Intelligent Workload Distribution Administrator's Guide

Lucene query syntax

8/13/2025

Contents

- Administrator

Workload Manager uses Lucene queries in matching rules with a predefined syntax. Lucene query uses a syntax to parse and split the provided query string based on operators, such as AND or NOT. The query then analyzes each split text independently before returning a result of positive or negative matching.

Learn about how you can use the query to create a complex rule that includes wildcard characters, searches across single or multiple fields, and more. Though the Lucene query is versatile, the query is strict and returns an error if the query string includes any invalid syntax.

Related documentation:

-

Query syntax

The query string “mini-language” is used by the Lucene rules definitions in categories. The query string is parsed into a series of terms and operators. A term can be a single word such as *quick* or *brown*, or a phrase that is surrounded by double quotes "*quick brown*" which searches for all words in the phrase, in the same order.

Operators allow you to further customize the search.

Field names

You can specify fields to search in the query syntax. The following query options are available:

```
state: Routing
```

where the **state** field equals *Routing*.

```
title: (quick OR brown)
```

where the **title** field equals *quick* or *brown.*

```
author:"John Smith"
```

where the **author** field is equal to the exact phrase "*John Smith*".

```
first*name: Alice
```

where the **first name** field equals *Alice* (note how we need to escape the space with an asterisk to follow the rules).

```
book.*: (quick OR brown)
```

where any of the fields **book.title**, **book.content** or **book.date** equals *quick* or *brown.*

```
_exists_:title
```

where the **title** field has any non-null value.

Deviations from Lucene query of Elasticsearch

Though the library supported by Workload Manager resembles the Lucene query of Elasticsearch, there are some deviations to be noted.

- The following characters are allowed in the field name: underscore, alpha-numeric characters (a-z, A-Z, 0-9), hyphen, dot, question mark, and asterisk. The query is invalid if any other character is used.
- The query matching logic uses the whole object or relevant field to match the expression, thus it requires query matching of the whole object or field. For example, to match the word *brown* in the **Subject** field with value *quick brown fox*, you must use the query: *Subject: *brown**
    - To match the word *quick* in the **Subject** field, you can use either *Subject: quick** or *Subject: *quick**
- Character escaping is not required in the field names and support for field names is basic, unlike in Lucene query in Elasticsearch.
- All queries are case-sensitive, unlike in Lucene query in Elasticsearch.
- Fuzzy logic in queries is not supported, unlike in Lucene query in Elasticsearch.
- Tags or predefined data (for example, *tags: [alpha TO omega]*) is not supported, unlike in Lucene query in Elasticsearch.
- Boost operators (for example, *brown^2 fox*) and Boolean operators (for example, *+brown -fox*) are not supported, unlike in Lucene query in Elasticsearch.

Wildcards

Wildcard searches can be run on individual terms, using *?* to replace a single character, * to replace zero or more characters, and *\s* to replace a blank character.

Examples of wildcards:

- **brown** - matches the word brown in any part of the object or field (if specified)
- *super** - matches only if the object or field (if specified) starts with word super
- **brown\sfox** - matches 'brown fox' in any part of the object or field (if specified)
- **brown*jumps** - matches the words brown and jumps in succession, in any part of the object or field (if specified)

Regular expressions

Regular expression patterns can be embedded in the query string by wrapping them in forward-slashes (/):

```
name:/joh?n(ath[oa]n)/
```

Syntax as in JavaScript specifications is supported. Additionally, a modifier for case-insensitive match is supported as shown in the following query syntax:

```
name: /.*john.*/i
```

The above query matches with any combination of uppercase and lowercase letters in the word *John* such as **John**, **JOHn**, and **johN**. Any symbol after the second forward-slash other than *i* makes the query invalid.

Examples of regular expressions:

- *Subject: /.*product.*/i* - matches category if the word *product* is found anywhere in the subject using case-insensitive match
- *BodyText: /.*(advanced|moderate|poor).*/* - matches category if the **BodyText** field contains the word *advanced*, *moderate,* or *poor*
- *Subject: /Product\s\d{5}/* - matches category if the subject begins with *Product* and has 5 digits after a blank space, for example, **Product 12345**

Ranges

Ranges can be specified for date, numeric, or string fields. Inclusive ranges are specified with square brackets *[min TO max]* and exclusive ranges with curly brackets *{min TO max}*.

Examples of ranges

All days in 2012:

```
date:[2012-01-01 TO 2012-12-31]
```

Numbers 1 to 5:

```
count:[1 TO 5]
```

Numbers from 10 upwards:

```
count:[10 TO *]
```

Dates before 2012:

```
date:{* TO 2012-01-01}
```

Curly and square brackets can be combined. Numbers from 1 up to but not including 5:

```
count:[1 TO 5}
```

Ranges with one side unbounded can use the following syntax:

> **Important**
>
> You must not use any blank characters between colon ':', comparison operator, and number.

```
age:>10

age:>=10

age:

age:
```

To combine an upper and lower bound with the simplified syntax, you would need to join two clauses with an AND operator:

```
age:(>=10 AND
```

Grouping

Multiple terms or clauses can be grouped together using parentheses to form sub-queries:

```
title: ((*quick* OR *brown*) AND *fox*)
```

Simple grouping can be used to connect multiple choices. The following query matches object where the subject field contains the word impact, service or equals to appointment

```
Subject: *impact*|*service*|appointment
```

This type of grouping can be used only if no blank characters or quoted strings are present. If needed, you can replace such characters with *\s* symbol:

```
Subject: *impact*|*service*|appointment\srequired\stoday
```

The above query matches if subject equals phrase *appointment required today*.

Conjunction operators

The following operators are supported:

- OR or ||
- AND or &&
- NOT or !

> **Important**
> The operators AND, OR, and NOT must be in uppercase.

Example

```
Subject: (*production* OR *sales*) AND FromAddress: *@client.com|customer*@gmail.com AND NOT
BodyText: /.*spam.*/i
```

This query matches if subject contains *production* or *sales* and received from an email address ending either with *client.com* or matching *customer*:gmail.com* and not containing the word *spam* in the body of email.

Miscellaneous

The following Boolean symbols are supported:

- true
- false

Example of a Boolean query

```
present: true
```

The above query matches if the **present** field has the value *true*.

- The query *present: true* matches only if the **present** field has the Boolean value *true*. It does not match if the **present** field has a string value *“true”* and vice versa. To match both the variants, you can use the query *present: (true OR "true")*
- If a field has an integer value, matching occurs only when the rules contain an integer value, not to a string value. Similarly, a field with string value can match only to a rule with string value, not to an integer value.