Working with JSON

ON THIS PAGE

The Zendesk REST API is a JSON API. If you want to send data to the API to update or create data in your Zendesk product, you need to convert it to JSON first. If you want to get data from your Zendesk product, the API will deliver it as JSON and you'll need to convert it to something you can use.

Disclaimer: Zendesk provides this article for instructional purposes only. Zendesk does not support or guarantee the code. Zendesk also can't provide support for third-party technologies such as JSON, JavaScript, Python, curl, or jq.

JSON basics

JSON is a lightweight data-interchange format that's easy for humans to read and machines to parse. Example:

```
{
  "posts": [
    {
      "id":    35467,
      "title": "How do I open the safe",
      ...
    },
    ...
  ]
}
```

JSON data typically consists of one or more named properties, such as the `"posts"` property above. The property's value in this case is a list of post objects. Each of the objects shares a set of properties, such as `"id"` , `"title"` , and others .

JSON data can be structured in many different ways. Example:

```
{
  "vote": {
    "id": 35467,
    "user_id": 888887,
    "value": -1,
    ...
  }
}
```

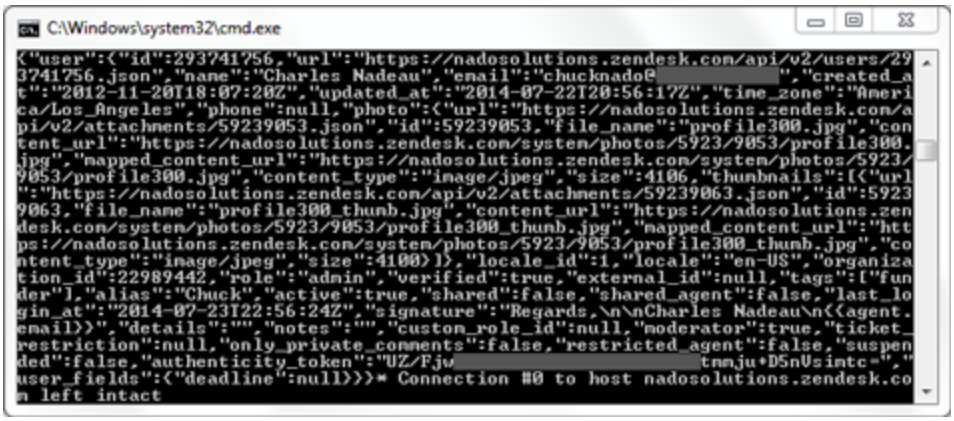
See the API documentation for the various API endpoints to learn the structure of the JSON data they return.

To learn more about JSON, see Introducing JSON on the json.org site.

Converting JSON to data you can understand

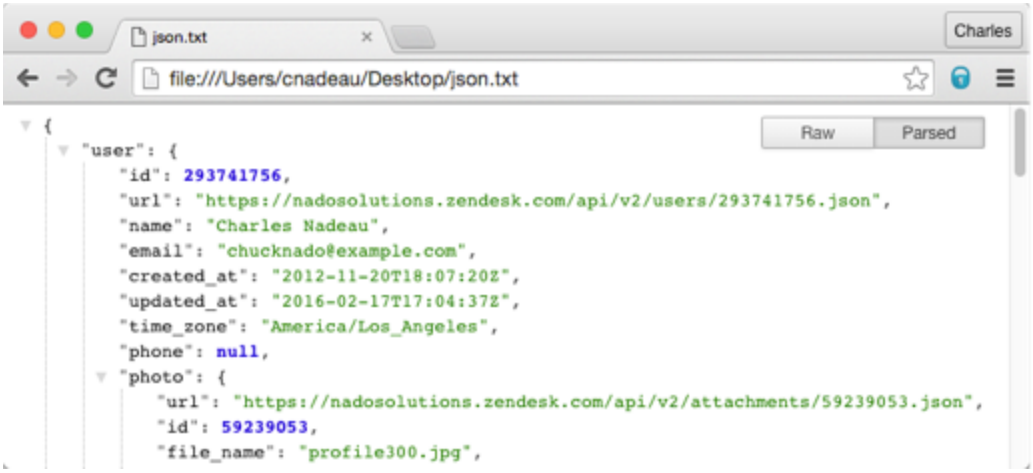
The API returns JSON on a single line. If it only has a few properties, you can read it easily enough. Pasting it into a text editor and manually inserting line breaks helps.

If the JSON contains multiple objects with lots of properties, the information becomes more challenging to read:



One solution is to pretty print the information. A Google search returns a number of solutions. The simplest in terms of setup is the "JSON Formatter" extension for Google Chrome. You can get it from https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa?hl=en . After installing it, in Chrome select **Windows** > **Extensions** , and select the "Allow access to file URLs" option for the JSON Formatter extension. This enables pretty printing local files.

Paste your JSON results to a text file and save it with a **.txt** extension. Then open the file in Chrome (**File** > **Open File**).



Converting JSON to data your code understands

If you want to use the JSON returned by the API in your code, you have to convert it into a data structure your code understands. Most programming languages have tools to help.

This article covers JavaScript and Python. If you don't use these languages, a quick search in your language's documentation should uncover equivalent tools.

JavaScript

Use the `JSON.parse()` method in JavaScript to convert JSON strings into JavaScript objects.

To give the method a test run, paste the following snippet in your browser's JavaScript console and press Enter:

```
var json_string = '{"article":{"title":"Great Expectations"}}';
var js_object = JSON.parse(json_string);
console.log(typeof js_object);
console.log(js_object.article.title);
```

Note : To access the console in Chrome, select **View** > **Developer** > **JavaScript Console** . In Firefox, select **Tools** > **Web Developer** > **Web Console** , and look for the entry field at the bottom of the console. If you get a scam warning when pasting the snippet, type "allow pasting" in the field and try again.

The script should print the following lines in the console:

```
object
Great Expectations
```

The first line tells you the data is now contained in a JavaScript object. The second line displays the value of the `article.title` property.

To learn more, see JSON.parse() in the MDN documentation.

Python

Python's built-in **json** library converts JSON strings or files into Python data structures.

Use `json.loads()` to convert JSON strings:

```
import json

json_string = '{"article":{"title":"Great Expectations"}}'
python_data = json.loads(json_string)
print(type(python_data))
print(python_data['article']['title'])
```

Running this code in a command line interface should print the following lines:

```
<class 'dict'>
Great Expectations
```

The first line tells you the data is now contained in a Python dictionary. The second line displays the value of the `title` property.

Use `json.load()` (singular) to convert files that contain JSON:

```
import json
with open('json.txt', mode='r', encoding='utf-8') as f:
    python_data = json.load(f)
print(type(python_data))
```

To learn more about these methods, see the json module documentation on the Python website.

To learn how to use Python to make API requests that return JSON data, see Making requests to the Ticketing API.

If you use the popular Requests package , you can use the `response` object's `json()` method to convert the JSON returned in HTTP responses. Example:

```
import requests
# ...
response = requests.get(url, credentials)
python_data = response.json()
```

You can also use the `json.loads()` method with the Requests library, but make sure to pass only the response content, not the whole response. HTTP responses contain other information such as headers that will cause errors. In the Requests API, the response content is contained in the the `response` object's `text` property. Example:

```
import json
import requests
# ...
response = requests.get(url, credentials)
python_data = json.loads(response.text)
```

See the response object API doc on the Requests website for more information.

Converting data in your code to JSON

If you want to use data in your code to update or create data in a Zendesk product, you need to convert it to JSON before sending it with the API request.

This article covers JavaScript and Python. If you don't use these languages, you should find the equivalent in your language's documentation.

JavaScript

Use the `JSON.stringify()` method in JavaScript to convert JavaScript objects to JSON strings.

To give the method a test run, paste the following snippet in your browser's JavaScript console and press Enter:

```
var js_object = {ticket: {comment: {html_body:
'<p style="color: red;">Review the settings.</p>' }}};
var json_string = JSON.stringify(js_object);
console.log(typeof json_string);
console.log(json_string);
```

Note : To access the console in Chrome, select **View** > **Developer** > **JavaScript Console** . In Firefox, select **Tools** > **Web Developer** > **Web Console** , and look for the entry field at the bottom of the console. If you get a scam warning when pasting the snippet, type "allow pasting" in the field and try again.

The script should print the following lines in the console:

```
string
{"ticket":{"comment":{"html_body":"<p style=\"color: red;\"
>Review the settings.</p>"}}}
```

The first line tells you the data is now a string and the second line displays the JSON.

To learn more, see JSON.stringify() in the MDN documentation.

Python

Use the `json.dumps()` method in the Python **json** library to convert Python data into JSON. Example:

```
import json

python_data = {'ticket': {'comment': {'html_body':
'<p style="color: red;">Review the settings.</p>'}}}
json_string = json.dumps(python_data)
print(type(json_string))
print(json_string)
```

Running this code in a command line interface should print the following lines:

```
<class 'str'>
{"ticket": {"comment": {"html_body":"<p style=\"color: red;\"
>Review the settings.</p>"}}}
```

The first line tells you the data is now a string and the second line displays the JSON.

To learn more about the `json.dumps()` method, see the json module documentation on the Python website.

To learn how to use Python to make API requests containing JSON data, see Making requests to the Ticketing API.

Pretty printing the JSON

You can also use `json.dumps()` to pretty print the JSON. Simply add an argument named `indent` to the `json.dumps()` method, as follows:

```
json_string = json.dumps(python_data, indent=2)
```

The example should now print:

```
{
  "ticket": {
    "comment": {
      "body": "The smoke is very colorful."
    }
  }
}
```

Writing the JSON to a file

You can write the JSON to a new file named **json.txt** with the following lines:

```
with open('json.txt', mode='w', encoding='utf-8') as f:
    f.write(json_string)
```

Using JSON in curl statements

You can use the JSON output produced in by the various methods in Converting data in your code to JSON above and paste it directly in curl statements to update and create data in a Zendesk product. Example:

```
curl https://{subdomain}.zendesk.com/api/v2/tickets/{id}.json \
  -d '{"ticket":{"comment":{"body":"The smoke is very colorful."}}}' \
  -H "Content-Type: application/json" \
```

```
    -v -u {email_address}/token:{api_token} \
    -X PUT
```

The curl statement includes the JSON data for adding a ticket comment (the **-d** flag stands for data). To learn more about curl, see Installing and using curl . Windows users: See also Using curl in Windows .

If the JSON data is too lengthy to fit comfortably in a curl statement, you can move it to a file and then import the file in the curl statement using the `@filename` syntax. Here's how:

1. Create a file named **json.txt** (or something along those lines) and move the JSON to the file. Example:

   ```
   {"ticket":{"comment":{"body":"The smoke is very colorful."}}}
   ```

   Python will also let you write JSON directly to a file. See the instructions for Python above.
2. Change the curl statement to import the JSON data with the `@filename` syntax:

   ```
   curl https://{subdomain}.zendesk.com/api/v2/tickets/{id}.json \
       -d @json.txt \
       -H "Content-Type: application/json" \
       -v -u {email_address}/token:{api_token} \
       -X PUT
   ```

   Note : In Windows, replace the line-ending backslash characters with caret (^) characters.
3. Before running the statement, use the **cd** command (for change directory) to navigate to the folder that contains the file. Example:

   ```
   $ cd json_files
   ```

4. Run the curl statement.

Parsing JSON responses using curl and jq

jq is a command-line tool for parsing and modifying JSON data. You can use jq to pretty print and extract data from JSON responses to your curl requests.

To install jq, check out the Download page of the jq site.

This section covers the basics of using jq. For complete documentation, see the jq manual.

Pretty printing JSON responses using jq

By default, curl outputs JSON responses from Zendesk APIs on a single line. You can use jq to pretty print the response into a more human-readable format.

For example, the following curl request retrieves a ticket object from the Ticketing API's Show Ticket endpoint.

```
curl https://{subdomain}.zendesk.com/api/v2/tickets/{id}.json \
  -u {email_address}/token:{api_token}
```

Without jq, the request's output looks like this:

```
{ "ticket": { "assignee_id": 123456, "collaborator_ids": [ 35334, 234 ],
"created_at": "2099-07-20T22:55:29Z", ... } }
```

jq uses pipes (|) and dot notation to access data in the JSON response. To pretty print the entire response, append `| jq '.'` to the command.

```
curl https://{subdomain}.zendesk.com/api/v2/tickets/{id}.json \
  -u {email_address}/token:{api_token} | jq '.'
```

The request's output now looks like this:

```
{
  "ticket": {
    "assignee_id": 123456,
    "collaborator_ids": [
      35334,
      234
    ],
    "created_at": "2099-07-20T22:55:29Z",
    ...
  }
}
```

Extracting JSON data using jq

You can also use jq to extract specific properties from a JSON response. Use pipes and dot notation to specify the property. The following example uses jq to extract the ticket object's `assignee_id` property.

```
curl https://{subdomain}.zendesk.com/api/v2/tickets/{id}.json \
  -u {email_address}/token:{api_token} | jq '.ticket | .assignee_id'
```

The output looks like this:

```
123456
```

Use commas to separate multiple properties. The following example uses jq to extract the `assignee_id` and `created_at` properties.

```
curl https://{subdomain}.zendesk.com/api/v2/tickets/{id}.json \
  -u {email_address}/token:{api_token} | jq '.ticket | .assignee_id, .created_at'
```

The output looks like this:

```
123456
"2099-07-20T22:55:29Z"
```
