BDFiddle

Introduction

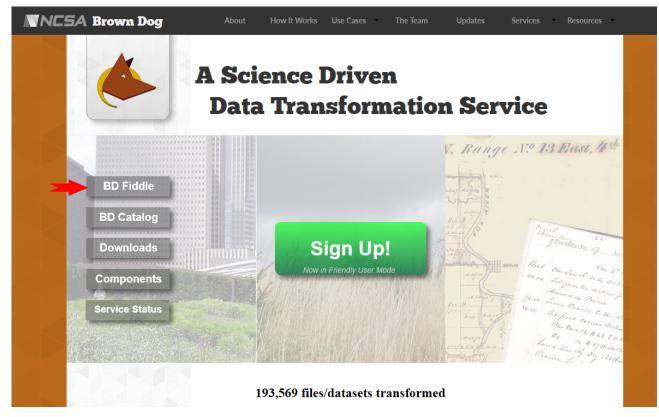
Like other "fiddle" type environments such as JSfiddle, ScalaFiddle and Python Fiddle, BDFiddle provides an easy way for non-programmers to generate and play around with metadata and code "snippets" needed to generate metadata or convert/extract data from files they submit to Brown Dog. At its core, Brown Dog provides two kinds of transformation services, namely extraction and conversion. Extraction refers to the process of automatically generating specific metadata from a file. For example, detecting the number of faces in a photograph. On the other hand, conversion services deal with changing a file from one type to another (e.g. converting an MP4 video file to AVI format). There are multiple modes in which you can use BDFiddle. For example, for a regular user there is the "Automatic" (or basic) mode in which all available transformations (conversions or extractions) are applied to the submitted input file. For a more experienced user, there is the "Manual" (or advanced) mode in which the user can pick and choose specific transformation services and hence currently we do not support uploading files that are larger than 5 MB through BDFiddle. If you need to test with larger files, there is a "Big Data" option in BDFiddle that will generate code snippets in multiple languages (currently shell script, native Python, and Brown Dog Python library are supported), which you can run at your end for local processing of files and uses Docker containers to move compute to data. You can also generate workflows using BDFiddle.

Navigation

Navigate to the Brown Dog website ans sign up:

https://browndog.ncsa.illinois.edu

Click on the "BD Fiddle" button on the left.



or you can navigate directly using the URL:

https://browndog.ncsa.illinois.edu/bdfiddle/

Login with your Brown Dog username and password.

BDF	Fiddle BD Tools Catalog BD AF	PI Examples Report Problem		
	L	Jsername		
		stead??@illindis.edu	-	
	F	Password		

.....

Sign In

••••]

You will be taken to the BDFiddle page itself. This is what you will see. The whole page is setup to assist the user in:

- Selecting a file
 Selecting a transformation
 Receiving the output
 Accessing the code snippets

0	2	3
Select File	Select Transformation	Copy Code Snippet
elect File:		
Select File:	Т	sken (Logout)
Select File: • File: URL:	Browse	xen (Logout) 48afe6d1-5992-4587-adfd-02e48627edec

2) Select Transformation:

Automatic	Convert:	Output:
Manual		
	Extract:	
	Email Output Big Data Submit Generate workflow	

3) Code Snippets:

>_ Terminal				
			Copy To Clipboard	Download Code
Python bd.py				
			Copy To Clipboard	Download Code
Python Native				
	Open Jupyter Notebook	C Download Jupyter Notebook	Copy To Clipboard	Download Code
R bd.r				
			Copy To Clipboard	Download Code
Matlab bd.m				

			C	Copy To Clipboard	Download Code	
JS JavaScript bd.js						
			c	Copy To Clipboard	Download Code	
	UNIVERSITY OF MARYLAND	SMU.				

1. Select a File

You can choose from a file on your computer via the "Browse" button or select URL to choose a remote file. The Examples link at the top provides a few example files to help you get going quickly. Click on the "Get Transformations" button.

0	2	3
Select File	Select Transformation	Copy Code Snippet
) Select File:		
) Select File:		
) Select File: • File: URL:		Token (Logout)
) Select File: • File: OURL: airplane.png	Browse	Token (Logout) 3e81aa03-571b-44e5-a757-a2a65543166a

2. Select Transformation

Based on the type of the input file, a list of available conversions and extractions for that file are displayed. Select the type of transformation needed from either of the scroll lists. If you are using "Manual" mode of operation, you will still need to do the steps mentioned above, except that you can be more specific about the transformation that you want to perform, specifically what individual applications should be used. Press "Submit" button to initiate the transformation.

Associated Options

- · Email Output: To receive the output via email, please select this checkbox and the output will be sent to the email address associated with your Brown Dog account. Currently this feature is available only for conversions.
- Big Data: To obtain code snippets for processing large files locally without sending them to remote services, please select this checkbox. These snippets use Docker and it needs to be installed in your machine.
- Generate Workflow: To generate workflows for your transformations using DataWolf (a web based scientific workflow management system associated), please select this checkbox.

_	1	2	3	
	Select File	Select Transformation	Copy Code Snippet	
Select Transfor	rmation:			
Automatic Co	nvert:	Output:		
Manuai az bi cl cs da di da da	alec			
Ext	ract:			
in	nage/png			
	mail Output 🛛 Big Data	Submit		

3. Receive Output

If a conversion was selected - the output box will provide the URL of the file after converting to the desired format. If an extraction was selected - the output box will display the extracted metadata. The below example displays metadata in the output box generated from a JPG file after extraction.

		1 Select File	2 Select Transformation		Copy Code Snippet
1) Select F					
	O File: O nosa.jpg	URL	Browse	Token (Logout)	0-473-98ed-a75eeeee834a
	Get Transf	formations		61911078-030	Host: https://bd-api-dw.ress.illinois.adu
2) Select T	ransformat	ion:			
Automatic	Convert		Output:		
Manual	asci		File Submitted:		8
	azw3 bil.zip bmp		"id": "690a01304f0c	ca1e46164d01f*	
	clim csv dalec		Extraction Status:		
	dib doc		("ncsa.cv.eyes": "DO!	NE,	
	docx Extract		"ncsa.file.digest": "D "ncsa.cv.cioseups":	DONE", "DONE",	
	image()		"nosa.image.oor": "D "nosa.image.humang	pref": "DONE",	
			"gi_detector": "DON	Message.start: Started p IE*,	rocessing",
			"nosa.versus.image" "nosa.image.preview	w": "DONE",	
	 Email 0 General 	Output Dig Data	Submit "nosa.cv.meangrey": "terra.plantov": "DOI "nosa.cv.caltech101"	NE",	
			"nosa.cv.profiles": "0		-
3) Code Sr	ippets:				
	P.4	id='curl -s -F "File=@ncs fer=Encoding: chunked" -H	a.jpg" https://bd-api-dev.ncsa.i f "Authorization: 6191fb7a-d564-d	illinois.edu/dts/api/ 4ff3-98ed-a75eeee834	/extractions/upload_file -H "Trans Ha" jq -r ".id"
	Terminal	curl https://bd-api-dev.n ff3-98ed-a75eeeee834a*	csa.illinois.edu/dts/api/files/f	\$id/metadata.jsonld -	H "Authorization: 6191fb7a-d560-4
					Copy To Clipboard Download Code
		import bd			
	Python		extract('https://bd-api-dev.ncsi	a.illinois.edu', 'ncr	a.jpg', '6191fb7a-d568-4ff3-98ed-
	bd.py				Copy To Clipboard Download Code
	2	import requests			
	Python	import time import json import os			
	Native	# Add the full path to th	e file on which you want to run	extractions	
		file_path = "ncsa.jpg" #Upload file			
		<pre>id = requests.post('https 'application/json', 'Auth e('ncsa.jpg'), open('file</pre>	orization': '6191fb7a-d568-4ff3-	/dts/api/extractions/ -98ed-a75eeeee834a'},	'upload_file', headers={'Accept': , files={'File' : (os.path.basenam
		#Poll until output is rea while True:			
		<pre>status = requests.get(' {'Accept': 'application/j if status['Status'] == time.sleep(1)</pre>	son', 'Authorization': '6191fb7#	s.edu/dts/api/extract a-d560-4ff3-98ed-a75e	<pre>iions/' + id + '/status', headers= seeee834a'}).json()</pre>
		rs={'Accept': 'applicatio	https://bd-api-dev.ncsa.illinois m/json', 'Authorization': '61919	s.edu/dts/api/files/ fb7a-d560-4ff3-98ed-a	' + id + '/metadata.jsonld', heade x75eeeee834a'}).json()
		json.dumps(metadata)	Open Jupyter Notebook Downlo	and Jupyter Notebook	Copy To Clipboard Download Code
	R	library(BrownDog)			
	bd.r	metadata ← extract_file(834a*)	"https://bd-api-dev.ncsa.illino:	is.edu", "ncsa.jpg",	"6191fb7a-d560-4ff3-98ed-a75eeeee
				1	Copy To Clipboard Download Code
	-	bd_funcs = bd_client;			
	Matlab bd.m	metadata = bd_funcs.extra 'ncsa.jpg', '6191fb7a-d56	ect('https://bd-api-dev.ncsa.ill; 8—4ff3-98ed-a75eeeee834a');	inois.edu/dts/api/ext	ractions/upload_file',
				1	Copy To Clipboard Download Code
	JS JavaScript	The bd.js library require	es URLs.		
	bd.js				
					Copy To Clipboard Download Code

4. Obtain Code Snippets

BDFiddle also generates code snippets, which can be used by the user to recreate the same transformation from within their applications. Currently we provide code snippets in Python, R, MATLAB, bash, and JavaScript using libraries for Brown Dog. For each language there are four possible ways to interact with Brown Dog on a particular file or dataset: convert a file from a URL, convert a local file after uploading, extract from a file from a URL, and extract from a local file after uploading. The appropriate snippet is generated according to the options you select from within BDFiddle. Some example snippets are show below:

Terminal - shell script that can be run from the command line:



curl https://bd-api.ncsa.illinois.edu/dts/api/files/\$id/metadata.jsonld -H "Authorization: dbff8324-d89e-46bf-b2cb-0c c8d7b7f995"

Copy To Clipboard

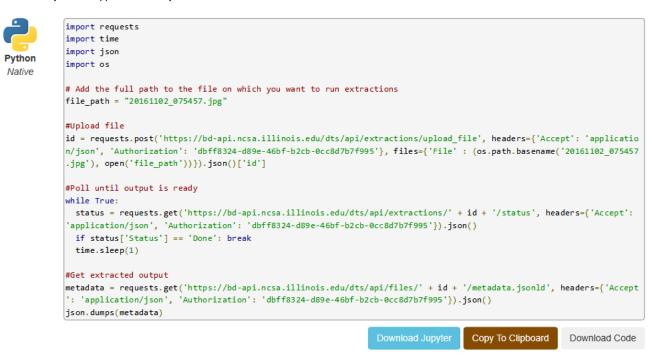
Download Code

Python - snippet that uses bd.py (Brown Dog python library):

Python bd.py	<pre>import bd metadata = json.dumps(bd.extract(https://bd-api.ncsa.illinois.edu, 20161102_075457.jpg, dbff8324-d89e-46bf-b2cb-0cc8d 7b7f995))</pre>

Copy To Clipboard Download Code

Native Python - snippet in native Python:



MATLAB - code to utilize in a MATLAB program to re-run the transformation:



• R - code necessary to re-run the transformation in R language:

R bd.r	<pre>source("bd.r") metadata <- browndog.extract(https://bd-api.ncsa.illinois.edu, 20161102_075457.jpg)</pre>	
	Copy To Clipboard	Download Code

Jupyter

In response to the popularity of Jupyter notebooks and the ease of combining runable code and documentation in one interface that Jupyter provides, Jupyter functionalities have been added to BDFiddle. In addition to copying the code snippet, a user can download a Jupyter notebook that contains the runnable code that is ready to implement locally (jupyter must be installed) or the user can simply open the notebook as an anonymous user on an NCSA multi-user service that utilizes tmpnb. Jupyter templates are JSON files that contain both the raw code (runnable) and markdown (documentation). Where Jupyter is available, two additional buttons appear below the code snippet.

# IMPORTANT: This code snippet needs Virtual Environment (http://docs.pyt	• • • • •			
import requests				
import time				
# Add the full path to the file you	want to convert or run this	code snippet from within the dir	ectory containing the	input file.
file_path = "The Project Manager vs	Business Analyst.pdf"			
# Issue conversion request				
<pre>response = requests.post('https://bd 'text/plain', 'Authorization': '0dd8</pre>			open(file_path, 'rb')}	, headers={'Accept':
text/prainty Autorization, 6000	5655-6625-4065-6704-15761886	2107 })		
# Conversion request succeeded				
if response.status_code == 200:				
print("File submitted for conversi	ion.")			
output = response.text				
output_filename = output.split('/')[-1]			
# Download output				
try:				
while True:				
r = requests.get(output, heade	rs={'Authorization': '0dd69a	95-ea25-4803-a764-f5701aaee167'}	, stream=True)	
if r.status_code == 200:				
output_filename = output.spl	it('/')[-1]			
with open(output_filename, '	wb') as f:			
for chunk in r.iter_conten	t(chunk_size=1024):			
if chunk: # Filter out	keep-alive new chunks			
f.write(chunk)				
f.flush()				
print "Converted file: " + o	output_filename			
break				
print "Waiting for conversion	to complete."			
time.sleep(1)				
except:				
raise				
# Entity too large error				
elif response.status_code == 413:				
print "Input file size is too larg	ge. For the purpose of explor	ing Brown Dog through this code	snippet, please use fi	les that are less th
an 5 MB in size."				
# Other http errors				
else:				
print "HTTP status code: " + respo	onse.status_code			
	Open Jupyter Notebook	Download Jupyter Notebook	Copy To Clipboard	Download Code
	open oppice necessory	e e transade suppres notebook	copy to onpoourd	201111044 0040

The buttons enable a user to:

• Download the Jupyter notebook - to use in a local installation of Jupyter - The downloaded file can be immediately opened in Jupyter by typing "jupyter notebook" in the directory where the file is located.

• Open the Jupyter Notebook - the input file and the Jupyter notebook file are posted to the remote NCSA server where it can be accessed.

When the files are posted to the server, a notebook container is allotted to the new user in which they can edit and run the Jupyter notebook.

<pre>The Edit View Inset Cell Kernel Help (Python 3</pre>		141.142.210.214:8000/user/kpZ9P0mOjD15/notebooks/convert_file.ipynb	€ ☆
<pre>(+) < () () (+) () () () () () () () () () () () () (</pre>			Python 3
<pre>This template allows converting a file in your local file system to a different format. For security reasons, the full path to the file is not included. Add the full path in front of the file name in the variable "file_path' to convert the file you selected in BDFiddle. Remember that security tokens expire, you might need to create a new one and insert it below. What else is editable? Use any file path in your filesystem in "file_path'. Change the output format in the get request (it will only work if it is a valid conversion). In []: import requests import time / Add the full path to the file you want to convert input_file_path = "UK_Creative_462805583.jpg" token = "b7052d5e-f507-4940-be56e-a0d222234346" bd_server = "b1052d5e-f507-4940-be56e-a0d222234346" bd_server = "b1064/best = "b106/best = "b106/best = format = "png" // Fissue conversion request output_filename = output.split('/')[-1] // Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code = 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code = 404): funct.: // Filter out keep-alive new chunks f.vrite(chun</pre>	+ ×	⊘ 10 ↑ ↓ H ■ C Markdown ↓ □ CellToolbar	
<pre>This template allows converting a file in your local file system to a different format. • For security reasons, the full path to the file is not included. • Add the full path in front of the file name in the variable 'file_path' to convert the file you selected in BDFiddle. • Hemember that security tokens expire, you might need to create a new one and insert it below. • What elebe is editable? • Use any file path in your filesystem in 'file_path'. • Change the output format in the get request (it will only work if it is a valid conversion). In []: import requests import time / Add the full path to the file you want to convert input_file_path = 'UK_Creative_462809583.jpg' token = 'bY052d5e-f507-4940-be8e-a0d2822394346' bd.server = 'https://bd-apli.ncsa.illinois.edu' output_format = 'png' /fissue conversion request output_filename = output.split('/')[-1] //Download output try; r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code = 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code = 404): if count; //illename : 'wb') as f: for othuck is r.iter content(chunk, sire=1024); if count; //illename, 'wb') as f: for othuck is r.iter content(chunk, sire=1024); if count; //illename, 'wb') as f: for othuck is r.iter content(chunk, sire=1024); if count; //illename, 'wb') as f: for othuck is r.iter content(chunk, sire=1024); if count; //illename, 'wb') as f: for othuck is r.iter content(chunk, sire=1024); if count; //illename, 'wb') as f: for othuck is r.iter content(chunk, sire=1024); if count; //illename, 'wb') as f: for othuck is r.iter content(chunk, sire=1024); if count; //illename, 'wb') as f: for othuck is r.iter content(chunk) format counts format('Conversion Finished.') </pre>			
<pre> • For security measons, the full path to the file is not included. • Add the full path in front of the file name in the variable 'file_path' to convert the file you selected in BDFiddle. • Remember that security tokens expire, you might need to create a new one and insert it below. • What else is editable? • Use any file path in your filesystem in 'file_path'. • Change the output format in the get request (it will only work if it is a valid conversion). In []: import requests import time # Add the full path to the file you want to convert input_file_path = 'UK_Creative_462809583.jpg' token = 'br052d59e-f507-4940-be6e-a0d282294346' bd_server = 'https://bd-api.nosa.illinois.edu' output_format = 'ngg' #Issue conversion request output_file_name = output.split('/')[-1] #Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code = 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code = 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #filter out keep-alive new chunks f.vrite(chunk) f.flush() except: raise prist('Conversion Finished.') </pre>		Convert Local File to Different Format	
<pre>Add the full path in front of the file name in the variable 'file_path' to convert the file you selected in BDFiddle. FRemember that socurity tokens expire, you might need to create a new one and insert it below. What else is editable? Use any file path in your filesystem in 'file_path'. Change the output format in the get request (it will only work if it is a valid conversion). In []: import requests import time # Add the full path to the file you want to convert input file path = "wx Creative_462809581.jpg" token = "b7052d5e-f507-4940-be8c-a0d282294346" bd_server = 'https://bd-api.ncss.illinois.edu" output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/)[-1] #Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code == 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #filter out keep-alive new chunks f.vrite(chunk) f.fush() except: raise print('Conversion Finished.') </pre>		This template allows converting a file in your local file system to a different format.	
<pre>• Remember that security tokens expire, you might need to create a new one and insert it below. • What else is editable? • Use any file path in your filesystem in 'file_path'. • Change the output format in the get request (it will only work if it is a valid conversion). In []: import requests import time # Add the full path to the file you want to convert input_file_path = "ux Creative 462809583.jpg' token = 'b7052d5e-f507-4940-be8c-a0d282294346' bd_server = 'https://bd-api.ncss.illinois.edu' output_format = 'png' #Issue conversion request output_split('/)[-1] #Download output try: r = requests.get(odtput, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): vith open(output_filename, 'wb') as f: for chunk is r.iter_content(chunk_size=1024): if chunk: #Ither out keep-alive new chunks f.vrite(chunk) f.ush() except: raise print('Conversion Finished.')</pre>		 For security reasons, the full path to the file is not included. 	
<pre>• What else is editable? • Use any file path in your filesystem in "file_path". • Change the output format in the get request (it will only work if it is a valid conversion). In []: import requests import time # Add the full path to the file you want to convert input_file_path = "UK_Creative_462809583.jpg" token = "b7052d5s-f507-4940-be8c-a0d282294346" bd_server = "https://bd-api.ncsa.illinois.edu" output_format = "png" #Issue conversion request output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output = requests.get(output, headers={'Authorization': token}, stream=True) #while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code i= 404): time.sleep(1) r = request.get(output, split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #filter out keep-alive new chunks f.vrite(chunk) f.flush() except: raise print('Conversion Finished.")</pre>			BDFiddle.
<pre>• Use any file path in your filesystem in 'file_path'. • Change the output format in the get request (it will only work if it is a valid conversion). In []: import requests import time # Add the full path to the file you want to convert input_file_path = "UK_Creative_462805983.jpg' token = "b1052365=-f507-4940-b86e-a0d282293436" bd_server = "https://bd-api.ncsa.illinois.edu" output_format = 'png' #Issue conversion request output_filename = output.split('/')[-1] #Download output trg: r = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/')[-1] #Download output trg: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code i= 404): output_filename = output.split('/')[-1] with open(output_filename, 'vb') as f: for chunk is r.iter_content(chunk size=1024): if chunk:</pre>			
<pre>• Change the output format in the get request (it will only work if it is a valid conversion). In []: import requests import time # Add the full path to the file you want to convert input_file_path = "UK_Creative_462809583.jpg" token = "b7052d5e=f507-4940-be8c=a0d282294346" bd_server = "https://bd-api.ncsa.illinois.edu" output_format = "png" #Issue conversion request output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_file_name = output.split('/')[-1] #Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.urite(chunk) f.flush() except: raise print('Conversion Finished.")</pre>			
<pre>In []: import requests import time # Add the full path to the file you want to convert input_file_path = "UK_Creative_46280583.jpg" token = "b705265e-f507-4940-be8c-a0d282294346" bd_server = "https://bd-api.ncsa.illinois.edu" output_format = "png" #Issue conversion request output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/')[-1] #Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.vrite(chunk) f.flush() exceept: raise print("Conversion Finished.")</pre>			
<pre>import time / Add the full path to the file you want to convert input_file_path = "UK_Creative_462809583.jpg" token = "br052d5e-f507-4940-be8e-add2239436" bd_server = "https://bd-api.ncsa.illinois.edu" output_format = "png" //Ssue conversion request output = request.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/')[-1] //Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk is r.iter_content(chunk_size=1024): if chunk:</pre>			
<pre>input_file_path = "UK_Creative_46280588.jpg" token = "b7052d5e-f507-4940-be8c-a0d282294346" bd_server = "https://bd-api.ncsa.illinois.edu" output_format = "png" //Issue conversion request output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/')[-1] //Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code == 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: //filter out keep-alive new chunks f.vrite(chunk) f.flueh() except: raise print{"Conversion Finished.") </pre>	In []:	import time	
<pre>bd_server = "https://bd-api.ncsa.illinois.edu" output_format = "png" #Issue conversion request output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/')[-1] #Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk im r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.write(chunk) f.flush() except: raise print("Conversion Finished.")</pre>			
<pre>output_format = "png" // Issue conversion request output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/')[-1] // Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if (r.status(chunk) f.flush() except: raise print("Conversion Finished.") </pre>			
<pre>#Issue conversion request output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/')[-1] #Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code i= 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk:</pre>			
<pre>output = requests.get(bd_server+'/dap/convert/'+output_format+'/', files={'file': open(input_fil output_filename = output.split('/')[-1] #Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.fush() except: raise print("Conversion Finished.")</pre>			
<pre>output_filename = output.split('/')[-1] #Download output try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code 1= 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk:</pre>			open(input fil
<pre>try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk:</pre>			opon(input_tit
<pre>try: r = requests.get(output, headers={'Authorization': token}, stream=True) while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code != 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk:</pre>		flownload output	
<pre>while(r.status_code == 404): time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code i= 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk:</pre>			
<pre>time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code 1= 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.write(chunk) f.flush() except: raise print("Conversion Finished.")</pre>		<pre>r = requests.get(output, headers={'Authorization': token}, stream=True)</pre>	
<pre>time.sleep(1) r = requests.get(output, headers={'Authorization': token}, stream=True) if(r.status_code 1= 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.write(chunk) f.flush() except: raise print("Conversion Finished.")</pre>		while(r.status code == 404):	
<pre>if(r.status_code i= 404): output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.write(chunk) f.flush() except: raise print("Conversion Finished.")</pre>		time.sleep(1)	
<pre>output_filename = output.split('/')[-1] with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk:</pre>		<pre>r = requests.get(output, headers={ Authorization : token}, stream=True)</pre>	
<pre>with open(output_filename, 'wb') as f: for chunk in r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.write(chunk) f.flush() except: raise print("Conversion Finished.")</pre>			
<pre>for chunk in r.iter_content(chunk_size=1024): if chunk: #Filter out keep-alive new chunks f.write(chunk) f.flush() except: raise print("Conversion Finished.")</pre>		<pre>output_filename = output.split('/')[-1]</pre>	
<pre>if chunk: #Filter out keep-alive new chunks f.write(chunk) f.flush() except: raise print("Conversion Finished.")</pre>			
<pre>f.write(chunk) f.flush() except: raise print("Conversion Finished.")</pre>			
except: raise print("Conversion Finished.")			
<pre>raise print("Conversion Finished.")</pre>			
		and to the second s	
		<pre>print("Conversion Finished.") print("File saved as: " + str(output_filename))</pre>	

See Jupyter documentation for specific usage of Jupyter notebooks:

https://jupyter.readthedocs.io/en/latest/

Report a Problem

Log in and click "Report Problem" in the Navigation bar then you will see the form below:

Report an issue Your Name (required)	
four Email (required)	
yanzhao3@illinois.edu	
Subject (required)	
Comment	
Attachment Choose Files No file chosen	
Submit	

Fill in the form with your name and details of the problem. Screenshots or any file can be uploaded in the attachment. If the report is sent successfully, you will see the message as follows:

Report submitted! Thank you for your feedback yanzhao3.