest value will not be visible in searches until [IndexDocuments](#) (p. 184) has been called and indexing is complete.
- `Processing`: the option's latest value is not yet visible in all searches but is in the process of being activated.
- `Active`: the option's latest value is completely visible. Any warnings or messages generated during processing are provided in `Diagnostics`.

Type: String

Valid Values: `RequiresIndexDocuments` | `Processing` | `Active`

Required: Yes

UpdateDate

A timestamp for when this option was last updated.

Type: DateTime

Required: Yes

UpdateVersion

A unique integer that indicates when this option was last updated.

Type: Integer

Required: No

RankExpressionStatus

Description

The value of a `RankExpression` and its current status.

Contents

Options

The expression that is evaluated for ranking or thresholding while processing a search request.

Type: [NamedRankExpression](#) (p. 204)

Required: Yes

Status

The status of an option, including when it was last updated and whether it is actively in use for searches.

Type: [OptionStatus](#) (p. 205)

Required: Yes

ServiceEndpoint

Description

The endpoint to which service requests can be submitted, including the actual URL prefix for sending requests and the Amazon Resource Name (ARN) so the endpoint can be referenced in other API calls such as [UpdateServiceAccessPolicies](#) (p. 187).

Contents

Arn

An Amazon Resource Name (ARN). See [Identifiers for IAM Entities](#) in *Using AWS Identity and Access Management* for more information.

Type: String

Required: No

Endpoint

The URL (including /version/pathPrefix) to which service requests can be submitted.

Type: String

Required: No

SourceAttribute

Description

Identifies the source data for an index field. An optional data transformation can be applied to the source data when populating the index field. By default, the value of the source attribute is copied to the index field.

Contents

SourceDataCopy

Copies data from a source document attribute to an `IndexField`.

Type: [SourceData](#) (p. 208)

Required: No

SourceDataFunction

Identifies the transformation to apply when copying data from a source attribute.

Type: String

Valid Values: `Copy` | `TrimTitle` | `Map`

Required: Yes

SourceDataMap

Maps source document attribute values to new values when populating the `IndexField`.

Type: [SourceDataMap](#) (p. 208)

Required: No

SourceDataTrimTitle

Trims common title words from a source document attribute when populating an `IndexField`. This can be used to create an `IndexField` you can use for sorting.

Type: [SourceDataTrimTitle](#) (p. 209)

Required: No

SourceData

Description

The source attribute name and an optional default value to use if a document doesn't have an attribute of that name.

Contents

DefaultValue

The default value to use if the source attribute is not specified in a document. Optional.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

SourceName

The name of the document source field to add to this `IndexField`.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

SourceDataMap

Description

Specifies how to map source attribute values to custom values when populating an `IndexField`.

Contents

Cases

A map that translates source field values to custom values.

Type: String to String map

Required: No

DefaultValue

The default value to use if the source attribute is not specified in a document. Optional.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

SourceName

The name of the document source field to add to this `IndexField`.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

SourceDataTrimTitle

Description

Specifies how to trim common words from the beginning of a field to enable title sorting by that field.

Contents

DefaultValue

The default value to use if the source attribute is not specified in a document. Optional.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

Language

An [IETF RFC 4646](#) language code. Only the primary language is considered. English (en) is currently the only supported language.

Type: String

Required: No

Separator

The separator that follows the text to trim.

Type: String

Required: No

SourceName

The name of the document source field to add to this `IndexField`.

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: Yes

StemmingOptionsStatus

Description

The stemming options configured for this search domain and the current status of those options.

Contents

Options

Maps terms to their stems, serialized as a JSON document. The document has a single object with one property "stems" whose value is an object mapping terms to their stems. The maximum size of a stemming document is 500 KB. Example: { "stems": { "people": "person", "walking": "walk" } }

Type: String

Required: Yes

Status

The status of an option, including when it was last updated and whether it is actively in use for searches.

Type: [OptionStatus](#) (p. 205)

Required: Yes

StopwordOptionsStatus

Description

The stopword options configured for this search domain and the current status of those options.

Contents

Options

Lists stopwords serialized as a JSON document. The document has a single object with one property "stopwords" whose value is an array of strings. The maximum size of a stopwords document is 10 KB. Example: { "stopwords": ["a", "an", "the", "of"] }

Type: String

Required: Yes

Status

The status of an option, including when it was last updated and whether it is actively in use for searches.

Type: [OptionStatus](#) (p. 205)

Required: Yes

SynonymOptionsStatus

Description

The synonym options configured for this search domain and the current status of those options.

Contents

Options

Maps terms to their synonyms, serialized as a JSON document. The document has a single object with one property "synonyms" whose value is an object mapping terms to their synonyms. Each synonym is a simple string or an array of strings. The maximum size of a stopwords document is 100 KB. Example: { "synonyms": { "cat": ["feline", "kitten"], "puppy": "dog" } }

Type: String

Required: Yes

Status

The status of an option, including when it was last updated and whether it is actively in use for searches.

Type: [OptionStatus](#) (p. 205)

Required: Yes

TextOptions

Description

Options that define a text field in the search index.

Contents

DefaultValue

The default value for a text field. Optional.

Type: String

Length constraints: Minimum length of 0. Maximum length of 1024.

Required: No

FacetEnabled

Specifies whether facets are enabled for this field. Default: False.

Type: Boolean

Required: No

ResultEnabled

Specifies whether values of this field can be returned in search results and used for ranking. Default: False.

Type: Boolean

Required: No

TextProcessor

The text processor to apply to this field. Optional. Possible values:

- `cs_text_no_stemming`: turns off stemming for the field.

Default: none

Type: String

Length constraints: Minimum length of 1. Maximum length of 64.

Required: No

UIntOptions

Description

Options that define a `uint` field in the search index.

Contents

DefaultValue

The default value for an unsigned integer field. Optional.

Type: Integer

Required: No

UpdateDefaultSearchFieldResult

Description

A response message that contains the status of an updated default search field.

Contents

DefaultSearchField

The value of the `DefaultSearchField` configured for this search domain and its current status.

Type: [DefaultSearchFieldStatus](#) (p. 196)

Required: Yes

UpdateServiceAccessPoliciesResult

Description

A response message that contains the status of updated access policies.

Contents

AccessPolicies

A `PolicyDocument` that specifies access policies for the search domain's services, and the current status of those policies.

Type: [AccessPoliciesStatus](#) (p. 195)

Required: Yes

UpdateStemmingOptionsResult

Description

A response message that contains the status of updated stemming options.

Contents

Stems

The stemming options configured for this search domain and the current status of those options.

Type: [StemmingOptionsStatus](#) (p. 209)

Required: Yes

UpdateStopwordOptionsResult

Description

A response message that contains the status of updated stopword options.

Contents

Stopwords

The stopword options configured for this search domain and the current status of those options.

Type: [StopwordOptionsStatus](#) (p. 210)

Required: Yes

UpdateSynonymOptionsResult

Description

A response message that contains the status of updated synonym options.

Contents

Synonyms

The synonym options configured for this search domain and the current status of those options.

Type: [SynonymOptionsStatus](#) (p. 210)

Required: Yes

Common Parameters

This section lists the request parameters that all actions use. Any action-specific parameters are listed in the topic for the action.

Action

The action to be performed.

Default: None

Type: string

Required: Yes

AuthParams

The parameters that are required to authenticate a Conditional request. Contains:

- `AWSSessionToken`
- `SignatureVersion`
- `Timestamp`
- `Signature`

Default: None

Required: Conditional

AWSSessionToken

The access key ID that corresponds to the secret access key that you used to sign the request.

Default: None

Type: string

Required: Yes

Expires

The date and time when the request signature expires, expressed in the format `YYYY-MM-DDThh:mm:ssZ`, as specified in the ISO 8601 standard.

Condition: Requests must include either *Timestamp* or *Expires*, but not both.

Default: None

Type: string

Required: Conditional

SecurityToken

The temporary security token that was obtained through a call to AWS Security Token Service. For a list of services that support AWS Security Token Service, go to [Using Temporary Security Credentials to Access AWS](#) in **Using Temporary Security Credentials**.

Default: None

Type: string

Required: No

Signature

The digital signature that you created for the request. For information about generating a signature, go to the service's developer documentation.

Default: None

Type: string

Required: Yes

SignatureMethod

The hash algorithm that you used to create the request signature.

Default: None

Type: string

Valid Values: HmacSHA256 | HmacSHA1

Required: Yes

SignatureVersion

The signature version you use to sign the request. Set this to the value that is recommended for your service.

Default: None

Type: string

Required: Yes

Timestamp

The date and time when the request was signed, expressed in the format YYYY-MM-DDThh:mm:ssZ, as specified in the ISO 8601 standard.

Condition: Requests must include either *Timestamp* or *Expires*, but not both.

Default: None

Type: string

Required: Conditional

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Default: None

Type: string

Required: Yes

Common Errors

This section lists the common errors that all actions return. Any action-specific errors are listed in the topic for the action.

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

AWS query string is malformed, 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

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

Throttling

Request was denied due to request throttling.

HTTP Status Code: 400

Amazon CloudSearch Document Service API Reference

Topics

- [documents/batch](#) (p. 217)

You use the document service API to manage the documents in your Amazon CloudSearch domain. The documents in your domain are automatically indexed and made searchable. You access the document service API through a domain-specific endpoint, `http://doc-domainname-domainid.us-east-1.cloudsearch.amazonaws.com`.

You use the Amazon CloudSearch document service API to:

- Add new documents to your search domain
- Replace existing documents in your search domain
- Remove existing documents from your search domain

Note

The document service API is a REST-style API that has a single resource, `documents/batch`. The API version must be specified in all requests. The current Amazon CloudSearch API version is 2011-02-01.

The other APIs you use to interact with Amazon CloudSearch are:

- [Amazon CloudSearch Configuration API Reference](#) (p. 164)—Set up and manage your search domain.
- [Amazon CloudSearch Search API Reference](#) (p. 227)—Search your domain.

documents/batch

This section describes the HTTP request and response messages for the `documents/batch` resource. You use the `documents/batch` resource to upload data to your search domain for indexing. It is accessed through a domain's document service endpoint at `/2011-02-01/documents/batch`. All requests must be submitted using HTTP POST.

Requests can only be submitted to your search domain's document service from authorized IP addresses. For information about authorizing IP addresses to submit document service requests, see [Configuring Access for an Amazon CloudSearch Domain](#) (p. 34).

The data you upload must be formatted in the Search Data Format (SDF), which can be encoded as either JSON or XML. SDF data provides all of the information Amazon CloudSearch needs for indexing. Each item that you want to be able to return as a search result (such as a product) is represented as a document in an SDF batch. An SDF batch is simply a collection of add and delete requests for individual documents. Every document has a unique ID, a version number, and one or more fields that contain the data that you want to search and return in results.

To update a document, you specify an add request with the document ID of the document you want to update and a higher version number. The version number in the add request must be greater than the current version number for the update to be applied. For more information, see [Adding and Updating Documents in Amazon CloudSearch](#) (p. 54). Similarly, to delete a document, you submit a delete request with the document ID of the document you want to delete and a higher version number. For a document to be deleted, the version number must be greater than the current version number. For information about deleting documents, see [Deleting Documents in Amazon CloudSearch](#) (p. 54).

For more information about submitting data for indexing, see [Uploading Data to an Amazon CloudSearch Domain](#) (p. 77).

documents/batch JSON API

JSON documents/batch Requests

The body of a `documents/batch` request uses [SDF](#) to specify the document operations you want to perform. An SDF JSON representation of a batch is a collection of objects that define individual add and delete operations. The `type` property identifies whether an object represents an add or delete operation. For example, the following JSON SDF batch adds one document and deletes one document:

```
[
  {
    "type": "add",
    "id": "tt0484562",
    "version": 1,
    "lang": "en",
    "fields": {
      "title": "The Seeker: The Dark Is Rising",
      "director": "Cunningham, David L.",
      "genre": ["Adventure", "Drama", "Fantasy", "Thriller"],
      "actor": ["McShane, Ian", "Eccleston, Christopher", "Conroy, Frances",
        "Crewson, Wendy", "Ludwig, Alexander", "Cosmo, James",
        "Warner, Amelia", "Hickey, John Benjamin", "Piddock, Jim",
        "Lockhart, Emma"]
    }
  },
  {
    "type": "delete",
    "id": "tt0484575",
    "version": 2
  }
]
```

Note

When specifying SDF in JSON, the value for a field cannot be `null`.

An add or delete operation is only applied to an existing document if the version number specified in the operation is greater than the existing document's version number. If a batch contains multiple add or delete operations for the same document, the operation with the highest version number is applied. (If

multiple operations in a batch specify the same document and version number, the document service arbitrarily picks which one to apply.)

The [JSON schema](#) representation of a batch is shown below:

```
{
  "type": "array",
  "minItems": 1,
  "items": {
    "type": "object",
    "properties": {
      "type": {
        "type": "string",
        "enum": ["add", "delete"],
        "required": true
      },
      "id": {
        "type": "string",
        "pattern": "[a-z0-9][a-z0-9_]{0,127}",
        "minLength": 1,
        "maxLength": 128,
        "required": true
      },
      "version": {
        "type": "number",
        "minimum": 1,
        "maximum": 4294967295
        "required": true
      },
      "lang": {
        "type": "string",
        "minLength": 2,
        "maxLength": 2
      },
      "fields": {
        "type": "object",
        "patternProperties": {
          "[a-zA-Z0-9][a-zA-Z0-9_]{0,63}": {
            "type": "string",
          }
        }
      }
    }
  }
}
```

documents/batch Request Properties (JSON)

Property	Description	Required
type	The operation type, add or delete.	Yes
id	An alphanumeric string. Allowed characters are: a-z (lower-case letters), 0-9, and _ (underscore). Document IDs cannot begin with an underscore. The max length is 128 characters.	Yes
version	Any non-negative number less than 2 ³² .	Yes

Property	Description	Required
lang	An ISO-639-1 two-letter language code. English (en) is currently the only supported language. Condition: Required for add operations.	Conditional
fields	A collection of one or more <i>field_name</i> properties that define the fields the document contains. Condition: Required for add operations. Must contain at least one <i>field_name</i> property.	Conditional
<i>field_name</i>	Specifies a field within the document being added. Field names must begin with a letter and can contain the following characters: a-z (lower case), 0-9, and _ (underscore). Field names must be at least 3 and no more than 64 characters. The names "body", "docid", and "text_relevance" are reserved names and cannot be used as field names. To specify multiple values for a field, you specify an array of values instead of a single value. For example: <pre>"genre": ["Adventure", "Drama", "Fantasy", "Thriller"]</pre>	Conditional
	Condition: At least one field must be specified in the fields object.	

documents/batch Response (JSON)

The response body lists the number of adds and deletes that were performed and any errors or warnings that were generated.

The JSON schema representation of a document service API response is shown below:

```
{
  "type": "object",
  "properties": {
    "status": {
      "type": "text",
      "enum": ["success", "error"],
      "required": true
    },
    "adds": {
      "type": "integer",
      "minimum": 0,
      "required": true
    },
    "deletes": {
      "type": "integer",
      "minimum": 0,
      "required": true
    }
  }
}
```

```

    "errors": {
      "type": "array",
      "required": false,
      "items": {
        "type": "object",
        "properties": {
          "message": {
            "type": "string",
            "required": true
          }
        }
      }
    },
    "warnings": {
      "type": "array",
      "required": false,
      "items": {
        "type": "object",
        "properties": {
          "message": {
            "type": "string",
            "required": true
          }
        }
      }
    }
  }
}

```

documents/batch Response Properties (JSON)

Property	Description
status	The result status, which is either <code>success</code> or <code>error</code> .
adds	The number of add document operations that were performed. Always zero when the status is <code>error</code> .
deletes	The number of delete document operations that were performed. Always zero when the status is <code>error</code> .
error	Provides information about a parsing or validation error. Specified only if the status is <code>error</code> .
warning	Provides information about a warning generated during parsing or validation.

documents/batch XML API

XML documents/batch Requests

The body of a `documents/batch` request uses the Search Data Format (SDF) to specify the document operations you want to perform. For example:

```

<batch>
  <add id="tt0484562" version="1" lang="en">

```

```
<field name="title">The Seeker: The Dark Is Rising</field>
<field name="director">Cunningham, David L.</field>
<field name="genre">Adventure</field>
<field name="genre">Drama</field>
<field name="genre">Fantasy</field>
<field name="genre">Thriller</field>
<field name="actor">McShane, Ian</field>
<field name="actor">Eccleston, Christopher</field>
<field name="actor">Conroy, Frances</field>
<field name="actor">Ludwig, Alexander</field>
<field name="actor">Crewson, Wendy</field>
<field name="actor">Warner, Amelia</field>
<field name="actor">Cosmo, James</field>
<field name="actor">Hickey, John Benjamin</field>
<field name="actor">Pidcock, Jim</field>
<field name="actor">Lockhart, Emma</field>
</add>
<delete id="tt0301199" version="1" />
</batch>
```

The [Relax NG](#) schema for an XML representation of a batch is shown below:

```
start = batch

sdf.field_name = xsd:token {
  minLength = "1"
  maxLength = "64"
  pattern = "[a-z0-9][a-z0-9_]{0,63}"
}

sdf.id = xsd:token {
  minLength = "1"
  maxLength = "128"
  pattern = "[a-z0-9][a-z0-9_]{0,127}"
}

sdf.version = xsd:integer {
  minExclusive = "0"
  maxExclusive = "4294967295"
}

batch = element batch {
  (element add {
    attribute id { sdf.id },
    attribute version { sdf.version },
    attribute lang { xsd:language },
    element field {
      attribute name { sdf.field_name },
      text
    }+
  }
  | element delete {
    attribute id { sdf.id },
    attribute version { sdf.version },
    empty
  }+
  )+
}
```

documents/batch Request Elements (XML)

Element	Description	Required
batch	The collection of add or delete operations that you want to submit to your search domain. A batch must contain at least one add or delete element.	Yes
add	<p>Specifies a document that you want to add to your search domain. The id, version, and lang attributes are required and an add element must contain at least one field.</p> <p>Attributes:</p> <ul style="list-style-type: none"> id—An alphanumeric string. Any characters other than A-Z (upper or lower case) and 0-9 are illegal. The max length is 128 characters. version—Any non-negative number less than 2³². lang—An ISO-639-1 two-letter language code. English (en) is currently the only supported language. 	No
field	<p>Specifies a field in the document being added. The name attribute and a field value are required. Field names must begin with a letter and can contain the following characters: a-z (lower case), 0-9, and _ (underscore). The names "body", "docid", and "text_relevance" are reserved names and cannot be used as field names. The field value can be text or CDATA.</p> <p>To specify multiple values for a field, you include multiple field elements with the same name. For example:</p> <pre><field name="genre">Adventure</field> <field name="genre">Drama</field> <field name="genre">Fantasy</field> <field name="genre">Thriller</field></pre> <p>Constraints:</p> <ul style="list-style-type: none"> name—An alphanumeric string that begins with a letter. Can contain a-z (lower case), 0-9, _ (underscore), - (hyphen), and . (period). <p>Condition: At least one field must be specified in an add element.</p>	Conditional

Element	Description	Required
delete	<p>Specifies a document that you want to remove from your search domain. The id and version attributes are required. A delete element must be empty.</p> <p>Constraints:</p> <ul style="list-style-type: none">• <code>id</code>—An alphanumeric string. Any characters other than A-Z (upper or lower case) and 0-9 are illegal.• <code>version</code>—Any number less than 2³². The version number specified must be higher than the document's current version number for the document to be deleted.	No

documents/batch Response (XML)

The response body lists the number of adds and deletes that were performed and any errors or warnings that were generated.

The RelaxNG schema of a document service API response is:

```
start = response

response = element response {
  attribute status { "success" | "error" },
  attribute adds { xsd:integer },
  attribute deletes { xsd:integer },
  element errors {
    element error {
      text
    }+
  }? &
  element warnings {
    element warning {
      text
    }+
  }?
}
```

documents/batch Response Elements (XML)

Element	Description
result	<p>Contains elements that list the errors and warnings generated when parsing and validating the request.</p> <p>Attributes:</p> <ul style="list-style-type: none"> <code>status</code>—The result status, which is either <code>success</code> or <code>error</code>. <code>adds</code>—The number of added documents. If the status is <code>error</code>, this is always zero. <code>deletes</code>—The number of deleted documents. If the status is <code>error</code>, this is always zero. <p>Constraints: If the status is <code>error</code>, the results element contains a list of errors. If the status is <code>success</code>, the results element can contain a list of warnings, but no errors.</p>
errors	Contains a collection of error elements that identify the errors that occurred when parsing and validating the request.
error	Provides information about a parsing or validation error. The value provides a description of the error.
warnings	Contains a collection of warning elements that identify the warnings that were generated when parsing and validating the request.
warning	Provides information about a parsing or validation warning. The value provides a description of the error.

documents/batch Status Codes

A document service request can return three types of status codes:

- 5xx status codes indicate that there was an internal server error. We recommend catching and retrying all 5xx error codes as they typically represent transient error conditions.
- 4xx status codes indicate that the request was malformed.
- 2xx status codes indicate that the request was processed successfully.

Error	Description	HTTP Status Code
No Content-Type	The Content-Type header is missing.	400
No Content-Length	The Content-Length header is missing.	411
Incorrect Path	URL path does not match " <code>/YYYY-MM-DD/documents/batch</code> ".	404
Invalid HTTP Method	The HTTP method is not POST. Requests must be posted to <code>documents/batch</code> .	405

Error	Description	HTTP Status Code
Invalid Accept Type	Accept header specifies a content type other than "application/xml" or "application/json". Responses can be sent only as XML or JSON.	406
Request Too Large	The length of the request body is larger than the maximum allowed value.	413
Invalid Content Type	The content type is something other than "application/json" or "application/xml".	415
Invalid Character Set	The character set is something other than "ASCII", "ISO-8859-1", or "UTF-8".	415

Common Request Headers

Name	Description	Required
Content-Type	A standard MIME type describing the format of the object data. For more information, see W3C RFC 2616 Section 14 . Default: application/json Constraints: application/json or application/xml only	Required
Content-Length	The length in bytes of the body of the request.	Yes
Accept	A standard MIME type describing the format of the response data. For more information, see W3C RFC 2616 Section 14 . Default: the content-type of the request Constraints: application/json or application/xml only	No

CommonResponse Headers

Name	Description
Content-Type	A standard MIME type describing the format of the object data. For more information, see W3C RFC 2616 Section 14 . Default: the value of the Accept header in the request, or the Content-Type of the request if the Accept header is missing or doesn't specify either application/xml or application/json. Constraints: application/xml or application/json only
Content-Length	The length in bytes of the body in the response.

Amazon CloudSearch Search API Reference

Topics

- [search \(p. 227\)](#)

You use the Search API to submit search requests to your Amazon CloudSearch domain. You access the Search API through a domain-specific endpoint, `http://search-domainname-domainid.us-east-1.cloudsearch.amazonaws.com`. The API version must be specified in all requests. The current Amazon CloudSearch API version is 2011-02-01.

The other APIs you use to interact with Amazon CloudSearch are:

- [Amazon CloudSearch Configuration API Reference \(p. 164\)](#)—Set up and manage your search domain.
- [Amazon CloudSearch Document Service API Reference \(p. 217\)](#)—Submit the data you want to search.

search

You use the `search` API to search the documents that you've uploaded to your search domain.

Search requests are submitted via GET with a set of field-value pairs specified directly in the HTTP query string. The maximum size of a search request is 8190 bytes, including the HTTP method, URI, and protocol version. The response format can be either JSON or XML. Setting the response format only affects responses to successful requests. The format of an error response depends on the origin of the error. Errors returned by the search service are always returned in JSON. 5xx errors due to server timeouts and other request routing problems are returned in XML.

Note

Requests can be submitted to your search domain's search service only from authorized IP addresses. For information about authorizing IP addresses to submit search requests, see [Configuring Access for an Amazon CloudSearch Domain \(p. 34\)](#).

Amazon CloudSearch processes search requests in two phases. First, it identifies the complete set of documents that match the terms specified with the `q` (query) and `bq` (Boolean query) [Search Request Parameters \(p. 229\)](#). Amazon CloudSearch then processes the match-set of search hits to:

- Filter the hits according to the value of the `t-FIELD` parameter (if specified).

- Rank the filtered hits using the fields specified in the `rank` parameter. If the rank parameter is not specified, results are ranked according to their [text_relevance](#) scores.
- Compute facet counts for the fields specified in the `facet` parameter and the constraints specified for each field (if any).
- Return the processed set of hits. The maximum number of hits returned is controlled by the `size` parameter. By default, the top ten results are returned. You can specify an offset with the `start` parameter to retrieve the next set of hits.

For more information about searching with Amazon CloudSearch, see [Searching Your Data with Amazon CloudSearch \(p. 85\)](#).

Search Requests

You submit search requests to your domain's search endpoint via HTTP GET. To construct a search request, you append the Amazon CloudSearch API version and the name of the resource you are accessing, `2011-02-01/search`, and a query string that specifies the terms and constraints for your search and what you want to get back in the response. The maximum size of a search request is 8190 bytes, including the HTTP method, URI, and protocol version.

For example, the following request performs a simple text search of the `search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.amazonaws.com` domain and gets the contents of the title field:

```
http://search-movies-rr2f34ofg56xneuemujamut52i.us-east-1.cloudsearch.
amazonaws.com/2011-02-01/search?q=star+wars&return-fields=title
```

Note

The API version must be specified in all search requests. When there are updates to the Amazon CloudSearch Search API, you access them using a new API version.

The query string in a search request must be URL-encoded. You can use any method you want to send GET requests to your domain's search endpoint—you can enter the request URL directly in a Web browser, use cURL to submit the request, or generate an HTTP call using your favorite HTTP library.

By default, Amazon CloudSearch returns the response in JSON. You can also get the results formatted in XML by specifying the `results-type` parameter, `results-type=xml`.

Note

You can also use the Search Tester in the Amazon CloudSearch console to search your data, browse the results, and view the generated request URLs and JSON and XML responses. For more information, see [Searching with the Search Tester \(p. 16\)](#).

Amazon CloudSearch can return up to 2 KB of data from a text field—if the contents of the field exceed 2 KB, only the first 2 KB is included in the results. (All of the data is searchable, only the result data is truncated.)

Search Syntax

```
GET /2011-02-01/search
```

Search Request Headers

Name	Description	Required
Cache-Control	Forces revalidation of results when a cached result document would otherwise be returned.	No
HOST	The search request endpoint for the domain you're querying. You can use DescribeDomains (p. 177) to retrieve your domain's search request endpoint.	Yes

Search Request Parameters

Name	Description	Required
bq	<p>One or more match expressions (p. 233) that define a Boolean search. Multiple expressions are joined with a top-level AND. If the bq parameter is specified in conjunction with the q parameter, the values are joined with a top-level AND.</p> <p>Within a match expression, you can use the - (NOT), (OR), and * (wildcard) operators to exclude particular terms, find results that match any of the specified terms, or search for a prefix. To search for a phrase rather than individual terms, you can enclose the phrase in double quotes. For more information, see Searching Your Data with Amazon CloudSearch (p. 85).</p> <p>Condition: Required if the q parameter is not specified.</p> <p>Type: String</p>	Conditional
facet	<p>A comma-separated list of the fields for which you want to compute facets. The specified fields must be numeric fields or defined as facet enabled in the domain configuration.</p> <p>By default, counts are computed for all field values. If you want to specify the field values that you want counted for a particular field, use the facet-FIELD-constraints parameter instead, where FIELD is the name of the field.</p> <p>You can specify the maximum number of constraints to include in the results with the facet-FIELD-top-n parameter. By default, the results include counts for the top 40 constraints.</p> <p>Type: String</p>	No

Name	Description	Required
<code>facet-FIELD-constraints</code>	<p>The field values (facet constraints) that you want to count for a particular field. <code>FIELD</code> is the name of the field. Constraints are specified as a comma-separated list of ranges or single-quoted strings. For example, <code>facet-year-constraints=2000..2011</code> calculates facet counts for the years 2000 through 2011, inclusive. You can omit the lower end of a range to count all of the values less than or equal to the specified value. Similarly, you can omit the upper end of a range to count all of the values greater than or equal to the specified value. To specify constraints for a text field, enclose the values in single quotes. For example, <code>facet-color-constraints='red','blue','green'</code>. If you don't specify facet constraints, counts are computed for all field values.</p> <p>Type: String</p>	No
<code>facet-FIELD-sort</code>	<p>How you want to sort facet values for a particular field. <code>FIELD</code> is the name of the field. There are four sorting options:</p> <ul style="list-style-type: none"> • <code>alpha</code>—Sort the facet values alphabetically (in ascending order). • <code>count</code>—Sort the facet values by their counts (in descending order). • <code>max</code>—Sort the facet values according to the maximum values in the specified field. This option is specified as <code>max(FIELD)</code>. By default, the facet values are sorted in ascending order. To sort in descending order, prefix the sort option with - (minus): <code>-max(FIELD)</code>. • <code>sum</code>—Sort the facet values according to the sum of the values in the specified field (in ascending order). This option is specified as <code>sum(FIELD)</code>. <p>Type: String</p>	No
<code>facet-FIELD-top-n</code>	<p>Set the maximum number of facet constraints to be included for the specified field in the search results. By default, the results include counts for the top 40 constraints.</p> <p>Type: Integer</p>	No

Amazon CloudSearch Developer Guide
Search Requests

Name	Description	Required
q	<p>The string to search for. You use the <code>q</code> parameter to perform simple text searches. This searches the default search field for the specified text. If the <code>q</code> parameter is specified in conjunction with the <code>bq</code> parameter, the values are joined with a top-level <code>AND</code>.</p> <p>If you separate search terms with plus (+) or a space, Amazon CloudSearch matches documents that contain all of the specified search terms—they are ANDed together. For example, <code>q=star+wars</code> searches the default field for star and wars. This is equivalent to specifying <code>bq='star wars'</code>.</p> <p>You can use the - (NOT), (OR), and * (wildcard) operators to exclude particular terms, find results that match any of the specified terms, or search for a prefix. To search for a phrase rather than individual terms, you can enclose the phrase in double quotes. For more information, see Searching Your Data with Amazon CloudSearch (p. 85).</p> <p>Condition: Required if the <code>bq</code> parameter is not specified.</p> <p>Type: String</p>	Conditional
rank	<p>A comma-separated list of fields or rank expressions to use for ranking. A maximum of 10 fields and rank expressions can be specified. You can use any uint field to rank results numerically. Any result-enabled text or literal field can be used to rank results alphabetically. To rank results by relevance, you can specify the name of a custom rank expression or <code>text_relevance</code>. Hits are ordered according to the specified rank field(s). By default, hits are ranked in ascending order. You can prefix a field name with a minus (-) to rank in descending order. If no rank parameter is specified, it defaults to <code>rank=-text_relevance</code>, which lists results according to their <code>text_relevance</code> scores with the highest-scoring documents first.</p> <p>Type: String</p>	No

Name	Description	Required
rank-RANKNAME	<p>Define a rank expression that can be used with the <code>rank</code> parameter. You can also specify the new rank expression as a return field and use it to set thresholds for the search results with the <code>t-FIELD</code> parameter. For more information about constructing rank expressions, see Customizing Result Ranking with Amazon CloudSearch (p. 105).</p> <p>Specifying a rank expression as part of the search request enables you to quickly test and tune new rank expressions. Once you have finished tuning your rank expression, you should configure it for use in all requests with the cs-configure-ranking (p. 108) command, from the Amazon CloudSearch console (p. 108), or using the DefineRankExpression (p. 109) configuration action.</p> <p>You can define and use multiple rank expressions in a search request. For example, the following request creates two rank expressions that are used to rank the results and returns one of them in the search results:</p> <pre style="border: 1px solid black; padding: 5px; margin: 10px 0;">search?q=terminator &rank-expression1=sin(text_relevance) &rank-expression2=cos(text_relevance) &rank=expression1,expression2 &return-fields=title,text_relevance,expression2</pre> <p>Type: String</p>	No
results-type	<p>Controls the content type of the response, <code>json</code> or <code>xml</code>. The default is <code>json</code>.</p> <p>Type: String</p>	No
return-fields	<p>The document fields to include in the response. Up to 2 KB of data can be returned from a text field. If the field contents exceed 2 KB, only the first 2 KB is included in the results. Specified as a comma-separated list of field names. If no <code>return-fields</code> are specified, only the document ids of the hits are returned.</p> <p>Type: String</p>	No
size	<p>The maximum number of search hits to return. The default is 10.</p> <p>Type: Positive Integer</p>	No
start	<p>The offset of the first search hit you want to return. The default is 0 (the first hit).</p> <p>Type: Positive Integer</p>	No

Name	Description	Required
t-FIELD	<p>Restrict the match set used in subsequent post-processing steps according to the specified rank expression. Only hits that have a score within the specified RANGE are included. Ranges are specified as described in Expression Syntax for Boolean Queries (p. 233).</p> <p>Type: RANGE</p>	No

Expression Syntax for Boolean Queries

Expression Syntax	Description
FIELD:'search string'	Search for a string in the specified text or literal field. For example, <code>bq=title:'star'</code> . Any single quotation marks or backslashes in the string must be escaped with a backslash.
(field FIELD 'search string')	Search for a string in the specified text or literal field. For example, <code>(field title 'star')</code> . Any single quotation marks or backslashes in the string must be escaped with a backslash. You can use this alternate fielded search syntax when you're specifying multiple fielded search expressions as part of Boolean expression. For example, <code>bq=(and (field title 'star') (filter year ..2000))</code> .
FIELD:value	Search for an integer value in the specified uint field. For example, <code>bq=year:2000</code> . Matches documents that have at least one value in the field that equals the specified value. You can specify a single value or a range of values. A pair of nonnegative integers separated by two dots matches documents that have at least one attribute in the field that falls in the specified range. You can omit one value to specify an open-ended upper or lower limit. The range is inclusive on both ends. For example, <code>bq=year:1998..2000</code> .

Expression Syntax	Description
<code>(filter FIELD value)</code>	Search for an integer in the specified uint field. For example, <code>(filter year 2000)</code> . Matches documents that have at least one value in the field that equals the specified value. You can use this alternate fielded search syntax when you're specifying multiple fielded search expressions as part of Boolean expression. For example, <code>bq=(and (field title 'star') (filter year ..2000))</code> . You can specify a single value or a range of values. A pair of nonnegative integers separated by two dots matches documents that have at least one attribute in the field that falls in the specified range. You can omit one value to specify an open-ended upper or lower limit. The range is inclusive on both ends. For example, <code>(filter year 1998..2000)</code> .
<code>(and expression1 expression2 expressionN)</code>	Include hits only if they match all of the specified expressions. (Boolean AND operator.) For example, <code>bq=(and (field title 'star') (field actor 'Ford, Harrison') (filter year ..2000))</code> .
<code>(not expression1)</code>	Exclude hits that match the specified expression. (Boolean NOT operator.) For example, <code>bq=(not (and (field actor 'Guinness, Alec') (field actor 'Ford, Harrison')))</code> .
<code>(or expression1 expression2 expressionN)</code>	Include hits that match any of the specified expressions. (Boolean OR operator.) For example, <code>bq=(or (field actor 'Guinness, Alec') (field actor 'Ford, Harrison') (field actor 'Jones, James Earl'))</code> .

Search Response

When a request completes successfully, the response body contains the search results. By default, search results are returned in JSON. If the `results-type` parameter is set to XML, search results are returned in XML.

Setting the response format only affects responses to successful requests. The format of an error response depends on the origin of the error. Errors returned by the search service are always returned in JSON. 5xx errors due to server timeouts and other request routing problems are returned in XML. When a request returns an error code, the body of the response contains information about the error that occurred. If an error occurs while the request body is parsed and validated, the error code is set to 400 and the response body includes a list of the errors and where they occurred.

The following example shows a JSON response.

```
{
  "rank": "-text_relevance",
  "match-expr": "(label 'star wars')",
  "hits": {
    "found": 7,
```



```
"start":0,
"hit":[
  {"id":"tt1185834",
   "data":{"actor":["Abercrombie, Ian","Baker, Dee","Burton, Corey"],
            "title":["Star Wars: The Clone Wars"]}
  },
  .
  .
  .
  {"id":"tt0121766",
   "data":{"actor":["Bai, Ling","Bryant, Gene","Castle-Hughes, Keisha"],
            "title":["Star Wars: Episode III - Revenge of the Sith"]}
  }
]
},
"info":{"rid":"b7c167f6c2da6d93531b9a7b314ad030b3a74803b4b7797edb905ba5a6a08",
        "time-ms":2,
        "cpu-time-ms":0
      }
}
```

The following example shows the equivalent XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<results xmlns="http://cloudsearch.amazonaws.com/2011-02-01/results">
  <rank>-text_relevance</rank>
  <match-expr>(label 'star wars')</match-expr>
  <hits found="7" start="0">
    <hit id="tt1185834">
      <d name="actor">Abercrombie, Ian</d>
      <d name="actor">Baker, Dee</d>
      <d name="actor">Burton, Corey</d>
      <d name="title">Star Wars</d>
    </hit>
    .
    .
    .
    <hit id="tt0121766">
      <d name="actor">Bai, Ling</d>
      <d name="actor">Bryant, Gene</d>
      <d name="actor">Castle-Hughes, Keisha</d>
      <d name="title">Star Wars: Episode III - Revenge of the Sith</d>
    </hit>
  </hits>
  <facets/>
  <info
    rid="b7c167f6c2da6d93531b9a7b314ad030a5ddfe34efbdd8959999ac792f37a1f"
    time-ms="2"
    cpu-time-ms="0"
  />
</results>
```

Search Response Headers

Name	Description
Content-Type	A standard MIME type describing the format of the object data. For more information, see W3C RFC 2616 Section 14 . Default: application/json Constraints: application/json or application/xml only
Content-Length	The length in bytes of the body in the response.

Search Response Properties (JSON)

Property	Description
match-expr	Shows the match expression constructed from the search parameters.
hits	Contains hit statistics (found, start) and a hit array that lists the document ids and data for each hit.
found	The total number of hits that match the search request after Amazon CloudSearch finished processing the match set.
start	The index of the first hit returned in this response.
hit	An array that lists the document ids and data for each hit.
id	The unique identifier for a document.
data	A list of returned fields.
facets	Contains facet information and facet counts.
FacetFieldName	A field for which facets were calculated.
constraints	An array of the facet values and counts.
value	The facet value being counted.
count	The number of hits that contain the facet value in FacetFieldName.
info	Contains information about the request processing.
rank	Lists the fields that were used to rank the search hits.
rid	The encrypted Resource ID.
time-ms	How long it took to process the search request in milliseconds.
cpu-time-ms	The CPU time required to process the search request in milliseconds.
messages	Contains any error messages returned by the search service. The severity, code, and message properties are included for each item.

Property	Description
severity	The severity of the message: warning, error, or fatal. Warning indicates a problem with the query string that did not prevent the request from being processed. Error indicates that an internal error occurred while processing the request. Fatal indicates a problem with the query string that prevented the request from being processed.
code	<p>The error code. The search service returns the following error codes:</p> <ul style="list-style-type: none"> • CS-InvalidFieldOrRankAliasInRankParameter—the specified ranking field could not be found. (Warning) • CS-RankExpressionParseError—one of the specified rank expressions could not be parsed. No query-time rank expressions will be used. (Warning) • CS-RankExpressionValidationError—one of the specified rank expressions could not be validated. No query-time rank expressions will be used. (Warning) • CS-UndefinedField—an unknown field was specified in the match expression. (Warning) • CS-UnknownFieldInMatchExpression—a field specified in the <code>bg</code> parameter could not be found. (Warning) • CS-WildcardTermLimit—more than 2000 terms matched the wildcard in the search request. The number of terms matched was limited to 2000. (Warning) • CS-IncorrectFieldTypeInMatchExpression—the type specified in the match expression does not match the field type. (Fatal) • CS-InvalidMatchSetExpression—the match expression could not be parsed. (Fatal) • CS-UnknownParameter—one of the specified parameters is not allowed. (Fatal)
message	A description of the error that was returned by the search service.

Search Response Elements (XML)

Name	Description
results	Contains the search results. Any errors that occurred while processing the request are returned as messages in the info element.
rank	Lists the fields that were used to rank the search hits.
match-expr	Shows the match expression constructed from the search parameters.
hits	Contains hit statistics and a collection of hit elements. The found attribute is the total number of hits that match the search request after Amazon CloudSearch finished processing the results. The contained hit elements are ordered according to their <code>text_relevance</code> scores or the rank option specified in the search request.
hit	A document that matched the search request. The id attribute is the document's unique id. Contains a <code>d</code> (data) element for each returned field.

Name	Description
<code>d</code>	A field returned from a hit. Hit elements contain a <code>d</code> (data) element for each returned field.
<code>facets</code>	Contains a facet element for each facet requested in the search request.
<code>facet</code>	Contains a constraint element for each value of a field for which a facet count was calculated. The <code>facet-FIELD-top-n</code> request parameter can be used to specify how many constraints to return. By default, facet counts are returned for the top 40 constraints. The <code>facet-FIELD-constraints</code> request parameter can be used to explicitly specify which values to count.
<code>constraint</code>	A facet field value and the number of occurrences (count) of that value within the search hits.
<code>info</code>	Information about the request processing. The <code>rid</code> attribute is the encrypted Resource ID. The <code>time-ms</code> attribute is how long it took to process the search request, in milliseconds. The <code>cpu-time-ms</code> attribute is the CPU time required to process the search request, in milliseconds.
<code>message</code>	<p>Information about an error returned by the search service while processing the request. The <code>severity</code> attribute can be: warning, error, or fatal. Warning indicates a problem with the query string that did not prevent the request from being processed. Error indicates that an internal error occurred while processing the request. Fatal indicates a problem with the query string that prevented the request from being processed. The <code>code</code> attribute specifies one of the following error codes:</p> <ul style="list-style-type: none"> • CS-InvalidFieldOrRankAliasInRankParameter—the specified ranking field could not be found. (Warning) • CS-RankExpressionParseError—one of the specified rank expressions could not be parsed. No query-time rank expressions will be used. (Warning) • CS-RankExpressionValidationError—one of the specified rank expressions could not be validated. No query-time rank expressions will be used. (Warning) • CS-UndefinedField—an unknown field was specified in the match expression. (Warning) • CS-UnknownFieldInMatchExpression—a field specified in the <code>fq</code> parameter could not be found. (Warning) • CS-WildcardTermLimit—more than 2000 terms matched the wildcard in the search request. The number of terms matched was limited to 2000. (Warning) • CS-IncorrectFieldTypeInMatchExpression—the type specified in the match expression does not match the field type. (Fatal) • CS-InvalidMatchSetExpression—the match expression could not be parsed. (Fatal) • CS-UnknownParameter—one of the specified parameters is not allowed. (Fatal)

Search Status Codes

A search request can return three types of status codes:

Amazon CloudSearch Developer Guide

Search Status Codes

- 5xx status codes indicate that there was an internal server error. We recommend catching and retrying all 5xx error codes as they typically represent transient error conditions.
- 4xx status codes indicate that the request was malformed.
- 2xx status codes indicate that the request was processed successfully.

Error	Description	HTTP Status Code
Not Found	The request path was not valid. This is often due to a missing or invalid API version in the URL. Consult the body of the response for details and fix the request before retrying.	404
Invalid HTTP Method	The HTTP method was not GET, HEAD, or OPTIONS. The search API does not support PUT or DELETE methods.	405
Request Timeout	The server did not receive a complete request within the time allowed.	408
Request URI Too Long	A GET request exceeded the maximum size of 8190 bytes. Use multiple simpler, smaller requests in place of one large request.	414
Internal Server Error	An internal problem occurred. The request can be retried.	500
Bad Gateway	This is typically an indication that your search service is overloaded. This can be due to the volume or complexity of search requests that you are submitting. The request can be retried. If necessary, Amazon CloudSearch will automatically scale your search domain to handle the load.	502
Gateway Timeout	A server timeout occurred while processing the request. The request can be retried.	504

Amazon CloudSearch Developer Guide
Search Status Codes

Error	Description	HTTP Status Code
Insufficient Storage	<p>A limit was reached and the request could not be processed. The request can be retried.</p> <p>507 and 509 errors are typically an indication that your search service is overloaded. This can be due to the volume or complexity of search requests that you are submitting. Amazon CloudSearch normally scales automatically to handle the load. Because it takes some time to deploy additional search instances, we recommend using an exponential backoff retry policy to temporarily reduce the request rate and minimize request failures. For more information, see Error Retries and Exponential Backoff.</p> <p>Certain usage patterns can sometimes result in timeouts without triggering automatic scaling. This is most often seen if you are submitting complex search queries to a single small search instance. If you continue to experience a high error rate, you can explicitly request additional capacity through the Amazon CloudSearch Service Limit Request form.</p>	507
Bandwidth Limit Exceeded	<p>The request was throttled. The request rate or resource consumption should be reduced before retrying the request.</p> <p>507 and 509 errors are typically an indication that your search service is overloaded. This can be due to the volume or complexity of search requests that you are submitting. Amazon CloudSearch normally scales automatically to handle the load. Because it takes some time to deploy additional search instances, we recommend using an exponential backoff retry policy to temporarily reduce the request rate and minimize request failures. For more information, see Error Retries and Exponential Backoff.</p> <p>Certain usage patterns can sometimes result in timeouts without triggering automatic scaling. This is most often seen if you are submitting complex search queries to a single small search instance. If you continue to experience a high error rate, you can explicitly request additional capacity through the Amazon CloudSearch Service Limit Request form.</p>	509

Handling Errors in Amazon CloudSearch

Topics

- [Error Types in Amazon CloudSearch \(p. 241\)](#)
- [Retrying Requests in Amazon CloudSearch \(p. 242\)](#)

This section provides information about how to handle errors when interacting with Amazon CloudSearch programmatically. For information about specific error codes returned by the Amazon CloudSearch services, see:

- [Search Status Codes \(p. 238\)](#)
- [documents/batch Status Codes \(p. 225\)](#)
- [Configuration Service Common Errors \(p. 241\)](#)

Error Types in Amazon CloudSearch

The HTTP status codes returned by the Amazon CloudSearch APIs indicate whether the request completed successfully, or if a client or server error occurred while processing the request:

- 2xx status codes indicate that the client request was processed successfully.
- 4xx status codes indicate that there was a problem with the client request. Common client request errors include providing invalid credentials and omitting required parameters. When you get a 4xx error, you need to correct the problem and resubmit a properly formed client request.
- 5xx status codes indicate that a server error occurred while processing the client request. Server errors are typically transient and are often the result of server timeouts, throttling, or capacity limitations. We recommend catching and retrying all 5xx errors.

An HTTP status code is returned for every request. In addition, the body of the response provides additional warning and error information.

Messages in a `search` response indicate the severity level, the warning or error code, and a description of the problem with the search request. For a list of the warnings and errors that can be returned by the

search service, see [Search Response Properties \(JSON\)](#) (p. 236) or [Search Response Elements \(XML\)](#) (p. 237).

Errors and warnings in a `documents/batch` response provide information about parsing and validation issues encountered while processing the SDF data. For more information, see [documents/batch Response \(JSON\)](#) (p. 220) or [documents/batch Response Elements \(XML\)](#) (p. 224).

Errors returned in a configuration service response provide information about what caused the request to return a 4xx or 5xx status code. For information about the common errors that all actions use, see [Common Errors](#) (p. 215). Action-specific errors are listed in the action topics in the [Amazon CloudSearch Configuration API Reference](#) (p. 164).

Retrying Requests in Amazon CloudSearch

For your application to run smoothly, you need to build in logic to catch and respond to errors. One typical approach is to implement your request within a try block or if-then statement.

We recommend catching and retrying all server errors (5xx). Because errors can be generated from anywhere within the request pipeline, you should implement a fallback for unexpected 5xx errors in addition to any special handling for specific status codes.

507 and 509 errors are typically an indication that your search service is overloaded. This can be due to the volume or complexity of search requests that you are submitting. Amazon CloudSearch normally scales automatically to handle the load. Because it takes some time to deploy additional search instances, we recommend using an exponential backoff retry policy to temporarily reduce the request rate and minimize request failures. For more information, see [Error Retries and Exponential Backoff](#).

Certain usage patterns can sometimes result in timeouts without triggering automatic scaling. This is most often seen if you are submitting complex search queries to a single small search instance. If you continue to experience a high error rate, you can explicitly request additional capacity through the Amazon CloudSearch [Service Limit Request](#) form.

Client errors (4xx) typically indicate that you need to revise the request to correct the problem—simply retrying the same request will most likely result in the same error. 409 errors returned by the configuration service can indicate that the request was rejected because a resource limit has been reached. For more information, see [Limits in Amazon CloudSearch](#) (p. 251).

Troubleshooting Amazon CloudSearch

The following topics describe solutions to problems you might encounter when using Amazon CloudSearch.

Topics

- [Uploading Documents](#) (p. 243)
- [Deleting All Documents in an Amazon CloudSearch Domain](#) (p. 244)
- [Amazon CloudSearch Domain Not Scaling Down After Deleting Documents](#) (p. 244)
- [Document Update Latency](#) (p. 244)
- [Retrieving a Document's Version Number](#) (p. 244)
- [Search Latency and Timeouts in Amazon CloudSearch](#) (p. 245)
- [Sudden Increase in 5xx Errors from Amazon CloudSearch](#) (p. 245)
- [Using Wildcards to Search Text Fields Doesn't Produce Expected Results](#) (p. 245)
- [Searches Not Returning All Matching Documents](#) (p. 246)

Uploading Documents

If your SDF is not formatted correctly or contains invalid values, you will get errors when you attempt to upload it or use it to configure fields for your domain. Here are some common problems and their solutions:

- **Invalid JSON**—if you are using JSON, the first thing to do is make sure there are no JSON syntax errors in your SDF batch. To do that, run it through a validation tool such as [Json Lint](#). This will identify any fundamental issues with the data.
- **Invalid XML**—SDF batches must be well-formed XML. You are especially likely to encounter issues if your fields contain XML data—the data must be XML-encoded or enclosed in CDATA sections. To identify any problems, run your SDF batch through a validation tool such as the [W3C Markup Validation Service](#).
- **Not Recognized as SDF**—if you are configuring your domain from SDF and Amazon CloudSearch doesn't recognize your data as valid SDF, it responds with a list of generic metadata fields:
 - content_encoding
 - content_language
 - content_type

- language
- resourcename

For example, this can happen if there are invalid document IDs or version numbers. Make sure that your SDF data contains all of the required properties for each document.

- **Document IDs with bad values**—capital letters, hyphens, and other special characters are not allowed in document IDs. Document IDs can only contain the characters a-z (lowercase letters), 0-9, and underscore (_). Document IDs must start with a letter or number; they cannot start with an underscore.
- **Bad version numbers**—version numbers must fit within a 32-bit unsigned integer. When specifying your SDF in JSON, make sure that the version number is not enclosed in quotes. If it is, the version is treated as a string and Amazon CloudSearch will reject the SDF as invalid.
- **Multi-valued fields without a value**—when specifying SDF in JSON, you cannot specify an empty array as the value of a field. Multi-valued fields must contain at least one value.
- **Bad characters**—one problem that can be difficult to detect if you do not filter your data while generating your SDF batch is that can contain characters that are invalid in XML. Both JSON and XML batches can contain only UTF-8 characters that are valid in XML. You can use a validation tool such as [Json Lint](#) or [W3C Markup Validation Service](#) to identify invalid characters.

Deleting All Documents in an Amazon CloudSearch Domain

Amazon CloudSearch currently does not provide a mechanism for deleting all of the documents in a domain. However, you can clone the domain configuration to start over with an empty domain. For more information, see [Cloning an Existing Domain's Indexing Options](#) (p. 64).

Amazon CloudSearch Domain Not Scaling Down After Deleting Documents

If your domain has scaled up to accommodate your index size and you delete a large number of documents, the domain scales down the next time the full index is rebuilt. While this is periodically done automatically, to scale down as quickly as possible you can explicitly [rebuild the index](#) (p. 82) when you are done deleting documents.

Document Update Latency

Sending a large volume of single-document batches can increase the amount of time it takes each document to become searchable. If have a large amount of update traffic, you need to batch your updates. We recommend using a batch size close to the 5 MB limit.

Retrieving a Document's Version Number

If you want to be able to query the index for your documents' version numbers, create a version field and populate it with the current version each time you update a document.

Search Latency and Timeouts in Amazon CloudSearch

If you are experiencing slow response times, frequently encountering internal server errors (typically 507 or 509 errors), or seeing the number of instance hours your search domain is consuming increase without a substantial increase in the volume of data you're searching, fine-tuning your search requests to reduce the processing overhead can help. For more information, see [Tuning Search Requests in Amazon CloudSearch](#) (p. 103).

507 and 509 errors are typically an indication that your search service is overloaded. This can be due to the volume or complexity of search requests that you are submitting. Amazon CloudSearch normally scales automatically to handle the load. Because it takes some time to deploy additional search instances, we recommend using an exponential backoff retry policy to temporarily reduce the request rate and minimize request failures. For more information, see [Error Retries and Exponential Backoff](#).

Certain usage patterns can sometimes result in timeouts without triggering automatic scaling. This is most often seen if you are submitting complex search queries to a single small search instance. If you continue to experience a high error rate, you can explicitly request additional capacity through the Amazon CloudSearch [Service Limit Request](#) form.

Sudden Increase in 5xx Errors from Amazon CloudSearch

If your search domain experiences a sudden spike in traffic, Amazon CloudSearch responds by adding search instances to your domain to handle the increased load. However, it takes a few minutes to set up the new instances. You are likely to see a temporary increase in 5xx errors until the new instances are ready to start processing requests. For more information about handling 5xx errors, see [Handling Errors in Amazon CloudSearch](#) (p. 241).

Using Wildcards to Search Text Fields Doesn't Produce Expected Results

Due to the way that Amazon CloudSearch tokenizes and stems text fields, there are two situations where you might not get the results you expect when using the * operator to search for a prefix string:

- If you search for an alphanumeric prefix, it might not match all of the documents you expect because the tokenization of the prefix string does not match the tokenization of the text field. While Amazon CloudSearch tokenizes both text fields and search strings (including prefix strings), the resulting tokens can be different when processing the shorter prefix string. To search alphanumeric prefixes, you can configure the field as a literal field, rather than a text field.
- If you search for a prefix that ends in `s`, it will not match occurrences of the term without the `s`. While Amazon CloudSearch normally performs basic stemming on text fields and search strings, when you use the * operator, the prefix string is *not* stemmed. If stemming is preventing your wildcard searches from returning all of the relevant matches, you can disable stemming for the text field or configure the field as a literal field.

For more information about how Amazon CloudSearch normalizes and tokenizes text and applies configured text options when indexing text fields and processing search requests, see [Text Processing in Amazon CloudSearch](#) (p. 247).

Searches Not Returning All Matching Documents

When Amazon CloudSearch tokenizes text fields, ASCII punctuation is treated as whitespace. However, non-ASCII (Unicode) punctuation characters such as curly quotes (“ ”) are treated as regular printable characters and indexed as-is. This can result in searches not matching all of the documents you expect. This is most likely to be an issue when you are indexing and searching user-generated content. One solution is to preprocess your data to remove any invalid characters and convert all punctuation to the ASCII equivalent before you upload it for indexing.

Another situation where a search might not return all of the results you expect is when you search for a prefix that ends in `s`. For more information, see [Using Wildcards to Search Text Fields Doesn't Produce Expected Results](#) (p. 245).

For more information about how Amazon CloudSearch normalizes and tokenizes text and applies configured text options when indexing text fields and processing search requests, see [Text Processing in Amazon CloudSearch](#) (p. 247).

Text Processing in Amazon CloudSearch

During indexing, Amazon CloudSearch (API Version 2011-02-01) processes text fields to determine what terms to add to the index. During this process, the text is *normalized* and *tokenized*, and the configured stems, stopwords, and synonyms are applied.

When you submit a search request, the text you're searching for undergoes the same text processing so that it can be matched against the terms that appear in the index. (Text processing is only done for text fields—no text processing is performed for literal fields.)

Normalization

Before any other processing is done, the text is normalized—that is, all upper case characters are converted to lower case, and accents are removed.

Tokenization

During tokenization, Amazon CloudSearch splits text strings into separate terms on detectable boundaries:

- Terms separated by whitespace are treated as separate tokens.
- Terms separated by ASCII punctuation are treated as separate tokens.

If a string contains both alphabetic and numeric characters and is at least three and no more than nine characters long, the alphabetic and numeric portions of the string are treated as separate tokens.

For example, the string `DOC298` is tokenized into two terms:

```
doc  
298
```

If there are multiple alphanumeric boundaries within a string, it is processed from left to right. If the first split results in an alphanumeric string that is at least three characters long, that string is split on the next alphanumeric boundary, and so on. For example, the string `DOC298G7` is indexed as three terms:

```
doc
298
g7
```

The string is split on the first boundary, which yields `doc` and `298g7`. `298g7` is then split on the next boundary, which results in `298` and `g7`. At this point, tokenization is complete because the resulting alphanumeric string, `g7`, is less than three characters.

Tokenization and Wildcard Searches

Because strings are tokenized on alphanumeric boundaries, wildcard searches might not behave the way that you expect. For example, if you have a text field that contains the string `FBAQ5L2`, it is tokenized during indexing as:

```
fbaq
5
12
```

If you search for `FBAQ5L*`, the query string is tokenized as `fbaq` and the prefix `51`. Since `51` is not in the index, the document that contains `FBAQ5L2` won't be returned as a match. Also keep in mind that while terms are both tokenized and stemmed during indexing, when you use the wildcard operator (`*`) in a query, the query string is only tokenized. For example, if you have a text field that contains the string `Kitties298`, Amazon CloudSearch splits it into two tokens during indexing, `kitties` and `298`, and then stems `kitties` to `kitty`:

```
kitty
298
```

If you search for `kitties2*`, however, the prefix string is tokenized as `kitties` and the prefix `2`. Because no stemming is done, this prefix search will not match the document that contains `kitties298`.

If you want to perform wildcard searches on alphanumeric strings, you might need to configure the field as a literal field rather than a text field to get the results you want. (No text processing is performed on literal fields.)

Tokenization and Non-ASCII Punctuation

If your data contains non-ASCII (that is, Unicode) punctuation characters such as curly quotes (`"`), keep in mind that they are not detected as boundaries during tokenization the way that ASCII punctuation characters are. While ASCII punctuation is treated as whitespace, non-ASCII punctuation is treated as regular printable characters and indexed as-is. For example, if a text field contains the string `"harry potter"`, the ASCII quotes are treated as whitespace and the string is tokenized as:

```
harry
potter
```

However, if a field contains the string `"harry potter"` with curly quotes, it is tokenized as:

```
"harry  
potter"
```

This means that searching for `harry potter` won't result in all the matches that you expect. This is most likely to be an issue when you are indexing and searching user-generated content. We recommend preprocessing the data to remove any invalid characters and convert all punctuation to the ASCII equivalent before you upload it for indexing. For more information, see [Uploading Data to an Amazon CloudSearch Domain \(p. 77\)](#).

Stemming

During indexing, Amazon CloudSearch uses each entry in your stemming dictionary to identify all of the word variations that should be indexed as occurrences of a particular stem. Only the stem is included in the index. When you submit a search request, Amazon CloudSearch stems the search string the same way. (Except when you search for a prefix—when you use the wildcard operator (*), the prefix string is not stemmed.)

For example, if you define `run` as the stem for the term `running`, the string `Running Man` is indexed as:

```
run  
man
```

When you search for the term `running`, it is also stemmed to `run`, which is a match for the indexed stem.

In addition to applying the stemming dictionary defined in your domain configuration, Amazon CloudSearch applies a basic stemming algorithm to map plurals to their singular form. This algorithm drops the plural `s` and `es` suffixes from terms, and maps `ies` suffixes to `y`.

To suppress stemming during indexing for a particular text field, you can specify the `cs_text_no_stemming` text processor when you configure the field with the [cs-configure-fields \(p. 143\)](#) command or the [DefineIndexField \(p. 167\)](#) configuration action.

For information about using a custom stemming dictionary, see [Configuring Stemming in Amazon CloudSearch \(p. 68\)](#).

Stemming and Wildcard Searches

Because plurals are automatically stemmed to their root during indexing, wildcard searches might not behave the way that you expect. When you use the wildcard operator (*) to search for a prefix, no stemming is performed on the prefix string. This means that a search for a prefix that ends in `s` won't match the singular version of the term. This can happen for any term that ends in `s`, not just plurals. For example, if you search the actor field in the sample movie data for `Gillanders`, there are three matching movies. If you search for `Gillander*`, you get the same three movies. If you search for `Gillanders*`, however, there are no matches. This is because the term stored in the index is `Gillander`, `Gillanders` does not exist.

If stemming is preventing your wildcard searches from returning all of the relevant matches, you can suppress stemming for the text field or map the data to a literal field instead of a text field.

Stopping

Amazon CloudSearch automatically removes a standard set of insignificant, frequently-occurring terms to reduce the size of the index and prevent searches from resulting in a massive number of hits. However, if multiple stopwords appear consecutively in a text field, they are not filtered out. This enables searches for phrases such as "to be or not to be" to return valid results.

By default, Amazon CloudSearch defines the following stopwords for English (en):

```
a  
an  
and  
are  
as  
at  
be  
but  
by  
for  
in  
is  
it  
of  
on  
or  
the  
to  
was
```

For information about defining your own stopwords, see [Configuring Stopwords in Amazon CloudSearch \(p. 71\)](#).

Note

Amazon CloudSearch always removes the following stopwords whether or not they are included in the stopword dictionary you configure: a, an, and, of, the.

Synonyms

Defining appropriate synonyms depends heavily on the content being indexed, so Amazon CloudSearch does not use a default synonym dictionary. If you want to use synonyms to facilitate retrieval based on alternate searches, you need to define a custom synonym dictionary. For example, you might define custom synonyms to:

- Map common misspellings to the correct spelling.
- Define equivalent terms, such as `film` and `movie`.
- Map a general term to a more specific one, such as `fish` and `barracuda`.

When you define a synonym, the synonym is added to the index everywhere the base token occurs. For example, if you define `fish` as a synonym of `barracuda`, the term `fish` is added to every document that contains the term `barracuda`. Adding a large number of synonyms can increase the size of the index as well as query latency—synonyms increase the number of matches and the more matches, the longer it takes to process the results.

For more information, see [Configuring Synonyms in Amazon CloudSearch \(p. 74\)](#).

Limits in Amazon CloudSearch

This table shows naming and size restrictions within Amazon CloudSearch. You can [submit a request](#) if you need to increase the maximum number of search instances or partitions for a search domain. For information about increasing other limits such as the maximum number of search domains, contact Amazon CloudSearch.

The current Amazon CloudSearch limits are summarized in the following table.

Item	Limit
Domain name	Allowed characters are a-z (lower-case letters), 0-9, and hyphen (-). Domain names must start with a letter or number and be at least 3 and no more than 28 characters long.
Field name	Allowed characters are a-z (lower-case letters), 0-9, and _ (underscore). Field names must begin with a letter and be at least 1 and no more than 64 characters long. The names "body", "docid", and "text_relevance" are reserved names and cannot be specified as field names.
Rank expression name	Allowed characters are a-z (lower-case letters), 0-9, and _ (underscore). Rank expression names must begin with a letter and be at least 3 and no more than 64 characters long. The names "body", "docid", and "text_relevance" are reserved names and cannot be specified as rank expression names.
Source field name	Source field names must be at least 1 and no more than 64 characters long.
Document ID (docid)	Allowed characters are: a-z (lower-case letters), 0-9, and _ (underscore). Document IDs must begin with a letter or numeral and must be at least 1 and no more than 128 characters long.
Document size	The maximum document size is 1 MB.
Batch size	The maximum batch size is 5 MB.
Document version number size	The maximum size of a document's version number is max(uint32_t).
Document language	English (en) is currently the only supported language.

Item	Limit
Maximum number of index fields	Up to 200 index fields can be configured for a domain.
Maximum number of sources for an index field	Up to 20 sources can be configured for a field.
Maximum number of field values	Up to 100 values can be specified in a field.
Maximum size of terms in an index field	Individual terms within a text or literal field are truncated if they exceed 256 characters.
Default value size	The maximum size of a default value for a field is 1 KB.
Uint field range	A uint field can contain values in the range 0 - max(uint32_t).
Maximum number of rank expressions	Up to 50 rank expressions can be configured for a domain.
Rank expression size	The maximum size of a rank expression is 10240 bytes. The maximum value that can be returned by a rank expression is max(uint32_t).
text_relevance score	An integer value in the range 0-1000.
Maximum search partitions	10
Maximum search instances	50
Policy document size	The maximum size of a Amazon CloudSearch policy document is 100 KB.
Stemming dictionary size	The maximum size of a Amazon CloudSearch stemming dictionary is 500 KB.
Stopwords dictionary size	The maximum size of a Amazon CloudSearch stopwords dictionary is 10 KB.
Synonym dictionary size	The maximum size of a Amazon CloudSearch synonym dictionary is 100 KB.
Search requests: size parameter	The size parameter can contain values in the range 0 - max(uint32_t).
Search requests: start parameter	The start parameter can contain values in the range 0 - max(uint32_t).
Search requests: rank parameter	Up to 10 uint fields and expressions can be specified in the rank parameter.
Search requests: GET requests	The maximum size of a search request submitted as an HTTP GET request is 8190 bytes.
Search requests: returned data	Up to 2 KB of data can be returned from a field. If the field contents exceed 2 KB, only the first 2 KB is included in the results.
Search requests: wildcard term limit	Prefix searches are expanded to a maximum of 2,000 indexed terms. If more than 2,000 terms match the prefix, the search results will not include all possible matches.
Number of Search Domains	Each AWS account can create up to 100 search domains.

Amazon CloudSearch Articles and Tutorials

For additional information about using Amazon CloudSearch, see the following articles and tutorials:

[Using Amazon CloudSearch with MySQL/Amazon RDS \(p. 255\)](#)

Integrating Amazon CloudSearch into your application gives you a world-class full-text search solution that can leverage your relational database's strengths and deliver an optimal user experience. This article shows you how to integrate a MySQL-based application with Amazon CloudSearch, and steps through how to analyze your data set, upload your data, submit search requests, and synchronize updates.

[Guide to Formatting Your Data in SDF for Amazon CloudSearch](#)

[Searching DynamoDB Data with Amazon CloudSearch](#)

While DynamoDB supports GET and query, it lacks a rich query capability—there is no simple way to retrieve items based on matches within item attributes. By using DynamoDB and Amazon CloudSearch together, you get both the throughput and durability of DynamoDB and the rich and powerful search capabilities of Amazon CloudSearch: simple text matches, complex Boolean queries, faceting, and integer range searching. This guide describes how to use Amazon CloudSearch to search your DynamoDB data.

[Guide to Formatting Your Data in SDF for Amazon CloudSearch](#)

Search Data Format (SDF) is the structured data format that you use to represent the data that you want to index and search with Amazon CloudSearch. This guide describes how to structure your data to support searching, describe it in SDF, and validate the SDF before uploading it to your search domain.

[Guide to Configuring Index Fields for an Amazon CloudSearch Domain](#)

A search domain's index fields control how document data in Search Data Format (SDF) is handled during indexing and how you will be able to search and use the data once your index is built. This guide will help you understand how the index field configuration impacts index size, performance, and the cost of your Amazon CloudSearch domain. It describes how to decide what index field types to use, how the document data is mapped to the index fields, and how you can configure fields to support faceting and return data in the search results.

[Guide to Using Elastic IPs to Manage Access to Amazon CloudSearch Domains](#)

When creating and configuring search domains, you use your AWS access keys for authentication. To control access to a particular search domain's document and search endpoints, you need to whitelist the specific IP addresses or address ranges that can submit document updates and search requests. This guide describes how to use elastic IPs to manage access to your document and search endpoints from EC2.

Using Amazon CloudSearch with MySQL/Amazon RDS

The best of both worlds: World-class full-text search and RDBMS performance

Topics

- [Abstract \(p. 255\)](#)
- [Full-text Search \(p. 255\)](#)
- [Integrating Amazon CloudSearch with MySQL/Amazon RDS \(p. 257\)](#)
- [Conclusion \(p. 264\)](#)

Abstract

Relational Database Management Systems such as Amazon RDS offer a simple, efficient way to store application data. You can quickly retrieve the specific pieces of information your application needs, even when you have a large amount of data. However, the very nature of a relational database such as MySQL makes it a less-than-optimal solution for searching a large volume of text to find the items that are most relevant to a set of search terms. Amazon CloudSearch offers a rich set of search features optimized for full-text search, including configurable search fields and text processing options, faceting, and customizable relevance ranking. Integrating Amazon CloudSearch into your application gives you a world-class full-text search solution that can leverage your relational database's strengths and deliver an optimal user experience. This article shows you how to integrate a MySQL-based application with Amazon CloudSearch, and steps through how to analyze your data set, upload your data, submit search requests, and synchronize updates.

Full-text Search

A relational database like MySQL (Amazon RDS for MySQL) does a great job of storing structured data and providing efficient access to specific items. In contrast, a full-text search system like Amazon CloudSearch excels at efficiently indexing large amounts of text to enable arbitrary searches. The following

table compares how data is organized, accessed, and retrieved in relational databases and Amazon CloudSearch domains.

Relational Databases	Amazon CloudSearch Domains
Organize data into tables. Every row in a table has the same set of fields.	Organize data into documents. It is not necessary for every document to have the same set of fields.
Provide access to data through SQL statements.	Provide access to data through REST-style requests that support complex Boolean queries.
Order results based on data stored in the table, such as an item's title or price field.	Order results based on relevance to the search request. Can also calculate custom relevance scores and sort results using field values.

While there's some overlap with relational database functionality, Amazon CloudSearch offers a feature set that's optimized to support full-text search. As you can see from the following table, most applications need the capabilities of a relational database as well as a full-text search solution like Amazon CloudSearch.

Features	Relational Database	Amazon CloudSearch
Transaction support	✓	–
Stored procedures	✓	–
Handles normalized data	✓	✓ (using denormalization)
Relevance ranking	Limited	✓
Natural language searches	–	✓
Unstructured data	–	✓
High volume of queries	–	✓
High volume of writes	✓	✓
High volume of text	–	✓

What About MySQL's Natural Language and Boolean Search Capabilities?

The full-text search capabilities in MySQL have some important limitations:

- MySQL full-text search tends to perform poorly with large data sets.
- The natural language search engine ignores short words, even if they're significant.
- Sorting based on relevance is not supported and there's no way to customize result ranking.
- The support for tokenization, stemming, and synonyms is minimal.

If full-text search is integral to your application, relying on MySQL's search capabilities might result in a less-than-optimal user experience. A better solution is to use your database for what it's best at: storing and retrieving structured data. Running Amazon CloudSearch in parallel with your relational database gives you the best of both worlds: world-class full-text search and RDBMS performance for managing structured data.

Integrating Amazon CloudSearch with MySQL/Amazon RDS

When you integrate Amazon CloudSearch into an RDBMS-based application, your application workflow for search looks like this:

1. Create a search request based on the user-specified information.
2. Submit the search request to your Amazon CloudSearch domain. Amazon CloudSearch searches your indexed data, identifies the documents that match the query, and returns the search results as a ranked list of matching documents. At a minimum, the results include the ID of each matching document. You can also retrieve other document data and facet information.
3. Display the search results to the user. When presenting the search results, you might display the information returned in the search results directly, or use the document IDs or other information to retrieve supplemental information from your database.
4. When the user selects a particular result, retrieve the corresponding record from the database to display a detail view.

To integrate Amazon CloudSearch into your application, you need to:

1. Analyze your data and configure an Amazon CloudSearch domain.
2. Extract and upload your data to your search domain.
3. Update your application to create and submit search requests and display the results.
4. Periodically update your search domain as changes are made to your database.

As an example, consider an e-commerce application for an online retailer that sells three categories of products: books, electronics, and mobile devices. In this application, we want to support three search-related features:

- Basic product search—enable customers to search all of the product titles and descriptions in each category.
- Advanced search—enable customers to filter results by price, manufacturer, and capabilities such as the amount of memory a mobile device has.
- Product detail views—enable customers to view all of the information available for any item shown in the search results.

The basic and advanced search capabilities are best supported by Amazon CloudSearch. To display the product detail views, we can use the document ID returned in the search results to look up the corresponding product information in the database.

Analyzing Your Data and Configuring a Search Domain

Before you can set up your search domain, you need to determine:

- What data you want to search.
- What information you want to be included in the search results.
- What information you can retrieve from your database.
- What attributes you want to use to refine the search results.

This information will guide what data you upload to your search domain for indexing and how you configure the corresponding index fields. For example, if you're not going to enable users to search on a particular attribute, you don't need to index it. Similarly, if the data doesn't need to be returned in the search results, you don't need to store it in your index. These are important considerations because the more fields you index and the more data you store in your index, the bigger it will be.

Because the size of your index has a direct impact on the cost of operating your search domain, you should:

- Only index the fields you are going to use.
- Use a key returned in the search results to look up data stored in your database, rather than duplicating the data in your search index.

By analyzing your existing database schema, you can evaluate what you want to search, what information you want to be able to display to the user, and what attributes are useful for refining the search results.

In our e-commerce example, each category of products is represented by a table in the database, and each product is a record within one of those tables. Manufacturer and device information is stored in two additional tables that are referenced by the product tables. (The data is normalized to reduce duplication.)

```
create table books (  
  ID integer PRIMARY KEY,  
  title varchar (255),  
  description text,  
  publisher_ID integer FOREIGN KEY manufacturers.ID,  
  Price numeric,  
  ISBN varchar (10))  
  
create table electronics (  
  ID integer PRIMARY KEY,  
  title varchar (255),  
  description text,  
  manufacturer_ID integer FOREIGN KEY manufacturers.ID,  
  Price numeric,  
  type ID_integer FOREIGN KEY devices.ID)  
  
create table mobile (  
  ID integer PRIMARY KEY,  
  title varchar (255),  
  description text,  
  manufacturer_ID integer FOREIGN KEY manufacturers.ID,  
  carrier_ID integer FOREIGN KEY manufacturers.ID,  
  Price numeric,  
  contract Boolean,  
  type_ID integer FOREIGN KEY devices that ID)  
  
create table manufacturers (  
  ID integer PRIMARY KEY,  
  manufacturer_name varchar (255),  
  address varchar (255))  
  
create table devices (  
  ID integer PRIMARY KEY,  
  device_type varchar (255))
```

Of the information stored in these tables, there are nine fields we want to use to support search in our e-commerce application:


```
product_ID (integer)
title (varchar)
description (text)
price (numeric)
manufacturer (varchar)
ISBN (varchar)
device_type (varchar)
carrier (varchar)
contract (boolean)
```

These fields represent all of the information about each product that we want to index. We've excluded fields such as the manufacturer's address that we don't need to be able to search—to display the address in the detail view, we can just look it up in the manufacturer table. We've included the `contract` field to enable users to filter the search results according to whether or not a mobile device requires a contract.

Note that there's a single list of fields. When we populate our search domain, we're de-normalizing the data—the relevant fields will be stored directly in each product document. Not every document will contain all of the fields—for example, the `contract` field is irrelevant for books, so the documents that represent books don't need to include that field.

Once we've identified which fields we're going to use, we need to map them to the index field types supported by Amazon CloudSearch. In a search domain configuration, data is represented as one of three types:

- **uint**—a 32-bit unsigned integer. To store floating point values, you can multiply by a constant factor to convert the values to integers. For example, to store prices represented in dollars and cents, multiply the price by 100 to store it as an integer that represents the value in cents—48.75 is stored as 4875.
- **text**—an arbitrary text string that will be normalized, tokenized, and processed according to the text processing options configured for the domain.
- **literal**—a string that you want to be able to match exactly, such as a product code or email address. (No text processing is performed on literal fields.)

Next, we need to identify which fields we want to search, return in the search results, and use as facets:

- Text and uint fields are always searchable. To search a literal field, you must explicitly configure it to be searchable.
- The value of a uint field is always included in the search results. To return the value of a text or literal field, you must explicitly configure it as result-enabled.
- Uint fields can always be used as facets. To get facet information for a text or literal field, you must explicitly configure it as facet-enabled.

Note that the value of a text or literal field can be returned in search results or the field can be used as a facet, but not both. If you want to use a returnable field as a facet, you can create an additional index field that's mapped to the same source field. For more information, see the [Guide to Configuring Index Fields for an Amazon CloudSearch Domain](#).

After mapping the fields, the index configuration for our e-commerce application looks like this:

Field	Type	Search	Facet	Result
product_id	uint	–	–	✓
title	text	✓	–	✓
description	text	✓	–	✓

Field	Type	Search	Facet	Result
price	uint	✓	✓	✓
manufacturer	literal	✓	✓	✓
isbn	literal	✓	–	✓
device_type	literal	✓	✓	✓
carrier	literal	✓	✓	✓
contract	literal	✓	✓	✓

You can create and configure a search domain through the Amazon CloudSearch console or with the command line tools or Configuration APIs. For step-by-step instructions for creating and configuring a search domain, see [Getting Started with Amazon CloudSearch](#).

It can take up to 30 minutes to provision and deploy a new search domain. While you're waiting, you can start preparing your data for indexing.

Extracting and Uploading Your Data

Data you upload to Amazon CloudSearch must be formatted in the Search Data Format (SDF), which can be encoded as either JSON or XML. SDF data provides all of the information Amazon CloudSearch needs for indexing. Each item that you want to be able to return as a search result (such as a product) is represented as a document in an SDF batch. An SDF batch is simply a collection of add and delete requests for individual documents. Every document has a unique ID, a version number, and one or more fields that contain the data that you want to search and return in results.

For example, this snippet of an SDF batch shows an add request for a book available in our e-Commerce application:

```
[
  {
    "type": "add",
    "id": "id99200034",
    "version": 1,
    "lang": "en",
    "fields": {
      "product_id": 99200034,
      "title": "Searching For Love",
      "description": "A sweeping tale of romance set against a backgro..."
      "price": "9.99",
      "manufacturer": "Romance Books Unlimited",
      "isbn": "9999999999",
      "manufacturer_address": "123 Main St, Anywhere, US"
    }
  },
  ...
]
```

The batch is an array of JSON objects, each of which specifies an individual document. (You can also encode SDF as XML.) The type indicates whether you are adding (or updating) the document or deleting it. The version is used to ensure that Amazon CloudSearch applies changes in the correct order—add and delete requests are only applied if the version number is greater than the last version that was applied, older versions are ignored. For more information about versioning, see [Document Versions in Amazon CloudSearch](#).

How you extract data from your database and generate SDF batches depends largely on your application and preferences. One option is to dump the data from your database and then use a script to parse the data and generate SDF batches. Alternatively, your script could read directly from the database.

When you generate SDF batches, there are several things to consider.

- The document ID value must uniquely identify the document. While you don't have to map the primary key in your table to the document ID, doing so makes it simpler to fetch data from the table when you're processing search results.
- If you de-normalize the data referenced by other tables and embed those fields directly in each document, you can use those fields as facets. For example, in the e-Commerce database schema each book, electronics item, and mobile device references the `manufacturers` table. If you insert the data from the `manufacturers` table into each document, you can facet by manufacturer. While this does result in increased data redundancy and increase the size of your index, the alternative would be to include the `manufacturers` table data as its own set of documents. In addition to not being able to facet on the manufacturer data, this would require that you perform an additional query to display manufacturer information for an item.
- To represent one-to-many relationships, you can specify an array of values as the value for a field.
- You might need to rearchitect many-to-many relationships, as there's no simple way to represent them in an SDF document.

For more information about generating SDF batches, see the [Guide to Formatting Your Data in SDF for Amazon CloudSearch](#) and [Preparing Your Data for Amazon CloudSearch](#).

Before uploading your generated SDF batches, you should verify that they are valid JSON or XML and contain only UTF-8 characters that are valid in XML. If your SDF batches are not formatted correctly or contain invalid values, you will get errors when you attempt to upload or use them to configure fields for your domain. For more information about validating SDF batches and troubleshooting upload errors, see [Troubleshooting Amazon CloudSearch \(p. 243\)](#).

Once you've generated and validated your SDF batches, you're ready to upload them to your search domain for indexing. For testing, you can upload SDF documents that are 5 MB or less through the Amazon CloudSearch console. If you're scripting the process, you can use the `cs-post-sdf` command or directly post batches to your domain's document service endpoint. A domain's document service endpoint is displayed on the domain dashboard in the Amazon CloudSearch console.

For example, the following HTTP request sends an SDF batch as the body of a POST request to a search domain's `documents/batch` resource:

```
curl -X POST --upload-file data.json --header "Content-Type: application/json"
doc-your-search-domain-name-yoursearchdomainid.us-east-1.cloudsearch.amazonaws.com/2011-02-01/documents/batch
```

For more information, see [Uploading Data to an Amazon CloudSearch Domain](#).

Searching Your Data with Amazon CloudSearch

Once you've uploaded data to your search domain, you're ready to start searching. To do that, you send HTTP GET requests to your domain's search endpoint. A domain's search endpoint is displayed on the domain dashboard in the Amazon CloudSearch console.

To see what a search request looks like, you can use the Search Tester in the Amazon CloudSearch console. You can search directly from your search domain's dashboard, or click the **Run a Test Search** link in the navigation panel to open the Search Tester. For more information, see <http://docs.aws.amazon.com/cloudsearch/latest/developerguide/GettingStartedSearch.html#searching.console>.

If you've configured facets for your domain, they are displayed in the **Filter Search Results** column.

To see the actual search request and the raw result information for your query, select the response format you want to view from **view raw: JSON or XML**. For example, if you select JSON, the actual search request that's submitted for the query shown above is:

```
http://search-samplecatalog-awfmweeb4nulbcmqh3hae7r45q.us-west-2.cloud
search.amazonaws.com/2011-02-01/search?q=romance&return-fields=contract%2Cde
scription%2Cisbn%2Cmanufacturer%2Cmanufacturers_ad
dress%2Cprice%2Cproduct_id%2Ctitle%2Ctext_relevance
```

To experiment with different search options, you can copy this request as a starting point, modify it, and then paste the URL into any Web browser.

Note that JSON is the default response format. To retrieve the response as XML, you specify XML as the `results-type`:

```
http://search-samplecatalog-awfmweeb4nulbcmqh3hae7r45q.us-west-2.cloud
search.amazonaws.com/2011-02-01/search?q=romance&return-fields=contract%2Cde
scription%2Cisbn%2Cmanufacturer%2Cmanufacturers_ad
dress%2Cprice%2Cproduct_id%2Ctitle%2Ctext_relevance&results-type=xml
```

Here's an example of what the raw search results look like formatted in JSON:

```
{
  "rank": "-text_relevance",
  "match-expr": "(label 'romance')",
  "hits": {
    "found": 1,
    "start": 0,
    "hit": [
      {
        "id": "id99200034",
        "data": {
          "contract": [],
          "description": ["A sweeping tale of romance set..."],
          "isbn": ["9999999999"],
          "manufacturer": ["Romance Books Unlimited"],
          "manufacturers_address": [],
          "price": ["9"],
          "product_id": [],
          "text_relevance": ["259"],
          "title": ["Searching For Love"]
        }
      }
    ]
  },
  "info": {
    "rid": "4e25a45517275041e691716c6405b16e32b7b188bffa53e1ff24ecfb441d",
    "time-ms": 3,
    "cpu-time-ms": 0
  }
}
```

A couple things to note in the search results:

- The matching documents are sorted according to the `rank` value. The default ranking is `-text_relevance`, which orders results in the descending order of their `text_relevance` scores. (The most-relevant documents are listed first.) A document's `text_relevance` score indicates how relevant it is to the search request, taking into account how many times the search terms appear (term frequency) and how close the search terms are to each other (proximity). You can also define and use custom rank expressions in your search requests. This is particularly useful if your existing application uses functions for ordering.
- The `match-expr` value shows the match expression generated from the `q` and `bq` parameters in your request. Amazon CloudSearch provides a sophisticated query language that enables you to build complex Boolean queries. For more information, see [Constructing Boolean Search Queries in Amazon CloudSearch](#).

To use Amazon CloudSearch from your application, you need to submit an HTTP GET request to your domain's search endpoint and parse the returned JSON or XML.

If you are upgrading an application that used MySQL for search, you need to replace your direct database query with an Amazon CloudSearch request. For example, you might have PHP code that looks like this:

```
<?php
$conn = mysql_connect("localhost", "mysql_user", "mysql_password");
if (!$conn) { echo "Connection error: " . mysql_error(); die;}
mysql_select_db("catalog");

$sql = "select * from books where MATCH (title, description) AGAINST
('".mysql_real_escape_string($_REQUEST['keyword']).")
UNION
select * from electronics where MATCH (title, description) AGAINST
('".mysql_real_escape_string($_REQUEST['keyword']).")
UNION
select * from mobile where MATCH (title, description) AGAINST
('".mysql_real_escape_string($_REQUEST['keyword']).")";

$result = mysql_query($sql);
if (!$result) {
    echo "Error running query $sql: " . mysql_error();
    exit;
} else if (mysql_num_rows($result) == 0) {
    echo "Zero results for keyword ".$_REQUEST['keyword'];
    exit;
} else {
    while ($row = mysql_fetch_assoc($result)) {
        echo $row["title"]." -- ";
        echo $row["description"]."<br />";
    }
}
mysql_free_result($result);
?>
```

To use Amazon CloudSearch instead, you need to:

1. Submit a request to your domain's search endpoint.
2. Parse the JSON response into an array (by setting the `assoc` parameter to `TRUE`).
3. Loop through the results to display the data.

For example:

```
<?php

$results = file_get_contents("http://search-samplecatalog-awfmweeb4nublbcmqh3hae7r45q.us-west-2.cloudsearch.amazonaws.com/2011-02-01/search?q=" . $_REQUEST['keyword'] . "&return-fields=contract%2Cdescription%2Cisbn%2Cmanufacturer%2Cmanufacturers_address%2Cprice%2Cproduct_id%2Ctitle%2Ctext_relevance");

$resultsArr = json_decode($results, TRUE);

$i = 0;
while ($i < $resultsArr["hits"]["found"]){
    echo $resultsArr["hits"]["hit"][$i]["data"]["title"] . " -- ";
    echo $resultsArr["hits"]["hit"][$i]["data"]["description"] . "<br />";
    $i++;
}
?>
```

Synchronizing Updates

As the data in your database changes, you need to update your search index so that search results reflect those changes. To keep Amazon CloudSearch in sync with your database, you need to either:

- Send all changes to your search domain as well as your database.
- Periodically poll your database for changes and upload them to your search domain.

If you choose to periodically poll your database for changes, you also need to keep track of deleted items so you can submit delete requests to your search domain. Alternatively, you can mark records as deleted in the database, rather than removing them.

To submit new or changed items for indexing, you include an add document request in an SDF batch. To delete items, you include a delete document request. For example, to delete the "Searching For Love" novel, you'd add:

```
{
  "type": "delete",
  "id": "id99200034",
  "version": 2
}
```

For a document to be deleted, the document ID must match an existing document and the version number must be greater than the last applied version number.

If you poll your database for changes, adding a `last_modified_date` to your database schema and domain configuration makes it easy to find and apply all of the changes since the last update.

Conclusion

Relational databases like MySQL provide a reliable datastore for a wide variety of applications. However, they offer limited search capabilities and are unable to provide the quality search results users have come to expect. By combining the strengths of an RDBMS with the world-class full-text search capabilities offered by Amazon CloudSearch, your applications can deliver an optimal user experience.

To do that, you need to analyze your database schema, identify the data you need to support your search solution, map those attributes to index fields in an Amazon CloudSearch domain, and upload the data to your search domain for indexing.

As your data changes, you need to synchronize updates to your database with your search domain. You can choose to send updates to both your database and search domain, or periodically poll your database for updates and propagate them to your domain. If you choose to poll your database for updates, you also need to track and submit delete requests.

Amazon CloudSearch Glossary

This section provides a summary of Amazon CloudSearch terminology. For the complete Amazon AWS glossary, see the [AWS General Reference](#).

Amazon CloudSearch	A fully-managed service in the cloud that makes it easy to set up, manage, and scale a search solution for your website.
batch	A collection of add and delete document operations in Search Data Format (SDF). You use the document service API to submit add and delete document operations to update the data in your search domain.
configuration API	The API that you use to create, configure, and manage search domains.
corpus	A collection of data that you want to search.
document	Represents an item that can be returned as a search result. Each document has a collection of fields that contain the data that can be searched or returned. The value of a field can be either a string or a number. Each document must have a unique ID, a version number, and at least one field.
document ID (docid)	A unique alpha-numeric identifier for a document. This is the <code>id</code> attribute that's specified in an add or delete operation when using the document service API to update your search domain.
document service API	The API that you use to submit SDF batches to update the data in your search domain.
document service endpoint	The URL that you connect to when sending document updates to a search domain. Each search domain has a unique document service endpoint.
document version	In SDF, each document has a numeric version number that's used to guarantee that a search domain always reflects the most recent document updates. Document updates are applied only if the version number specified in the add or delete operation is <i>greater</i> than the existing version number.
domain	See search domain .

facet	A search index field that represents a category that you want to use to refine and filter search results.
facet constraint	A particular facet value that you want to count when searching.
facet enabled	A search index field option that enables facet information to be calculated for the field.
field weight	The relative importance of a text field in a search index. Field weights control how much matches in particular text fields affect a document's <code>text_relevance</code> score.
hit	A document that matches the criteria specified in the search request. Also referred to as a <i>search result</i> .
index	See search index.
index field	A name-value pair that is included in a search domain's index. An index field can contain text, literal, or unsigned integer data.
index field name	The name of a text, literal, or uint field in a search index.
indexing options	Configuration settings that define a search domain's index fields, how SDF data is mapped to those index fields, and how the index fields can be used.
query time rank expression	A rank expression (p. 267) that's defined within a search request. You can use a query time rank expression to rank results for a request or set a threshold for the search results.
rank expression	A numeric expression that you can use to control how search hits are ranked. You can construct rank expressions using uint fields, other rank expressions, a document's default <code>text_relevance</code> score, and standard numeric operators and functions. When you use the <code>rank</code> option to specify a rank expression in a search request, the expression is evaluated for each search hit and the hits are listed according to their rank expression values.
result enabled	An index field option that enables the field's value(s) to be returned in the search results.
SDF	See Search Data Format.
search API	The API that you use to submit search requests to an Amazon CloudSearch domain.
Search Data Format	The format that you use to describe the data that you want to add or delete from an Amazon CloudSearch domain. Search Data Format (SDF) can be represented as either JSON or XML.
search domain	Encapsulates your searchable data and the search instances that handle your search requests. You set up a separate domain for each different collection of data that you want to search.
search domain configuration	A search domain's indexing options, text options, access policies, and rank expressions.
search domain name	A user-specified name that is used to construct a unique identifier for a search domain.
search enabled	An search index field option that enables the field data to be searched.

search index	A representation of your searchable data that facilitates fast and accurate data retrieval.
search instance	A compute resource that indexes your data and processes search requests. A search domain has one or more search instances, each with a finite amount of RAM and CPU resources. As your data volume grows, more search instances or larger search instances are deployed to contain your indexed data. When necessary, your index is automatically partitioned across multiple search instances. As your request volume or complexity increases, each search partition is automatically replicated to provide additional processing capacity.
search request	A request that is sent to a search domain to retrieve documents that match particular search criteria.
search result	A document that matches a search request. Also referred to as a <i>search hit</i> .
search service endpoint	The URL that you connect to when sending search requests to a search domain.
source	An SDF document field that is used to populate a search index field.
stem	The common root or substring shared by a set of related words.
stemming	The process of mapping related words to a common stem. This enables matching on variants of a word. For example, a search for "horse" could return matches for horses, horseback, and horsing, as well as horse.
stemming dictionary	A domain-specific collection of mappings of words to their stems. Amazon CloudSearch does not define a default stemming dictionary.
stopping	The process of filtering stop words from an index or search request.
stopword	A word that is not indexed and is automatically filtered out of search requests because it is either insignificant or so common that including it would result in too many matches to be useful. Stop words are language-specific.
stopword dictionary	A domain-specific collection of stopwords. Amazon CloudSearch defines a default stopwords dictionary for English that you can use as-is, or customize to suit your collection of data.
synonym	A word that is the same or nearly the same as an indexed word and that should produce the same results when specified in a search request. For example, a search for "Rocky Four" or "Rocky 4" should return the fourth <i>Rocky</i> movie. This can be done by designating that <i>four</i> and <i>4</i> are synonyms for <i>IV</i> . Synonyms are language-specific.
synonym dictionary	A domain-specific collection of synonym mappings. Amazon CloudSearch does not define a default synonym dictionary.
text_relevance	A built-in relevance score that's based on the repetition of search terms in the document and proximity of search terms to each other in each matching index field in the document. A document's <code>text_relevance</code> score is an integer value from 0 to 1000 (inclusive).
text options	Domain-specific stopwords, stemming, and synonym dictionaries used during text processing when building a search index. Stopwords and

stems are also used at search time to process the search terms before looking for matching documents in the index.

tokenization

Part of the text processing that Amazon CloudSearch performs when indexing and processing search requests. During indexing, the contents of each text field are split into a collection of tokens that can be indexed separately. Punctuation is stripped and each word (that isn't in the stopword list) becomes a token. For example, the string "spider-man" would be split into two tokens: *spider* and *man*. At search time, the search terms are tokenized using the same rules before being matched against the indexed tokens.

version

See document version.

Document History for Amazon CloudSearch

This Document History describes the important changes to the documentation in this release of *Amazon CloudSearch*.

Relevant Dates to this History:

- **Current product version**—2011-02-01
- **Latest product release**—29 August 2013
- **Last documentation update**—15 January 2014

Change	Description	Release Date
Using Amazon CloudSearch with MySQL/Amazon RDS	Using Amazon CloudSearch with MySQL/Amazon RDS (p. 255) is a new article that describes how to integrate a MySQL-based application with Amazon CloudSearch, and steps through how to analyze your data set, upload your data, submit search requests, and synchronize updates.	5 September 2013
Support for Uploading Data from DynamoDB	Searching DynamoDB Data with Amazon CloudSearch (p. 117) describes how to upload data from DynamoDB to configure a search domain and index data stored in a DynamoDB table. This release also improves how CSV files are processed when generating SDF batches.	29 August 2013
Added Information about Automatic Scaling	Automatic Scaling in Amazon CloudSearch (p. 6) describes how Amazon CloudSearch scales a search domain to handle changes in the volume of data and traffic.	17 April 2013
Additional Endpoints, Tips for Error Handling and Tuning Searches	Amazon CloudSearch has expanded to the US West (Oregon), US West (N. California), and Asia Pacific (Singapore) Regions. For a current list of supported regions and endpoints, see Regions and Endpoints . This document update also includes two new sections: Handling Errors in Amazon CloudSearch (p. 241) and Tuning Search Requests in Amazon CloudSearch (p. 103) .	27 February 2013

Change	Description	Release Date
New Endpoint	Amazon CloudSearch has expanded to the EU (Ireland) Region. For a current list of supported regions and endpoints, see Regions and Endpoints .	05 February 2013
Searching and Ranking Results by Geographic Location in Amazon CloudSearch	Although Amazon CloudSearch does not have a native type to support latitude and longitude, you can implement geographically-based searching and sorting by representing latitude and longitude as integers. For more information, see Searching and Ranking Results by Geographic Location in Amazon CloudSearch (p. 114).	30 January 2013
Viewing Search Analytics Reports and Comparing Rank Expressions in the Amazon CloudSearch Console	Through the console, you can now access information about your search traffic and view a side-by-side comparison of how results are ranked using two different rank expressions. For more information, see Viewing Search Metrics for an Amazon CloudSearch Domain (p. 44) and Comparing Rank Expressions in Amazon CloudSearch (p. 111).	18 December 2012
Added Support for Field-Weighted Relevance and Query Time Rank Expressions	You can now customize how text relevance is calculated by assigning relative weights to text and literal fields. To test and tune your rank expressions, you can define rank expressions within a search request and use them to rank and threshold the search results. For more information, see Customizing Result Ranking with Amazon CloudSearch (p. 105).	5 November 2012
Updated the Searching, Articles & Tutorials, and Troubleshooting sections	Reorganized Searching Your Data with Amazon CloudSearch (p. 85), added the Guide to Using Elastic IPs to Manage Access to Amazon CloudSearch Domains to Amazon CloudSearch Articles and Tutorials (p. 253), and added an item about retrieving document versions to Troubleshooting Amazon CloudSearch (p. 243).	27 July 2012
Added Getting Started video and the Troubleshooting and Articles sections	A screencast of the Getting Started tutorial is now available on YouTube . The Troubleshooting Amazon CloudSearch (p. 243) provides solutions to common SDF issues, a workaround for deleting all documents from a domain, and tips for reducing document update latency. The Articles section provides a link to the new Guide to Formatting Your Data in SDF for Amazon CloudSearch available from aws.amazon.com/articles .	9 July 2012
Added how to clone a domain	You can clone an existing search domain to get an empty domain that has the same indexing options. For more information, see Cloning an Existing Domain's Indexing Options (p. 64).	25 April 2012
Initial product release	Amazon CloudSearch is introduced as a new service in Beta release.	10 April 2012