Using the Ticketing API

ON THIS PAGE

You can use the Zendesk REST API to read, update, and create data in a Zendesk product. This tutorial shows you how to start working with the API. It provides examples of completing the following common tasks with the API:

- Getting data from your Zendesk product
- Updating data in your Zendesk product
- Creating data in your Zendesk product

Make sure you enable token access to the API in Admin Center under **Apps and integrations** > **APIs** > **Zendesk APIs**. For more information, see Using the API dashboard.

The article uses Python because its syntax is relatively clear and readable. If you work in another language or if you're just getting started, you should still be able to follow along.

If you just want to follow along with the tutorial and don't want to try out the requests yourself, you can skip ahead.

If you want to try out the requests yourself, you'll need version 3 of Python. To install it on your system, see http://www.python.org/download/.

Also download and install the Requests library if you don't already have it. The Requests library greatly simplifies making HTTP requests in Python. To install it, run the following command in the Terminal on the

Mac or the command prompt in Windows:

```
$ pip3 install requests
```

Note: The dollar sign ($) represents the command prompt. Don't enter it.

If you have Python 3.3 or earlier, see these instructions to install the library. Then use `pip` instead of `pip3` on the command line.

Finally, when copying the examples in this tutorial, make sure to indent lines exactly as shown. Indentation matters in Python.

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 Python.

Getting data from your Zendesk product

The following example gets all the groups in a Zendesk Support instance using the groups API.

Make sure you've defined a few groups in your Zendesk Support instance before trying out the script. See Creating, managing, and using groups. To learn how the script works, see the inline comments (which start with # in Python) as well as the explanations that follow.

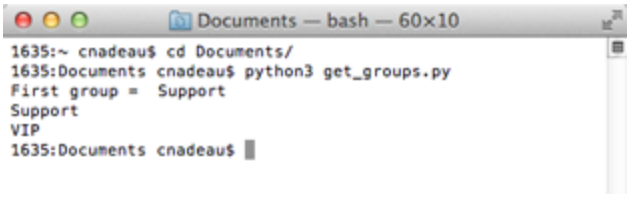
```
import requests
import os

ZENDESK_API_TOKEN = os.getenv('ZENDESK_API_TOKEN')  # Make sure this is
 correctly set in your environment
ZENDESK_SUBDOMAIN = 'your_subdomain.zendesk.com'  # Replace with your Zendesk
 subdomain
ZENDESK_USER_MAIL = 'you_zendesk_email' # Replace with the Zendesk
 email address used to access the subdomain

# Check if ZENDESK_API_TOKEN was correctly retrieved from environment
if not ZENDESK_API_TOKEN:
    print('ZENDESK_API_TOKEN environment variable is not set. Exiting.')
    exit()

url = f'https://{ZENDESK_SUBDOMAIN}/api/v2/groups.json'

auth = f'{ZENDESK_USER_EMAIL}/token', ZENDESK_API_TOKEN

# Perform the HTTP GET request
response = requests.get(url, auth=auth)

# Check for HTTP codes other than 200
```

```
21  if response.status_code != 200:
22      print('Status:', response.status_code, 'Problem with the request. Exiting.')
23      exit()
24
25  # Decode the JSON response into a dictionary and use the data
26  data = response.json()
27
28  # Example 1: Print the name of the first group in the list
29  print('First group = ', data['groups'][0]['name'])
30
31  # Example 2: Print the name of each group in the list
32  group_list = data['groups']
33  for group in group_list:
34      print(group['name'])
```

Save the script in a folder, use the command line to navigate to the folder, and run the script from the command line. Example:

```
1  python3 get_groups.py
```

Example:



How it works

The script uses the following URL to make the API call:

```
1  url = f'https://{ZENDESK_SUBDOMAIN}/api/v2/groups.json'
```

See Listing groups for details about the API.

The script uses the requests library to authenticate and make the HTTP get request:

```
1  response = requests.get(url, auth=auth)
```

Next, the script decodes the JSON returned by the API and packages the data in a Python dictionary:

```
data = response.json()
```

A dictionary is simply a collection of key/value pairs. Decoding the JSON into a dictionary lets you work with the data using regular Python operators and expressions.

Consult the Zendesk REST API docs to figure out what's in the dictionary. For example, according to the List Groups doc, the JSON returned by a call to the API has the following structure:

Example Response

```
Status: 200 OK

{
  "groups": [
    {
      "name":       "DJs",
      "created_at": "2009-05-13T00:07:08Z",
      "updated_at": "2011-07-22T00:11:12Z",
      "id":         211
    },
    {
      "name":       "MCs",
      "created_at": "2009-08-26T00:07:08Z",
      "updated_at": "2010-05-13T00:07:08Z",
      "id":         122
    }
  ]
}
```

Decoding this JSON produces a Python dictionary consisting of one key named 'groups'. The square brackets in the doc tell you the value of **groups** is a list. Each item in the list is a dictionary of group properties. Armed with this information, you can access the data in the dictionary. For example, the following statement accesses and prints the name of the first group in the dictionary:

```
print('First group = ', data['groups'][0]['name'])
```

The following statement iterates through all the groups in the dictionary and prints the name of each group:

```
group_list = data['groups']
for group in group_list:
    print(group['name'])
```

You can also write the data to a file. Replace the previous example in the script with the following snippet:

```
group_list = data['groups']
output = ''

for group in group_list:
    output += group['name'] + '\n'  # add each name to the output variable

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

```
        f.write(output)

print("Done.")
```

The snippet creates a file named groups.txt in the same folder as the script, and writes the group names in a column in the file.

The script described in this section is fine for getting up to two dozen or so records from your Zendesk. However, to retrieve several hundred or several thousand records, the script has to be modified to perform a few more tasks. To learn how, see Getting large data sets with the Zendesk API and Python.

To explore getting other kinds of data from your Zendesk product, see the rest of the Zendesk REST API docs.

Updating data in your Zendesk product

The following example adds a comment to a ticket in Zendesk Support using the tickets API.

To learn how the script works, see the inline comments as well as the explanations that follow.

```
import json
import requests
import os

# Ticket to update
ticket_id = '13'
body = 'Thanks for choosing Acme Jet Motors.'

# Package the data in a dictionary matching the expected JSON
data = {'ticket': {'comment': {'body': body}}}

# Encode the data to create a JSON payload
payload = json.dumps(data)

# Set the request parameters
zendesk_subdomain = 'your_subdomain'  # Replace this placeholder
url = f'https://{zendesk_subdomain}.zendesk.com/api/v2/tickets/{ticket_id}.json'

api_token = os.getenv('ZENDESK_API_TOKEN')  # Ensure this env variable is set
user_email = 'your_zendesk_email'  # Replace with the Zendesk
 email address used to access the subdomain

auth = f'{ZENDESK_USER_EMAIL}/token', ZENDESK_API_TOKEN

# Perform the HTTP PUT request
response = requests.put(url, data=payload, auth=auth)
```

```
25
26  # Check for HTTP codes other than 200 (OK) or 201 (Created)
27  if response.status_code not in [200, 201]:
28      print(f'Status: {response.status_code}, Problem with the request. Exiting.')
29      exit()
30
31  # Report success
32  print(f'Successfully added comment to ticket #{ticket_id}')
```

Save the script in a folder, use the command line to navigate to the folder, and run the script from the command line. Example:

```
1  python3 put_comment.py
```

Open the ticket in Zendesk Support to view the new comment.

How it works

In addition to importing the requests library, the script imports a library called json:

```
1  import json
```

You'll use the library to convert data in your script into JSON for the put request. The json library is a standard Python library. You don't need to download and install it.

Next, the script packages the data in a Python dictionary matching the structure of the JSON expected by the API. Consult the REST API doc for the expected JSON. For example, if you want to add a comment to a ticket, the API expects the following JSON:

```
1  {
2    "ticket": {
3      "comment": {
4        "body": "New comment."
5      }
6    }
7  }
```

Accordingly in Python, package your data as nested dictionaries matching the JSON:

```
1  data = { 'ticket': { 'comment': { 'body': body } } }
```

Encode the data to create a JSON payload:

```
payload = json.dumps(data)
```

If you print the payload variable, you'll get the following result:

```
{"ticket":{"comment":{"body":"Thanks for choosing Acme Jet Motors."}}}
```

You should always encode your data to prevent characters like quotes from breaking the JSON. For example, the quotes in the following body would prematurely end the string and cause an error:

```
"body": "Learn <a href="faq">more</a>."
```

Encoding the data escapes the quotes. Example: "Learn <a href=\"faq.html\">more."

Next, set the request parameters. The following URL is used to make the API call:

```
url = f'https://{zendesk_subdomain}.zendesk.com/api/v2/tickets/{ticket_id}.json'
```

See Updating tickets for details about the API.

Pass the JSON payload to the put request, along with the other request parameters:

```
response = requests.put(url, data=payload, auth=auth)
```

Tip: If you want to make repeated API calls in your script -- for example, to update a collection of tickets -- you can create a requests session and persist certain parameters across the requests. For example, replace the put request above with the following snippet, which creates and configures a session:

```
session = requests.Session()
session.auth = (user, pwd)
session.headers = headers

# make repeated requests with session.put(), not requests.put()
... response = session.put(url, data=payload)
```

To explore updating other kinds of data in your Zendesk product, see the rest of the Zendesk REST API docs.

Creating data

The following example creates a ticket using the tickets API.

Creating things with the API is almost identical to updating things, except that you use a post request instead of a put request.

```
import json
import requests
import os

# New ticket info
subject = 'My printer is on fire!'
body = 'The smoke is very colorful.'

# Package the data in a dictionary matching the expected JSON
data = {'ticket': {'subject': subject, 'comment': {'body': body}}}

# Encode the data to create a JSON payload
payload = json.dumps(data)

# Set the request parameters
zendesk_subdomain = 'your_subdomain'  # Replace this placeholder
url = f'https://{zendesk_subdomain}.zendesk.com/api/v2/tickets.json'
api_token = os.getenv('ZENDESK_API_TOKEN')  # Ensure this env variable is set
user_email = 'your_zendesk_email'  # Replace with the Zendesk
 email address used to access the subdomain

auth = f'{user_email}/token', api_token

# Do the HTTP post request
response = requests.post(url, data=payload, auth=auth)

# Check for HTTP codes other than 201 (Created)
if response.status_code != 201:
    print('Status:', response.status_code, 'Problem with the request. Exiting.')

    exit()

# Report success
print('Successfully created the ticket.')
```

Save the script in a folder, use the command line to navigate to the folder, and run the script from the command line. Example:

```
python3 post_ticket.py
```

Go to the Unassigned Tickets view in Zendesk Support to see the new ticket.

How it works

The script uses the following URL to make the API call:

```
1   url = 'https://your_subdomain.zendesk.com/api/v2/tickets.json'
```

See Creating tickets for details about the API.

To learn how the script works, see the explanations in Updating data in your Zendesk product. They're basically the same, except for the post request.

To explore adding other kinds of data in your Zendesk product, see the rest of the Zendesk REST API docs.
