Batch Media Grabber

This document describes the Media Grabber available as a command line program on the Pirate Stick & Pirate Box. It has evolved over the course of several years to a state of complexity that warrants an explanation of how it works and how to use it. The code is written in the python3 programming language, is open source and available for free on my github repository at .https://github.com/ThomasFreedman/Pirate-Stick The Media Grabber is a very close derivative of the *ipfs-VideoGrabber* production code I use on my IPFS servers. The differences are mainly cosmetic.

Why I Created the Media Grabber

The original *ipfs-VideoGrabber* program was created as a way for me to save videos from youtube for later viewing. As my video collection grew it wasn't long before I started to run out of local storage space to save such large media, so I added IPFS storage to it. Trying to locate that documentary or topic "*I just know I saved*" out of the 100s (and now 1000s) of video and audio files in my collection required some means of keeping track of what I downloaded and added to IPFS. That's when I discovered the lightweight SQLite database, a non-server, file-based SQL format similar to Micr$oft Access of which I was highly familiar.

youtube-dl is the Core Component of the Media Grabber

youtube-dl is capable of downloading both video and audio media from many other sources besides youtube. However, it's important to note that youtube-dl cannot scrape content from all URLs that contain media. It is difficult to describe why some succeed and others fail. This is due to the limitations of the open source youtube-dl software the Media Grabber relies on to grab video and or audio files. More precisely, it is due to how the youtube-dl *extractor* modules are written.

The Media Grabber is built with the modular youtube-dl python library that utilizes small modules named *extractors* that are written for specific media sources such as bitchute or vimeo. There are hundreds of these *extractors*, some of which handle multiple media sources which give youtube-dl the capability to obtain media from 1000s of different sources such as vimeo, bitchute, lrby and odysee, just to name a few. The variety of these extractors covers a diverse range of sources.

Please be aware youtube-dl is updated frequently by its' developers. The Pirate Stick and Pirate Box use a scheduled cron task that runs periodically to keep youtube-dl up-to-date. The Media Grabber may mysteriously stop working one day for reasons totally outside of your control. It could be due to changes by the media source, such as changes to their "Terms Of Service", or changes to the media formats offered or for many other reasons. The one thing you do have control over however is how up to date you keep youtube-dl. Refer to *Appendix A* later in this document for how to do that. The developers of youtube-dl are very pro-active. If a change to terms of service or policies cause youtube-dl to fail, it doesn't take long for a new release to be produced that addresses the problem. Users such as yourself report these issues and youtube-dl responds quickly.

Basic Operation

The Media Grabber is designed to run periodically by some type of scheduler such as Linux's cron. I use cron to run video grabbers on my IPFS servers once a day. However, the Media Grabber can also be run on demand to obtain content from a single URL. A single playlist URL could produce 100s or 1000s of downloads. The Media Grabber tool in the main Pirate menu will download the media for one or more URLs and add them to IPFS, and update the "catalog.sqlite" database with the metadata for the media files downloaded. The URLs can be individual videos or audios, or even playlists. This is a simple front end for the **ytdlMediaGrabber.py** python3 program. It doesn't provide the more advanced filtering possible with batch operation available with the **ytdlMediaGrabber** tool in the *~/bin/mediaGrabber* folder.

Batch Operation

A more powerful mode of operation is using a configuration file to specify a list of URLs, in multiple groups and with various filter options such as publisher (synonymous with groups which are designated as “grupes” in the source code), length of play (duration), upload date and others. Publisher usually corresponds with something like a youtube channel or perhaps a blogger’s media webpage. It could also be topical. It is simply a way to organize the media downloaded that is meaningful to the user.

For example, the “singles” Publisher / grupe is a catch-all group where “one-shot” videos are grouped together even though different URLs are provided from a variety of publishers and topics.

Batch mode is very useful for downloading content for a Publisher / grupe whenever new items become available. youtube-dl will not download content it has already downloaded.

The Configuration File

The configuration file is a JSON formatted text file that specifies the parameters that control downloading in batch mode. You can edit this file with any text editor however the JSON format can be cumbersome to deal with for those unfamiliar with it. It is highly recommended to make use of the many online tools available for editing and syntax checking JSON files. Doing so may save you from going bald prematurely! Refer to the list of such tools at: https://geekflare.com/json-online-tools for other choices you may find useful.

The JSON configuration file contains three major sections, all of which are python dictionaries. A python3 dictionary (known as associative arrays in other languages) is a name-value pair structure. For our purposes here the name of each section is the key for a dictionary and are always strings and must be unique within the JSON file. The value side of the pair can be any valid python type, including lists, numbers, strings and even another dictionary. Unlike python, the JSON format doesn’t care about indentation at all. Refer t *Appendix B* for an example JSON configuration file. The names of the three sections of the configuration file are:

1. **Comments**
   The comments section is optional but highly recommended. The value is a list of strings, all of which are double quoted, comma separated. The last item in any python list does not have a comma. The list must be enclosed in [square brackets].

2. **Global Settings**
   This section contains values that affect all downloads. It contains a list of dictionaries, the last one being DLOpts whose value is also a list of dictionaries:

   DLbase: the value of this dictionary key is a string containing the full path to the folder where all downloaded files are saved and where all of the following files reside.
   DLeLog: name of the error log file.
   DLarch: name of the file where youtube-dl keeps track of downloads
   DLmeta: the name of the SQLite database schema file that describes the metadata obtained from youtube-dl and most of the fields in the database. The IPFS Search Tool relies on this database and these metadata fields to function.
   DLOpts: a dictionary of option settings for youtube-dl. **Do not change!**

3. **Grupes**
   This is a list of one or more dictionaries that define a grupe of associated URLs to download and any filter parameters they share. Each grupe name is totally arbitrary, however keep in mind this string will be used to create a folder in the file system so choose the names that

don’t violate file system restrictions. In particular I recommend no spaces in these names, and instead use “CamelCaseFormat”. The value of each grupe is a dictionary of named values. The value of the final “urls” dictionary is a list of one or more comma separated URL strings for the grupe. Here is a brief explanation for each of the dictionaries that represent filters for what is downloaded:

Quota: Limits the number of files or the amount of disk space allowed for this grupe.
Active: Either true or false. If false no downloads will be done for this group.
Duration: Minimum duration of media play in seconds required to download.
Start: Earliest upload date (“YYYYMMDD”) required or *null* for no lower limit.
End: Latest upload date string limit (“YYYYMMDD”) or *null* for no upper limit.
Stop: The maximum downloads for a single run of the youtube-dl program for this grupe.
Urls: A list of one or more media source URLs (where to download media from)

Appendix A: Keeping youtube-dl up to date

As mentioned earlier, the youtube-dl program and its’ python library are updated automatically by the Pirate Box & Pirate Stick if it is connected to the Internet. If you wish to use the Media Grabber it is highly recommended to periodically run the ytdl-update.bash script manually from the ~/bin/mediaGrabber folder if your system is not online frequently or for only short periods of time. How often you do this is up to you, but if you encounter errors while running the Media Grabber that might be a good time!

Appendix B: Example JSON Configuration File for PBVG

```
{
  "Comments": [ "This is an example configuration file that defines 2 grupes,",
                "The first is fictitious and disabled and the 2nd an actual",
                "working example.",

                "RECOMMENDED CONVENTIONS: CamelCaseNames and no spaces"
  ],
  "DLbase": "/home/ipfs/bin/mediaGrabber/ytDL/",
  "DLeLog": "error.log",
  "DLarch": "youtube-dl.history",
  "DLmeta": "metadataFields.json",
  "DLOpts": {
    "retries": 5,
    "format": "best",
    "continuedl": true,
    "ignoreerrors": true,
    "merge-output-format": "mp4"
  },

  “Grupes": {
    "Example1": {
      "Quota": 0,
      "Active": false,
      "Duration": 600,
      "Start": 20200301,
      "End": null,
      "Stop": 30,
      "urls": [
        "https://www.youtube.com/bigPlaylist...",
        "https://www.youtube.com/whatever more...",
        "https://www.youtube.com/whatever even more..."
      ]
    },

    "Crrow777": {
      "Quota": 0,
      "Active": true,
      "Duration": 0,
      "Start": null,
      "End": null,
      "Stop": 6,
      "urls": [
        "https://www.crrow777radio.com/wp-content/uploads/2021/03/Episode5YbWdPLkkN-300.2-Hour-1.mp3",
```

```
        "https://www.crrow777radio.com/wp-content/uploads/2021/03/Episode4dOgNsCxZZ-300-Hour-1.mp3",
        "https://www.crrow777radio.com/wp-content/uploads/2021/03/Episode-299-5-Free.wav",
        "https://www.crrow777radio.com/wp-content/uploads/2021/02/Episode67GwEqAsFxX-299-Hour-1.mp3",
        "https://www.crrow777radio.com/wp-content/uploads/2021/02/Episode11HlAvBgCCc-298-Hour-1.mp3",
        "https://www.crrow777radio.com/wp-content/uploads/2021/02/Episode67hVdCeXXZ-297-Hour-1.mp3"
      ]
    }
  }
}
```

Appendix C: Quickstart for Batch Operation

These are the essential steps for using the Media Grabber in batch mode. The files can be found under the */home/ipfs/bin/mediaGrabber* folder.

1. Copy the **ytdl-exampleConfig.json** file to a name of your choosing in the *config* subfolder.
2. Edit your copy to replace the example grupes with your own.
3. Run the program to test using this command line:

/home/ipfs/bin/mediaGrabber/ytdlMediaGrabber.py -c /home/ipfs/bin/mediaGrabber/config/<your config file> -d <anyFullPathNameYouWant.sqlite>

Run the **mediaGrabber.py** program without any arguments to show command line syntax.

The videos / audios at the URLs in the config file will be downloaded to subfolders with your grupe names under the DLbase folder you specified in your configuration file. They will also be added to your IPFS node. The command line above can be run from a program scheduler such as cron to capture new content regularly.

A SQLite database <anyFullPathNameYouWant.sqlite> will also be created with the metadata information about the downloads. This database can be inspected with the IPFS Search tool by selecting the Media Grabber Database from the *File→ Open Search Source* menu of the IPFS Search tool.

To publish your newest SQLite database on IPFS you will need to create a static IPnS name using the IPFS Keys tool and add the hash for that IPnS key on line 10 in the **ytdlServerDefinitions.py** file.

PLEASE TAKE NOTE: this tool is not intended for "grandma" or casual non-technical users to use in its' current form, but rather is a prototype to demonstrate one method of providing an IPFS search capability and how to publish the metadata on IPNS. Optimistically this release will act as a seed for others who can read python3 code or the comments in **ytdlMediaGrabber.py** and take this capability to the next level by creating a GUI to capture input to create the JSON configuration file.

The important part is the metadata capture, useful for global search providers such as Presearch.org. How the catalog.sqlite metadata databases get aggregated by Presearch or others has yet to be defined. If an IPNS key is shared with others, they could search the content they added to their IPFS node. The **Dblist** python3 dictionary on line 26 of /*home*/bin/ipfsSearch/ipfs.py would need to be changed to include the IPNS hash for any other remote servers so it appears in the menus.

Another fairly simple tool on my todo list to create is a metadata & media capture tool. It will be a GUI form that allows a content creator to enter metadata such as publish date, media type, publisher name, title, description etc along with the actual media itself (mp4 video, mp3 audio, PDF document etc) and add it to IPFS directly, updating the catalog.sqlite database for easy tracking & searching later.