# Using the Search API

ON THIS PAGE

Basic query syntax
How the query syntax works
URL encoding parameter values
Search request examples
Search using curl
Search using Python
Search using JavaScript with Node.js

You can use the Ticketing API's List Search Results endpoint to search for tickets, users, organizations, and groups in Zendesk. The endpoint uses the following request format:

```
GET /api/v2/search?query={query}
```

This article shows how you can use the endpoint to make search requests.

**Disclaimer:** Zendesk provides this article for instructional purposes only. Zendesk doesn't provide support for example code in this article. Zendesk doesn't support third-party technologies, such as curl, Python, Node.js, or JavaScript.

## Basic query syntax

The search endpoint has a `query` parameter:

```
.../api/v2/search.json?query={search_string}
```

When making a search request, you use the `query` parameter to specify a resource type, such as tickets or users. You also specify the criteria for the search, such as "users named Jane Doe" or "tickets with an open status".

This section contains example syntax for common `query` strings. For more details, see the Zendesk Support search reference in Zendesk help.

**Important:** The `query` strings in the following table must be URL encoded before use. See URL encoding parameter values.

| Task | Unencoded `query` value |
|---|---|
| Search for a specific word | `Greenbriar` |
| Search for an exact string | `"Greenbriar Corporation"` |
| Search for a ticket by id | `3245227` |
| Search for a resource type | `type:user "Jane Doe"` |
| Search by ticket status | `type:ticket status:open` |
| Search by date | `type:organization created<2099-05-01` |

## How the query syntax works

The `query` string syntax uses the following rules:

- The `:` character is the equality operator. Other operators include `>` (greater than) and `<` (less than), the minus sign `-` (negation), and the wildcard (`*`) character. See Search operators in Zendesk help
- Double quotes (`""`) are used for search phrases. The search only returns records that contain an exact match of the phrase.
- The `type` property returns records of the specified resource type. Possible values include `ticket`, `user`, `organization`, and `group`. See Using the 'type' keyword in Zendesk help
- The `status` property returns tickets assigned to a specified ticket status. Other supported properties vary based on the record type. See the following reference guides in Zendesk help:
    - Ticket property keywords
    - User property keywords
    - Organization property keywords
    - Group property keywords
- Date properties, such as `created`, `updated`, and `solved`, return records related to a specific date. The date format is YYYY-MM-DD. See Searching by date and time in Zendesk help

## URL encoding parameter values

Zendesk REST APIs require URL encoding for any URL parameter values, including the `query` parameter. URL parameters are also called query parameters or query strings.

For example, to submit a search request using a `query` value of `type:ticket status:open`, the `:` and space characters must be URL encoded as follows:

```
.../api/v2/search.json?query=type%3Aticket+status%3Aopen
```

## Search request examples

This section includes example search requests using the following tools and programming languages:

- curl
- Python
- JavaScript with Node.js

The examples in this section use API tokens. An admin must create an API token in Admin Center. See API authentication in the API reference.

For clarity, the examples don't handle pagination. To learn about paginating list results, see Paginating through lists.

### Search using curl

To include URL parameters in a curl request, use the `-G` flag. To URL-encode the parameter's value, use the `--data-urlencode` option. Example:

```
curl "https://{subdomain}.zendesk.com/api/v2/search.json" \
-G --data-urlencode "query=type:ticket status:open" \
-v -u {email_address}/token:{token}
```

You can use multiple `-G` flags to include multiple URL parameters in a single curl request. Example:

```
curl "https://{subdomain}.zendesk.com/api/v2/search.json" \
-G --data-urlencode "query=type:ticket status:open" \
-G "sort_by=created_at" \
-G "sort_order=asc" \
-v -u {email_address}/token:{token}
```

For more information about the `-G` and `--data-urlencode` flags, see the curl documentation.

## Search using Python

You can use the Requests library for Python 3 to make REST API requests. The library allows you to pass URL parameters as a Python dictionary. When it makes the request, the library automatically URL-encodes the parameter values.

The following Python 3 script uses the Requests library to make a search request.

```python
import requests

url = 'https://SUBDOMAIN.zendesk.com/api/v2/search.json'

params = {
    'query': 'type:ticket status:open',
    'sort_by': 'created_at',
    'sort_order': 'asc'
}

auth = 'EMAIL_ADDRESS/token', 'API_TOKEN'

response = requests.get(url, params=params, auth=auth)

print(response.json())
```

Before running the script, replace the following placeholders:

- "SUBDOMAIN" with your Zendesk subdomain
- "EMAIL_ADDRESS" with your Zendesk email address
- "YOUR_API_TOKEN" with your Zendesk API token

Using environment variables is a standard practice for managing sensitive data like your Zendesk API token. If you place this script in a public repository or share it with others, ensure that the API token is not included in the code. If you've accidentally exposed the token in the script, it may be compromised and should be regenerated in your Zendesk Admin Center.

## Search using JavaScript with Node.js

Node.js is a developer tool that lets you run JavaScript code outside of the browser. You can use the Axios library for Node.js to make REST API requests. The library allows you to pass URL parameters as a JavaScript object. When it makes the request, the library automatically URL-encodes the parameter values.

The following Node.js script uses the Axios library to make a search request.

```javascript
const axios = require("axios");

const url = "https://SUBDOMAIN.zendesk.com/api/v2/search.json";

const params = {
  query: "type:ticket status:open",
  sort_by: "created_at",
  sort_order: "asc",
};

const auth = {
  username: "EMAIL_ADDRESS" + "/token",
  password: "YOUR_API_TOKEN",
};

axios
  .get(url, { params, auth })
  .then((response) => {
    console.log(response.data);
  });
```

Before running the script, replace the following placeholders:

- "SUBDOMAIN" with your Zendesk subdomain
- "EMAIL_ADDRESS" with your Zendesk email address
- "YOUR_API_TOKEN" with your Zendesk API token

Using environment variables is a standard practice for managing sensitive data like your Zendesk API token. You can access environment variables in Node.js using `process.env.VARIABLE_NAME`, after setting them in your environment or in a .env file using a package like `dotenv`. See How to read environment variables from Node.js.