

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 transformations that they want to apply to the input file. BDFiddle is mainly meant to be a tool for exploring Brown Dog by playing around with its 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 and sign up:

<https://browndog.ncsa.illinois.edu>

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

NCSA Brown Dog About How It Works Use Cases The Team Updates Services Resources

A Science Driven Data Transformation Service

BD Fiddle
BD Catalog
Downloads
Components
Service Status

Sign Up!
Now in Friendly User Mode

193,569 files/datasets transformed

or you can navigate directly using the URL:

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

Login with your Brown Dog username and password.

Username


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:

1. Selecting a file
2. Selecting a transformation
3. Receiving the output
4. Accessing the code snippets

1) Select File:

☒ File: ☐ URL:

Browse...

Get Transformations

Token (Logout)

48afe6d1-5992-4587-adfd-02e48627edec

Host: <https://bd-api-dev.ncsa.illinois.edu>

2) Select Transformation:

Automatic

Manual

Convert:

Extract:

☐ Email Output

☐ Big Data

☐ Generate workflow

Submit

Output:

3) Code Snippets:



Terminal

Copy To Clipboard

Download Code



Python
bd.py

Copy To Clipboard

Download Code



Python
Native

Open Jupyter Notebook

Download Jupyter Notebook

Copy To Clipboard

Download Code



bd.r

Copy To Clipboard

Download Code



Matlab
bd.m

Copy To Clipboard

Download Code

[Copy To Clipboard](#)[Download Code](#)[Copy To Clipboard](#)[Download Code](#)

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.

[BD Fiddle](#)[BD Tools Catalog](#)[BD API](#)[Examples](#)[Report Problem](#)

1

Select File

2

Select Transformation

3

Copy Code Snippet

1) Select File:

☒ File: ☐ URL:

airplane.png

[Browse...](#)[Get Transformations](#)

Token (Logout)

3e81aa03-571b-44e5-a757-a2a65543166a

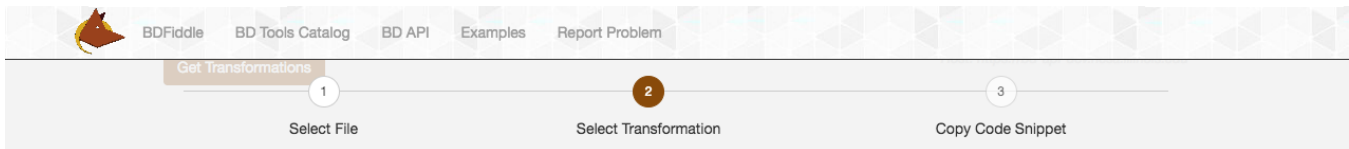
Host: <https://bd-api-dev.ncsa.illinois.edu>

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.



2) Select Transformation:

Automatic

Manual

Convert:
ascii
azw3
bil.zip
bmp
clim
csv
dalec
dib
doc
docx
etc.

Extract:
image/png

☐ Email Output
☐ Generate workflow

☐ Big Data

Submit

Output:

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:

☒ File: ☐ URL:

Token (Logout)

6191fb7a-d560-4ff3-98ed-a75eeeee834a

Host: https://bd-api-dev.ncsa.illinois.edu

2) Select Transformation:

Automatic
Manual

Convert:

ascii
azw3
bzip
bmp
cgm
csv
dazle
dib
doc
docx
...

Extract:

image/jpeg

☐ Email Output
☐ Big Data

Output:

File Submitted:

{"id": "590a01304f0c4e46164d01f"}

Extraction Status:

{"ncaa.cveyes": "DONE",
"ncaa.file.digest": "DONE",
"ncaa.cv.closeups": "DONE",
"ncaa.image.occr": "DONE",
"ncaa.image.humanpref": "DONE",
"siegfried": "StatusMessage.start: Started processing",
"gl_detector": "DONE",
"ncaa.versus.image": "DONE",
"ncaa.image.preview": "DONE",
"ncaa.cv.meangrey": "DONE",
"terra.planetv": "DONE",
"ncaa.cv.caboch101": "DONE",
"ncaa.cv.profiles": "DONE",
...

3) Code Snippets:

Terminal

```
id="curl -s -F "File=@ncaa.jpg" https://bd-api-dev.ncsa.illinois.edu/dts/api/extractions/upload_file -H "Transfer-Encoding: chunked" -H "Authorization: 6191fb7a-d560-4ff3-98ed-a75eeeee834a" | jq -r ".id"
curl https://bd-api-dev.ncsa.illinois.edu/dts/api/files/$id/metadata.json?id="Authorization: 6191fb7a-d560-4ff3-98ed-a75eeeee834a"
```

Python
bd.py

```
import bd
metadata = json.dumps(bd.extract('https://bd-api-dev.ncsa.illinois.edu', 'ncaa.jpg', '6191fb7a-d560-4ff3-98ed-a75eeeee834a'))
```

Python
Native

```
import requests
import time
import json
import os

# Add the full path to the file on which you want to run extractions
file_path = "ncaa.jpg"

#Upload file
id = requests.post('https://bd-api-dev.ncsa.illinois.edu/dts/api/extractions/upload_file', headers={'Accept': 'application/json', 'Authorization': '6191fb7a-d560-4ff3-98ed-a75eeeee834a'}, files={'File': (os.path.basename(file_path), open(file_path))}).json()['id']

#Poll until output is ready
while True:
    status = requests.get('https://bd-api-dev.ncsa.illinois.edu/dts/api/extractions/' + id + '/status', headers={'Accept': 'application/json', 'Authorization': '6191fb7a-d560-4ff3-98ed-a75eeeee834a'}).json()
    if status['Status'] == 'Done': break
    time.sleep(1)

#Get extracted output
metadata = requests.get('https://bd-api-dev.ncsa.illinois.edu/dts/api/files/' + id + '/metadata.json?id=' + id, headers={'Accept': 'application/json', 'Authorization': '6191fb7a-d560-4ff3-98ed-a75eeeee834a'}).json()
json.dumps(metadata)
```

R
bd.r

```
library(BrownDog)
metadata <- extract_file("https://bd-api-dev.ncsa.illinois.edu", "ncaa.jpg", "6191fb7a-d560-4ff3-98ed-a75eeeee834a")
```

Matlab
bd.m

```
bd_funcs = bd_client;
metadata = bd_funcs.extract('https://bd-api-dev.ncsa.illinois.edu/dts/api/extractions/upload_file', 'ncaa.jpg', '6191fb7a-d560-4ff3-98ed-a75eeeee834a');
```

JavaScript
bd.js

```
The bd.js library requires URLs.
```

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:



```
id=`curl -s -F "File=@20161102_075457.jpg" https://bd-api.ncsa.illinois.edu/dts/api/extractions/upload_file -H "Transfer-Encoding: chunked" -H "Authorization: dbff8324-d89e-46bf-b2cb-0cc8d7b7f995" | jq -r ".id"`

curl https://bd-api.ncsa.illinois.edu/dts/api/files/$id/metadata.jsonld -H "Authorization: dbff8324-d89e-46bf-b2cb-0cc8d7b7f995"
```

[Copy To Clipboard](#)
[Download Code](#)

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



```
import bd

metadata = json.dumps(bd.extract(https://bd-api.ncsa.illinois.edu, 20161102_075457.jpg, dbff8324-d89e-46bf-b2cb-0cc8d7b7f995))
```

[Copy To Clipboard](#)
[Download Code](#)

- Native Python - snippet in native Python:



```
import requests
import time
import json
import os

# Add the full path to the file on which you want to run extractions
file_path = "20161102_075457.jpg"

#Upload file
id = requests.post('https://bd-api.ncsa.illinois.edu/dts/api/extractions/upload_file', headers={'Accept': 'application/json', 'Authorization': 'dbff8324-d89e-46bf-b2cb-0cc8d7b7f995'}, files={'File' : (os.path.basename('20161102_075457.jpg'), open('file_path'))}).json()['id']

#Poll until output is ready
while True:
    status = requests.get('https://bd-api.ncsa.illinois.edu/dts/api/extractions/' + id + '/status', headers={'Accept': 'application/json', 'Authorization': 'dbff8324-d89e-46bf-b2cb-0cc8d7b7f995'}).json()
    if status['Status'] == 'Done': break
    time.sleep(1)

#Get extracted output
metadata = requests.get('https://bd-api.ncsa.illinois.edu/dts/api/files/' + id + '/metadata.jsonld', headers={'Accept': 'application/json', 'Authorization': 'dbff8324-d89e-46bf-b2cb-0cc8d7b7f995'}).json()
json.dumps(metadata)
```

[Download Jupyter](#)
[Copy To Clipboard](#)
[Download Code](#)

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



```
bd_funcs = bd_client;

metadata = bd_funcs.extract('https://bd-api.ncsa.illinois.edu/dts/api/extractions/upload_file', '20161102_075457.jpg', 'dbff8324-d89e-46bf-b2cb-0cc8d7b7f995');
```

[Copy To Clipboard](#)
[Download Code](#)

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



```
source("bd.r")

metadata <- browndog.extract(https://bd-api.ncsa.illinois.edu, 20161102_075457.jpg)
```

[Copy To Clipboard](#)[Download Code](#)

Jupyter

In response to the popularity of Jupyter notebooks and the ease of combining runnable 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 [tmprnb](#). 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 Python Requests (http://docs.python-requests.org/) to be installed. You can install it in your Python
Virtual Environment (http://docs.python-guide.org/en/latest/dev/virtualenvs/) using the command "pip install requests."
import requests
import time

# Add the full path to the file you want to convert or run this code snippet from within the directory containing the input file.
file_path = "The Project Manager vs Business Analyst.pdf"

# Issue conversion request
response = requests.post('https://bd-api.ncsa.illinois.edu/v1/conversions/txt/', files={'file': open(file_path, 'rb')}, headers={'Accept':
'text/plain', 'Authorization': '0dd69a95-ea25-4803-a764-f5701aeee167'})

# Conversion request succeeded
if response.status_code == 200:
    print("File submitted for conversion.")
    output = response.text
    output_filename = output.split('/')[-1]

# Download output
try:
    while True:
        r = requests.get(output, headers={'Authorization': '0dd69a95-ea25-4803-a764-f5701aeee167'}, stream=True)

        if r.status_code == 200:
            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()
            print "Converted file: " + 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 large. For the purpose of exploring Brown Dog through this code snippet, please use files that are less th
an 5 MB in size."

# Other http errors
else:
    print "HTTP status code: " + response.status_code
```

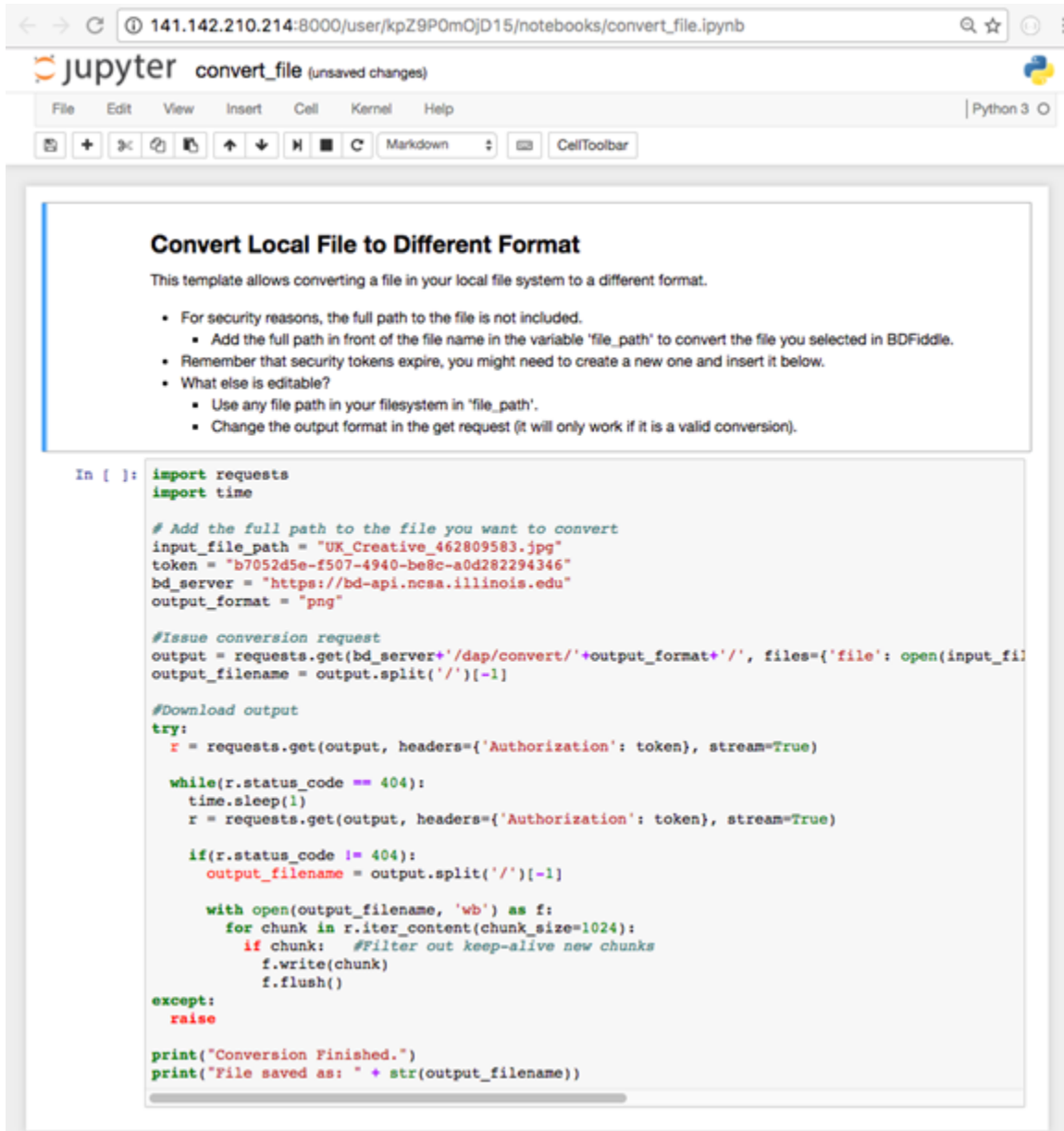
[Open Jupyter Notebook](#)[Download Jupyter Notebook](#)[Copy To Clipboard](#)[Download Code](#)

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.



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)

Your Email (required)

Subject (required)

Comment

Attachment

No file chosen

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.