Exporting a ticket view to a CSV file

ON THIS PAGE

- Get the view ID
- Get ticket data from the API
- Select and format the data
- Code complete

This tutorial shows you how to write a Python script to get ticket data in a ticket view, then export and format the data to a CSV file. For more information about views, see Creating views to manage ticket workflows.

You don't have to use the API to export ticket views to a CSV file. You can use the export view feature in the Zendesk admin interface. See Exporting a view to a CSV file. Each option has its strengths and weaknesses. For example, the export data feature in Zendesk only includes a limited set of columns in the CSV: ticket ID, status, subject, requester, requested date, type, and priority. The API exports all the ticket data, including conversations. Unfortunately, the data isn't in a neat 2-dimensional structure that can be easily written to CSV. You'll need to munge the data to fit in the simple columns and rows of a CSV table. Munging is the process of converting, or mapping, data from one format to another.

Before getting started, you'll need to install Python 3 and the Requests library, which simplifies making HTTP requests in Python.

Related information:

- Getting large data sets with the Zendesk API and Python
- Writing large data sets to Excel with Python and pandas

A note about the code examples: Some lines of code in the examples may wrap to the next line because of the article's page width. When copying the examples in this tutorial, ignore the line wrapping. Line breaks matter in Python.

Get the view ID

In this example, you want to retrieve ticket data listed under a ticket view named "Feature requests".

The first step is finding the view id for the ticket view by making an API request to the List Views endpoint. You can use an API testing tool such as curl or Postman to make the request.

Example curl request:

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

In the response, the views object array contains a `title` named "Feature requests" and the view `id`. Example:

```
{
 "count": 2,
 "next_page": null,
 "previous_page": null,
 "views": [
   {
     "active": true,
     "conditions": {},
     "description": "View for recent tickets",
     "execution": {},
     "id": 25,
     "position": 3,
     "restriction": {},
     "title": "Tickets updated less than 12 Hours"
   },
   {
     "active": false,
     "conditions": {},
     "description": "View for tickets about feature requests",
     "execution": {},
     "id": 23,
     "position": 7,
     "restriction": {},
     "title": "Feature requests"
   }
 ]
}
```

Get ticket data from the API

Now that you have the view ID, you can retrieve the list of tickets using the List Tickets from a View endpoint.

In your favorite text editor, create a file named **export_tickets.py** and paste the following code:

```
import requests
import csv
import os

# Settings
ZENDESK_API_TOKEN = os.getenv('ZENDESK_API_TOKEN')  # Load the API token from an environment variable for security
ZENDESK_USER_EMAIL = 'your_email'
ZENDESK_SUBDOMAIN = 'your_subdomain'

view_tickets = []
view_id = 'your_view_id'

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

print(f'Getting tickets from view ID {view_id}')
url = f'{ZENDESK_SUBDOMAIN}/api/v2/views/{view_id}/tickets.json'
while url:
    response = requests.get(url, auth=auth)
    if response.status_code == 200:
        page_data = response.json()
        tickets = page_data['tickets']
        view_tickets.extend(tickets)
        url = page_data['next_page']
    else:
        print(f"Failed to retrieve tickets: {response.reason}")
        url = None  # Exit loop on failure
```

Replace the following placeholders in the script with your information: your_subdomain, your_email_address, and your_view_id.

In the script:

- You import the Requests module to make API calls
- The `print()` function prints the message to the screen that it is retrieving tickets from a view when the script is executed
- The List Tickets from a View endpoint URL is specified
- To paginate through all the results, a while loop is used to make an API request and store the page data incrementally in a tickets variable, then get the 'next_page' url. You can learn more about this in Getting large data sets with the Zendesk API and Python.

Select and format the data

The data includes a lot of information you don't need such as custom fields. For each record, you only want to include the following properties for each ticket: `id`, `subject`, `requester_id`, `assignee_id`, `created_at`, and `status`.

You can use the CSV library to select the ticket properties you want to include in the CSV file and format it.

1. In **export_tickets.py**, paste the following code at the end of the file:

```
rows = [('Ticket ID', 'Subject', 'Requester ID',
         'Assignee ID', 'Created', 'Status', 'URL')]

for ticket in view_tickets:
    row = (
        ticket['id'],
        ticket['subject'],
        ticket['requester_id'],
        ticket['assignee_id'],
        ticket['created_at'],
        ticket['status'],
       f'https://{ZENDESK_SUBDOMAIN}.zendesk.com/agent/tickets/{ticket["id"]}'

    )
    rows.append(row)

with open('tickets.csv', mode='w', newline='') as csv_file:
    report_writer = csv.writer(csv_file, dialect='excel')
    for row in rows:
        report_writer.writerow(row)
```

2. Save the file and run the script.

The first line of code creates a header row for the ticket properties you want to include. Next, a `for` loop is used to iterate through the ticket properties for each ticket as a well as create a ticket URL.

The with statement opens **tickets.csv** and the `csv.writer()` function is used to create a `report_writer` object. The `for` loop uses the `report_writer.writerow()` function to write the ticket properties for each ticket as rows in the **tickets.csv** file.

Code complete

Your script should look something like this:

```
import requests
```

```
import csv
import os

# Settings
ZENDESK_API_TOKEN = os.getenv('ZENDESK_API_TOKEN')  # Load the API token from an environment variable for security
ZENDESK_USER_EMAIL = 'your_email'
ZENDESK_SUBDOMAIN = 'your_subdomain'

view_tickets = []
view_id = 'your_view_id'  # Your view ID

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

# List tickets from a View
print(f'Getting tickets from view ID {view_id}')
url = f'{ZENDESK_SUBDOMAIN}/api/v2/views/{view_id}/tickets.json'
while url:
    response = requests.get(url, auth=auth)
    if response.status_code == 200:
        page_data = response.json()
        tickets = page_data['tickets']  # Extract the "tickets" list from the page
        view_tickets.extend(tickets)
        url = page_data['next_page']
    else:
        print(f"Failed to retrieve tickets: {response.reason}")
        url = None  # Exit loop on failure

# Initializing rows with an initial header row
rows = [('Ticket ID', 'Subject', 'Requester ID', 'Assignee ID', 'Created', 'Status', 'URL')]

# Define a row per ticket and append
for ticket in view_tickets:
    row = (
        ticket['id'],
        ticket['subject'],
        ticket['requester_id'],
        ticket['assignee_id'],
        ticket['created_at'],
        ticket['status'],
        f'https://{ZENDESK_SUBDOMAIN}.zendesk.com/agent/tickets/{ticket["id"]}'
    )
    rows.append(row)

with open('tickets.csv', mode='w', newline='') as csv_file:
    report_writer = csv.writer(csv_file, dialect='excel')
    for row in rows:
```

```
48          report_writer.writerow(row)
```