SCANTOMARK

Canvas API User Guide

Disclaimer This user guide is provided on an "as is" basis for information only. This user guide and its associated website (scantomark.com) carry absolutely NO WARRANTIES of any kind, either express or implied.

SCANTOMARK.COM
E-mail: techsupport@scantomark.com

Contents

Overview

This guide provides an example of how to upload exam scores and send forms back to individual students on Canvas, a learning management system (LMS).[1] Here we use the cloud version hosted by Instructure as a running example.[2]

Note that this guide is not meant to be a comprehensive guide for using Canvas; instead, it provide "just enough" information for interested faculty to handle data and documents more efficiently on the platform.

Prerequisites

Some familiarity with the following topics will be helpful for using this guide:

- Python programming language, and
- RESTful framework for handing web requests and responses.

[1] https://github.com/instructure/canvas-lms/wiki
[2] https://www.instructure.com/canvas/login

Chapter 1

Canvas Basics for Instructors

1.1 General System Structure

When logging onto the Canvas system, instructors usually see a list of courses. From there they can navigate into a course and manage its content. All contents, such as courses, assignments, and submissions, are organized in a directory tree of many levels. An example is shown below.

```
https://yourschool.instructure.com/courses/98765/assignments/12345
```

The above example is an "assignment URL", in which `98765` is the course ID number and `12345` is the assignment ID number. Accordingly, the course URL is:

```
https://yourschool.instructure.com/courses/98765
```

ID numbers are always required whenever you try to access course content by using Canvas REST API, which has two more levels in the directory tree:

```
https://yourschool.instructure.com/api/v1/courses/98765
```

1.2 Authentication

Canvas REST API is controlled by token authentication rather than the typical security mechanism using user ID and password. So the first step is to generate an API key or access token (which is basically a long string of random characters).

Once you log onto the Canvas system, go to:

Account
↳Settings
↳Approved Integration
↳+ New Access Token

Leave the expiration fields blank for no expiration is you expect to use the token very often. Click on `Generate Token` and a long string will be created for you. You will need this token to access course content by using API.

Note that the token is on the per-user basis, i.e., you will use the same token for all of your courses.

1.3 Student ID Numbers

Similar to all course content, students has unique ID numbers on Canvas, but the numbers may not be the same as whatever numbers used by schools in what is called "SIS" (student information system).

The best way to get student ID numbers on Canvas is to export grades into CSV. Below is an example.

Student	ID	SIS User ID	SIS Login ID	Section	Exam (123456789)
Points Possible					300
Last Name, First Name	123456	987654321	987654321	Spring 2025	270

In the above example, it is the `ID` field (`123456`) that is used for identifying students. The other two fields `SIS User ID` and `SIS Login ID` are generally useless for Canvas REST API. In the heading of the field `Exam (123456789)` `123456789` is the ID number of the "assignment" called "Exam".

It is, however, more convenient for students to use their SIS IDs (sometimes called campus-wide IDs) rather than Canvas IDs on their exam forms. One particular reason is that Canvas IDs are not used by students or instructors for login purposes and as such are not easy to find.

Chapter 2

Send Custom Documents to Individual Students

In this user guide we focus on only one method of sending custom documents to individual students: sending the document as an attachment to a comment an instructor makes on a student's submission for an assignment. There are a few assumptions:

1. Campus-wide SIS IDs (rather than Canvas user IDs) used on exam forms.
2. An assignment (exam) has been created on Canvas.
3. All exam files (all PDFs, exam scores, and Python code) are within the same folder (see below for an example, which is basically the content of the zip file you would get if the grading is successfully completed plus the Python code file).

```
ExamFolder
└──→987654321-A-10-95-some-random-test.pdf #individual student form
└──→scores.csv #all scores
└──→exam-all-forms-graded.pdf #all graded forms in one file
└──→answer-key.csv #original answer keys you uploaded
└──→exam-all-forms-original.pdf #the original forms you uploaded
└──→attach_file_to_assignment_comments.py #the Python code
```

In this example, you would navigate to the directory `ExamFolder` in a terminal and then run

```Py
python3 attach_file_to_assignment_comments.py
```

The Python file (module) is not meant to be "installed"; rather, it is supposed to be "created" or "copied" for each exam. It is designed to do a single task and as such it is far from a complete package. The content of the Python file is explained below section by section but everything should be in the same file.

2.1 Required Modules

The following modules are standard ones and should be available with default installation.

```Py
import requests
import json
import os
import re
import csv
```

2.2 Initial Configuration

Two pieces of information are required for initial configuration:

1. Your access token. Generate one on Canvas and copy/paste to `access_token`.

2. The subdomain name of your school, which is supposed to replace `yourschool` but keep everything else.

```Py
# Configuration
access_token = 'generate token on canvas and copy here'
base_url = 'https://yourschool.instructure.com/api/v1'

# Headers for authentication
headers = {
    'Authorization': f'Bearer {access_token}',
    'Content-Type': 'application/json'
}
```

2.3 Matching SIS IDs with Canvas User IDs

Assuming you use campus-wide or SIS IDs (called "CWID" throughout this guide) on the exam form, it is necessary to match them with Canvas user IDs. The following function returns a dictionary `{cwid: user_id}`. The three inputs are:

1. `csvfile`: the name of the CSV file that hold the matching of CWID and Canvas user ID.
2. `cwid_col`: the heading of the CWID column in the CSV file.
3. `user_id_col`: the heading of the Canvas user ID column in the CSV file.

```Py
def get_user_ids(csvfile, cwid_col, user_id_col):
    stu_dict = {}
    with open(csvfile, mode='r') as file:
        csv_reader = csv.DictReader(file)
        stu_dict = {row.get(cwid_col): row.get(user_id_col) for
   row in csv_reader}
        # cwid: user_id
    return stu_dict
```

2.4 Search for Student PDF Files

The following function is used to search a directory for student exam forms based on their IDs marked on CWIDs. Marked student forms are named with IDs, exam version, scores, and some random text.

```Py
def find_files(directory, regex_pattern):
    matching_files = []
    regex = re.compile(regex_pattern)
    for root, dirs, files in os.walk(directory):
        for file in files:
            if regex.search(file):
                matching_files.append(os.path.join(root, file))
    return matching_files
```

2.5 Upload File

Before you can "send" a file to any student, you must upload the file to Canvas first and obtain a file ID number. Then you add a comment and use that file as attachment. The following function does the first step—upload a file to a student's submission and get an ID number. Notice the long URL. There is a "folder" of submissions although a typical exam students do not submit anything on Canvas.

```Py
def upload_file(course_id, assignment_id, user_id, file_path):
    # Step 1: Start the file upload process
    url = f"{base_url}/courses/{course_id}/assignments/{assignment_id}/submissions/{user_id}/comments/files"
    # Initiate file upload
    params = {
        'name': file_path.split('/')[-1],
        'size': str(os.path.getsize(file_path)),
        'content_type': 'application/octet-stream',
        'parent_folder_path': '/'
    }
    response = requests.post(url, headers=headers, params=params)
    if response.status_code != 200:
        print(f"Failed to initiate file upload: {response.status_code}")
        print(response.json())
        return None
    # Step 2: Upload the file
    upload_url = response.json()['upload_url']
    upload_params = response.json()['upload_params']
    with open(file_path, 'rb') as f:
        files = {'file': f}
        upload_response = requests.post(upload_url, data=upload_params, files=files)
    if upload_response.status_code not in [200, 201]:
        print(f"Failed to upload file: {upload_response.status_code}")
        print(upload_response.json())
        return None
    # Step 3: Confirm the upload and get the file ID
    file_info = upload_response.json()
    file_id = file_info['id']
    return file_id
```

2.6 Attach File to Comment

The following function is to attach a file (using the filed ID from the previous function) to a comment.

```Py
def attach_file_to_comment(course_id, assignment_id, user_id, file_id, comment_text):
    # Add the file to the assignment comment
    url = f"{base_url}/courses/{course_id}/assignments/{assignment_id}/submissions/{user_id}"
    payload = {
        "comment": {
            "text_comment": comment_text,
            "file_ids": [file_id]
        }
    }
    response = requests.put(url, headers=headers, data=json.dumps(payload))
    if response.status_code in [200, 201]:
        print("File attached to comment successfully!")
        #print(response.json())
    else:
        print(f"Failed to attach file to comment: {response.status_code}")
        print(response.json())
```

2.7 Settings for Actual Exams

Once you finish the grading and collect required files and ID numbers, you can now run the code to send graded exam forms to students individually.

```Py
#####Change accordingly
csvfile = 'students.csv'
cwid_col = 'CWID'
user_id_col = 'ID'
course_id = '987654'
assignment_id = '123456'
comment_text = 'Your exam xyz Scantron form attached.'
#####
stu_dict = get_user_ids(csvfile, cwid_col, user_id_col)
for cwid, user_id in stu_dict.items():
    pattern = f'{cwid}.+pdf'
    student_files = find_files('./', pattern)
    print(cwid, user_id, student_files)
    if student_files:
        for stu_f in student_files:
            file_id = upload_file(course_id, assignment_id,
   user_id, stu_f)
            if file_id:
                attach_file_to_comment(course_id, assignment_id,
   user_id, file_id, comment_text)
```

Make sure to change the settings based on your own exam.

To summarize, if you copy/paste the code shown in this guide, you only need to change the following variables:

- `access_token` which you can generate and get from your Canvas account.
- `base_url` make sure to use your school's subdomain on instructure.com.
- `csvfile` this is the CSV file that matches CWID (or SIS ID) with Canvas user ID.
- `cwid_col` the heading of the CWID column in the CSV file.
- `user_id_col` the heading of the Canvas user ID column in the CSV file.
- `course_id` this is the ID number of your course on Canvas.
- `assignment_id` this is the ID number of the exam (which is an assignment) on Canvas.


SCANTOMARK