Search API Overview

Learn about search options when using the Community API `/search` endpoints

This guide gives an overview of the Community API `/search` endpoint as it is used in Community API v1 and v2. The Community API provides support for text search against messages, users, images, and products.

In this article, you will:

- learn how to make search calls with API v1 and v2
- learn about the v1 and v2 search parameters
- learn how to form LiQL queries for keyword and combinatorial searches
- learn which search features are supported by v1 but not v2

Community API v1

The Community /search endpoint enables you to search for messages, users, images, and products across the community.

In addition, the Category, Board, Blog, and Group resources include methods to perform a text search on Message objects filtered to the specified node. Note that the Board resource /search endpoints are used for Forum, Q&A, Idea Exchange, Contest, and KB discussion styles.

A basic call to the `/search` endpoint looks something like the following example. The example searches for messages with the label “wireless”.

Search for messages with a specified label

```
https://[COMMUNITY DOMAIN]/restapi/vc/search/messages?q=label.text:wireless
```

/search query parameters

The API v1 `/search` methods can take the following parameters to perform text searches.

Query Parameters for Text Searches

without	search terms to exclude from the result set

You can also use these parameters to sort and group the results.

Query Parameters for Sorting and Grouping Results

Query Parameter	Description
sort_by	whether to sort by the dates of the returned messages or the dates of the messages' parent topics
collapse_discussion	whether or not to group search results by thread

q

The `q` parameter takes a text string. Text passed with this parameter is case-insensitive. Use `q` for general keyword searches. This option supports AND and OR functions, and can be filtered by field.

This example searches for messages with the label "wireless".

Search for messages with the label "wireless"

```
https://[COMMUNITY DOMAIN]/restapi/vc/search/messages?q=label.text:wireless
```

- Use `+` to search for multiple text strings in cases where all of the strings must exist within the same field. For example: `q=speaker+wireless` returns messages where both "speakers" and "wireless" appear in the field value.
- Use `OR` (case sensitive and encoded with %20 before and after) to search for multiple text strings when you want to search across multiple fields of a message. For example, `q=iPhone%20OR&20Android` returns messages where either iPhone or Android appear in a field value.
- Optionally specify a field to filter by using the format: `q=field_name:text_string`, such as `subject:stereo`

phrase must appear in the exact order passed. Encode spaces between strings with `%20`.

This example returns messages with the phrase “point and shoot”:

Text

```
https://[COMMUNITY DOMAIN]/restapi/vc/search/messages?phrase=point%20and%20shoot
```

one_or_more

Use the one_or_more parameter to search for a set of one or more search terms. Separate terms with a space. Results must match one or more of the terms given. The strings are searched for across all searchable fields and do not have to appear in a specific order. The parameter supports OR functionality, but not AND (+).

Use OR (case-sensitive and encoded with %20 before and after) to search for multiple text strings when you want to search across multiple fields of a message.

This example returns messages that contain at least one the terms “apples”, “bananas”, or “cherries” in the subject and body fields.

Search with 'one_or_more' query parameter

```
https://[COMMUNITY DOMAIN]/restapi/vc/search/messages?one_or_more=applies%20OR%20banana
```

without

Use the `without` parameter to pass search terms to exclude from the results set. The parameter filters out messages that contain the term(s) passed. Combine terms with OR to exclude multiple terms. Note that spaces between terms are interpreted as OR.

Using `without` along with other query parameters can help perform faster, more accurate queries.

sort_by

Use the `sort_by` parameter to determine whether to return the result set by the dates the individual messages were posted or the dates that the messages' respective topics were posted. Possible values are `date` and `topicpostdate`.

collapse_discussion

The `collapse_discussion` parameter returns the result set in a threaded view. When set to `true`, entire threads are searched at once and results are grouped and returned by thread (the `root_unique_id` field on the message).

> 📘 The Community v2 search API does not support `collapse_disucssion` functionality.

Access an item in an index

To get the item at a certain index for each page, chain the /index/[index] call at the end of the related `/search` method.

- `search/images/index/[index]`
- `search/messages/index/[index]`
- `search/products/index/[index]`
- `search/users/index/[index]`

Community API v2

Community API v2 uses the /search endpoint, passing a LiQL query as a string in the `q` parameter. The `q` parameter works differently in v2 than it does with the v1 /search endpoint. We'll discuss the differences later in this section.

context objects, passing the LiQL query as the value of the `q` parameter. This snippet calls the API v2 `/search` endpoint to retrieve the subject and ID from messages that contain the term “dogs” in the subject or body and assigns the result set to a variable:

Text search with FreeMarker

```
<#assign x= rest("2.0","/search?q=" + "SELECT subject, id FROM messages WHERE subject M
```

You can also call the `/search` endpoint in a GET call over HTTPS. This example performs the same search as the FreeMarker snippet above:

Text search over HTTP

```
https://[COMMUNITY DOMAIN]/api/2.0/search?q=SELECT+subject+id+FROM+messages+WHERE+subje
```

Alternatively, you can use the liql, liqladmin, and restBuilder.liql to simply make a LiQL query without referring to the `/search` endpoint. This LiQL example performs the same search as the Community API v1 and v2 `/search` endpoint.

LiQL search

```
<#assign resp = liql("SELECT subject, id FROM messages WHERE subject MATCHES ‘dogs’ OR
```

This restBuilder.liql example performs the same search as the Community API v1 and v2 `/search` endpoint.

restBuilder.liql search

```
<#assign resp = restBuilder().liql("SELECT subject, id FROM messages WHERE subject MATC
```

- a single term
- one or more terms

In Combinatorial searches with LiQL, we describe how to search for:

- a combination of terms (two or more)
- a combination of terms that includes one or more of the terms
- a combination of terms that includes exact phrases

Most search functionality provided with the API v1 `/search` endpoint can be performed with API v2 LiQL queries. The following table lists the text search query parameters in V1, and how they are supported in LiQL.

API v1 Query Parameter	LiQL Equivalent
f	The `f` parameter in API v1 becomes an intrinsic part of LiQL WHERE clause. Every text search in v2 is constrained to a field or set of fields. For example, to filter the search to the body field: `WHERE body MATCHES 'search text'`
q	In the v2 `/search` endpoint, use the `q` parameter to pass the LiQL query. The `q` parameter in v1 (the one that defines the text to analyze) maps to the MATCHES operator in the query. Look in the Constraints tables in the API v2 reference documentation for a collection to see which constraints support MATCHES. In the messages collection documentation, for example, you'll see that the

	Use OR to combine text searches on different fields. (OR works like AND in LiQL at this time). For example, to search both the `subject` and `body` fields: `WHERE body MATCHES 'wireless' OR subject MATCHES 'wireless'`
phrase	In v2, define an exact phrase wrapped in double quotes inside a string with the MATCHES operator. `WHERE subject MATCHES '"I enjoy apples"'`
one_or_more	In v2, perform a one-or-more term search with a comma-separated list of terms wrapped in parentheses. Use this when you want to return messages that have at least one of these terms in the defined fields. `WHERE subject MATCHES ('apples', 'bananas', 'cherries')`
without	This search functionality is not currently supported in LiQL.
sort_by	Sorting in v2 is done in the ORDER BY clause of a LiQL query. `ORDER BY post_time DESC`
collapse_discussion	LiQL does not support collapse_discussion functionality.

Updated about 2 years ago

← Application error codes (v1)

Federated search strategies →

Search API Overview

Learn about search options when using the Community API `/search` endpoints