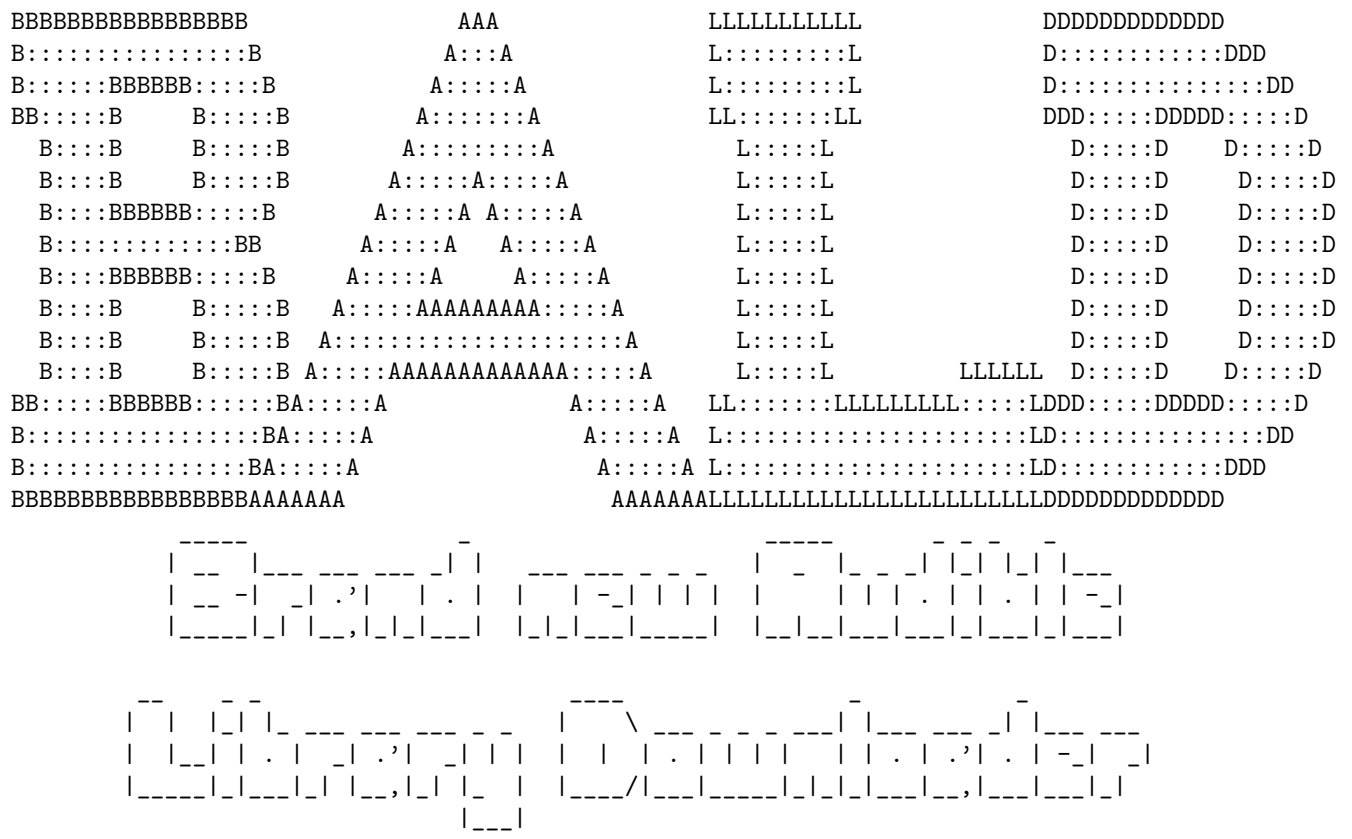

Brand new Audible Library Downloader

Author: damajor

Source code: BALD @ GitHub

Topics: Automatic download / art cover embedding / metadata tagging / conversion / renaming of Audible books

Table of content

Description

BALD is a highly configurable tool to download your Audible library and build your local library.
BALD uses various tools to download, enrich metadata, convert audiobooks, rename files and move them to target location. This script is intended to be run on a Linux system (WSL may work) with bash and some other tools installed

Last run time is stored in a status file for next runs, so only deltas are downloaded.

Some logs will be created & deleted inside `script_dir/tmp` directory. All working files during processing are stored in the same directory as the downloaded books.

A Dockerfile is also provided for those who want to use it in rootless containers.

In case of a problem please open an issue on GitHub https://github.com/damajor/BALD/issues

> **NB:** Some metadata may be wrong or malformed, don't blame the script but complains to Audible If you find a reliable logic or source of metadata to fix Audible quirks, open an issue & let me know.

Feature details

- automatic Audible audiobooks downloads
- crontab friendly (delta runs)
- library history downloads (full & deltas)
- wishlist download
- metadata enrichment (from normal & hidden AAX/AAXC tags, or from history TSV file)
- cover art embedding
- audiobooks conversion to **oga** (Ogg container with Opus codec)
- custom bit rate for size control (96k ~2/3 of original size, 64k ~1/2 of original size, etc.)
- parallelization of heavy work (conversion & metadata extraction)
- "perfect" embedded metadata for AudioBookShelf (AudioBookShelf)
- dynamic naming scheme for destination directories & file names
- helper script for downloading external scripts
- container Dockerfile provided with all dependencies & external scripts (tested on Podman rootless)

Pre-requisites

- install `audible-cli` and create a profile (check instructions here https://github.com/mkb79/audible-cli)
- modify script user config variables to match your needs
- run `parallel --citation` and follow instructions to avoid output warning garbage

Required tools

- bash (v4.3+)
- jq
- xxd (for ogg image blobber script)
- audible-cli (at least version 0.3.2b3)
- ffmpeg / ffprobe (tested 6.1.2 & 7.1)
- mediainfo
- parallel

Also download these scripts and put them in the same directory as this script

- `ogg-image-blobber.sh` (from https://github.com/twopoint71/ogg-image-blobber)

- `update_chapter_titles.py` (from https://github.com/mkb79/audible-cli/blob/master/utils/update_chapter_titles.py)

Optional tools

- Java + Tika (download jar from Apache Tika)
  It slows down metadata extraction but gives very good language accuracy detection.

Or

- Your own Tika server (**processing is faster than using the jar**)

Repository files description

- `BALD.sh` it is the main script with all the processing logic.
- `grab_additional_scripts.sh` a quick and dirty script for lazy boys. It will download external scripts and optional tool for you, as well creating a user config file with default values (just remove `myconfig` if you want to set parameters in the script itself).
- `docker_mod.sh` not to be used directly by the user, it is sourced by the main script to make it compatible with containers.
- `Dockerfile` used to build your own container image.
- `compose.yaml` example of for Podman/Docker compose.
- `README.md` the file you are currently reading.
- `README.pdf` for convenience purposes the README is also exported as PDF file.
- `LICENSE` is the license file of this project.

Installation

It is as simple as:

1. Clone GitHub repository or download archive of it
   `git clone https://github.com/damajor/BALD`

2. Manually grab external tools and put them in the same directory of the script **OR** execute the download script (this will also create a default `myconfig` file)
   `./grab_additional_scripts.sh`

3. Carefully configure all settings in `myconfig` or directly inside the script

Usage examples

Double check prerequisites and settings :)

Manual run

Once configured the only thing is to run the script.

On the first run the script will download ALL your library to the desired location and will do its jobs for metadata, conversion and moving converted files to target location.

Upcoming runs will only download new books from Audible (delta run) and process them as usual.

> **Example:**
> either
> `/full/path/of/script/BALD.sh`
> or if the script is in your path, just use:
> `BALD.sh`

or if you start it manually from the script location:
`./BALD.sh`

Automatic run (crontab)

It is recommended to NOT run it every hour, it works better with a daily scheduling or more like every week or month.

Example:
Run every day at 4PM
```
0 4 * * * /full/path/of/script/BALD.sh >dev/null 2>/dev/null
```
Run every first day of each month at 4PM
```
0 4 1 * * /full/path/of/script/BALD.sh >dev/null 2>/dev/null
```

Notes:
- Please note that automated runs does NOT prevent you to manually run the script at any time you want.
Example, if you set up a weekly update, but you want to quickly run it manually once because you purchased a new audiobook, then just do it! (the script will store the last execution to the time of the manual run)
- Before setting up your automation, try to run manually the script to see if it behaves the way you have configured it.
- **Also, it is HIGHLY recommended doing the first run manually. Just because it will sync entire library and can take hours of processing for big libraries.**

Container usage

Notes:
The script has been developed using Podman rootless.
It should work on Docker (rootless or not).
Everything inside the container runs as 'root' user, **use rootless containers without CAP_ADMIN for security**.

IMPORTANT:
Initialize audible-cli

When used in containers the script makes some settings 'readonly'. The user MUST map specific volumes (the script will check all volumes are mapped, or it will exit).

The following settings are hard coded using the script inside container:

```
HIST_LIB_DIR="/audible_history"
STATUS_FILE="/status_file"
DOWNLOAD_DIR="/audible_dl"
DEST_BASE_DIR="/audiobooks_dest"
DEBUG_USEAAXSAMPLE=/sample.aax
DEBUG_USEAAXCSAMPLE=/sample.aaxc
```

The user is responsible to map volumes accordingly.

Container volume maps list

When mapping local directories to your container, make sure that they all exist.

Required The following volumes MUST be mapped to the container, if one map is missing the script will exit with an error message with the missing volume:

Local	Container	Type
HIST_LIB_DIR	/audible_history	dir
STATUS_FILE	/status_file	file

Local	Container	Type
DOWNLOAD_DIR	/audible_dl	dir
DEST_BASE_DIR	/audiobooks_dest	dir
myconfig	/BALD/myconfig	file
tmp logs directory	/BALD/tmp	dir
audible-cli config dir	/root/.audible	dir

Optional Those volumes are optional, just if you want to debug the script:

Local	Container	Type
DEBUG_USEAAXSAMPLE	/BALD/sample.aax	file
DEBUG_USEAAXCSAMPLE	/BALD/sample.aaxc	file
aaxc voucher file	/BALD/sample.voucher	file

Audible CLI initialization

If you have `audible-cli` installed and initialized then you can directly map your `audible-cli` config directory and skip this part.

If you had never used `audible-cli` then follow what comes next.

Make local `audible-cli` configuration directory:
`mkdir -p /home/myuser/.audible`

Run `audible-cli` quick-start enrollment:
`podman run -it --rm -v /home/myuser/.audible:/root/.audible quay.io/damajor/bald:latest audible quickstart`

For more information refers to `audible-cli` documentation here https://github.com/mkb79/audible-cli/blob/master/README.md#getting-started.

Podman command line

If you want to use the following settings with Podman:

```
HIST_LIB_DIR="/home/myuser/Audible/lib_history"
STATUS_FILE="/home/myuser/Audible/audible_last_sync"
DOWNLOAD_DIR="/home/myuser/Audible/Downloads"
DEST_BASE_DIR="/home/myuser/AudioBookShelf/audiobooks"
DEBUG_USEAAXSAMPLE=sample.aax
DEBUG_USEAAXCSAMPLE=sample.aaxc
```

Use the following command (*change local paths to match your needs*):

```
podman run -it --rm \
    -v /home/myuser/.audible:/root/.audible \
    -v /home/myuser/Audible/lib_history:/audible_history \
    -v /home/myuser/Audible/audible_last_sync:/status_file \
    -v /home/myuser/Audible/Downloads:/audible_dl \
    -v /home/myuser/AudioBookShelf/audiobooks:/audiobooks_dest \
    -v /home/myuser/BALD/myconfig:/BALD/myconfig \
    -v /home/myuser/BALD/tmp:/BALD/tmp \
    quay.io/damajor/bald:latest
```

If you want to debug using sample files, just add:

```
    -v /home/myuser/BALD/sample.aax:/BALD/sample.aax \
    -v /home/myuser/BALD/sample.aaxc:/BALD/sample.aaxc \
    -v /home/myuser/BALD/sample.voucher:/BALD/sample.voucher \
```

Podman compose

Use compose file only after you have done `audible-cli` initialization (Help here). Compose file example:

```
services:
  BALD:
    image: quay.io/damajor/bald:latest
    environment:
      - TZ=Europe/Rome
    volumes:
      - /home/mysuser/.audible:/root/.audible:z
      - /home/mysuser/Audible/lib_history:/audible_history:z
      - /home/mysuser/Audible/audible_last_sync:/status_file:z
      - /home/mysuser/Audible/downloads:/audible_dl:z
      - /home/mysuser/AudioBookShelf/audiobooks:/audiobooks_dest:z
      - /home/mysuser/BALD/myconfig:/BALD/myconfig:z
      - /home/mysuser/BALD/tmp:/BALD/tmp:z
```

Run with:
`podman compose -f compose.yaml up --force-recreate`

Configuration

Make sure you read all setting descriptions and usage.

Change the variables in the script section **User config** according to your needs.

All the parameters MUST have a value set (there is only basic validation).

You can avoid to change settings in the script itself by creating a file named `myconfig` and put all the settings in it. The script will load this file if it exists, otherwise it will use default values from script itself. `myconfig` file takes precedence over internal settings.

Script main settings

AUDIBLECLI_PROFILE

This variable holds the profile name created with `audible-cli`. It should match one of the existing JSON file in `~/audible/`

> **Example config (`~/audible/myprofile.json`):**
> `AUDIBLECLI_PROFILE=myprofile`

HIST_LIB_DIR

Location of the downloaded library history files. Those files are TSV files containing your entire list of Audiobooks (and Podcasts) and delta TSV files.

> **Snapshot of what it looks:**
>
> ```
> 2024-02-01T00:00:00_library_full.tsv
> 2024-02-01T00:00:00_library_new.tsv
> 2024-03-01T00:00:00_library_full.tsv
> 2024-03-01T00:00:00_library_new.tsv
> 2024-04-01T00:00:00_library_full.tsv
> ```
>
> **Example config:**
> `HIST_LIB_DIR=$HOME/Audible/lib_history`

HIST_FULL_LIB

This flag allows to download the full history TSV file of your Audible library each time the script is run. If you run the script every day I recommend disabling it. If you run every week or month then you can keep it to 'true'.

Allowed values:
- 'true'
- 'false' or anything else

> **Example config:**
> `HIST_FULL_LIB=true`
>
> **Note:** This setting is ignored when using the container. You MUST use volume mapping instead.

STATUS_FILE

This is the full path and filename where the script stores its last execution time. The next run will use the last execution time to download only new audiobooks. **If you want to force a full sync, delete the file and run again.**

The status file stores the complete date of the script's last execution (the date is formulated as `+%Y-%m-%dT%H:%M:%S`).

> **Example config:**
> `STATUS_FILE=$HOME/Audible/audible_last_sync`
>
> **Note:** This setting is ignored when using the container. You MUST use volume mapping instead.

Download settings

DOWNLOAD_PDF

This flag allows downloading companion PDF file for each audiobook (if there is any).

Allowed values:
- 'true'
- 'false' or anything else

> **Example config:**
> `DOWNLOAD_PDF=true`

DOWNLOAD_ANNOT

This flag allows downloading annotations (bookmarks) for each audiobook (if there is any).

Allowed values:
- 'true'
- 'false' or anything else

> **Example config:**
> `DOWNLOAD_ANNOT=true`

DOWNLOAD_COVERS

This flag allows downloading art covers for each audiobook.

Allowed values:
- 'true'
- 'false' or anything else

> **Example config:**
> `DOWNLOAD_COVERS=true`

DOWNLOAD_COVERS_SIZE

This parameter allows to specify the cover sizes to download for each audiobook. Multiple values are allowed. 500 seems the default size for every Audible audiobooks.

Allowed values, any mix of the following:
252 315 360 408 500 558 570 882 900 1215

Example config:
`DOWNLOAD_COVERS_SIZE=(500 1215)`

DOWNLOAD_WISHLIST

This flag allows to download your Audible account wishlist.

Allowed values:
- 'true'
- 'false' or anything else

Example config:
`DOWNLOAD_WISHLIST=false`

DOWNLOAD_JOBS

This setting is the concurrency level during Audiobook download. At this moment it doesn't do much because all audiobooks are not downloaded in parallel because of an issue with `audible-cli` (https://github.com/mkb79/audible-cli/issues/218)

It sets the parallel level inside `audible-cli` (for example it parallelizes download of covers, AXX, annotations for a single audiobook).

Example config:
`DOWNLOAD_JOBS=2`

DOWNLOAD_RETRIES

Numbers of retries if a download fails. Careful of not hammering Amazon servers by keeping this setting low as there is no cooldown in the script.

Example config:
`DOWNLOAD_RETRIES=3`

DOWNLOAD_DIR

AAX & AAXC Audible files will be downloaded here

Example config:
`DOWNLOAD_DIR=$HOME/Audible/MyDownloads`

Note: This setting is ignored when using the container. You MUST use volume mapping instead.

Metadata related settings

METADATA_PARALLEL

Parallelization setting for metadata processing. This parameter allows the user to specify the number of jobs for metadata extraction.

Allowed values: any number >= 1

Example config:
`METADATA_PARALLEL=3`

METADATA_SOURCE

This setting allow to specify the source of metadata. It can be `aax`, meaning that only metadata from AAX/AAXC files will be considered, or `all` which means metadata will be fetched from original AAX/AAXC files, but also from `mediainfo` tool extraction, and finally also from library file.

Allowed values:
- 'aax'
- 'all'

Example config:
`METADATA_SOURCE=all`

METADATA_TIKA

Use Tika for language detection if title description/comments is long enough.

(slow) For local java execution, the jar file **must be** in the same directory of this script.

(faster) For remote execution, just put the URL of your Tika server.

Allowed values:
- jar file name
- HTTP URL of Tika server

Example config:
`METADATA_TIKA=tika-app-2.9.2.jar`

Example config:
`METADATA_TIKA=http://mytikahost:9998`

METADATA_CLEAN_AUTHOR_PATTERN

I noticed that author/narrators metadata was not always clean, so I added a pattern matching. If you want to remove all authors that match this regex, set it here.
The script first split author between ',' and then remove all authors that match this regex (*this is a non-case-sensitive regex match*).
Basically any authors/narrators matching any of the regex words will be removed from the list. This applies also to narrators.

Allowed values: regex string
- '*' => keep all authors
- 'traducteur|traductrice|editeur|editrice' => remove all authors matching any of these words

Example config:
`METADATA_CLEAN_AUTHOR_PATTERN='traducteur|traductrice|editeur|editrice'`

METADATA_SINGLENAME_AUTHORS

If an audiobook has multiple authors, this flag allows the script to discard all authors that are identified by a single word name (like pseudo/nicknames/etc.).
This setting is ignored if there is only a single author.
This applies also to narrators.

Note: This was a dirty workaround to bad AudioBookShelf metadata parser. But it gets fixed.

Allowed values:
- 'true' or anything else => Keep single name authors - 'false' => Discard single name authors

Example config:
`METADATA_SINGLENAME_AUTHORS=false`

Conversion settings

CONVERT_BITRATE

This parameter sets the target bit rate of the converted file. Lower bit rates mean smaller sizes but also lower quality.

Allowed values: any string that can be parsed by `ffmpeg` (ex: 96k, 128k, etc.)

Example config:
`CONVERT_BITRATE=96k`

CONVERT_PARALLEL

Parallelization setting for file conversion. This parameter allows the user to specify the number of jobs for audiobook conversion. This is entirely independent of METADATA_PARALLEL.

Optimal setting depends on how many cores and memory you have.

Allowed values: any number >= 1

Example config:
`CONVERT_PARALLEL=3`

CONVERT_DECRYPTONLY

This flag allows to bypass all metadata enrichment and audio format conversion of the script. Only AAX/AAXC decryption will be done during the processing of audiobooks resulting in a playable mp4 file (this will also bypass file moving to destination library).

This can be considered as a debug flag. Keep 'false' if you don't know what you are doing.

Allowed values:
- 'true'
- 'false' or anything else

Example config:
`CONVERT_DECRYPTONLY=false`

File move settings

Available keys for the naming schemes

Key	Description
title	audiobook title
album	audiobook title
subtitle	audiobook subtitle
year	publish year
lang	language country code (e.g.: en, de, fr, etc.)
language	language country code (e.g.: en, de, fr, etc.)
genre	genre list (should not be used in any scheme)
artist	author
album_artist	author
composer	narrator
asin	ASIN
audible_asin	ASIN
copyright	Copyright text (should not be used in any scheme)
publisher	publisher (not relevant in any naming scheme)
comment	long description of the audiobooks (should not be used in any scheme)
series	Series name
mvnm	Series name

Key	Description
series-part	Series volume
mvin	Series volume

It is recommended to stick with the following ones:
- title
- subtitle
- lang
- artist
- asin
- series
- series-part

In short the script will produce dynamic directory names based on metadata and aggregated based on user settings.

Directory as follows: `DEST_BASE_DIR + DEST_DIR_NAMING_SCHEME_AUDIOBOOK + DEST_BOOKDIR_NAMING_SCHEME_AUDIOBOOK`

Audiobook filename: `DEST_BOOK_NAMING_SCHEME_AUDIOBOOK`

These settings can be confusing pay attention to the parameter names!

DEST_BASE_DIR

Base directory for converted files (will be created if it doesn't exist). If you want your converted files in a different location, change this setting.

> **Example config:**
> `DEST_BASE_DIR=$HOME/AudioBookShelf/audiobooks`
>
> **Note:** This setting is ignored when using the container. You MUST use volume mapping instead.

DEST_DIR_NAMING_SCHEME_AUDIOBOOK

Directory path for audiobooks, it will be created under DEST_BASE_DIR.

Example: `(artist series)` will produce `/Isaac Asimov/Foundation`.

Keep the keys inside parentheses. If a key is non-existent for an audiobook then it's ignored.

To have only one directory per audiobook then use empty value like `()` and look for next parameter to configure the naming scheme of the audiobook directory.

Allowed values:
- any combination of the available keys
- or empty ()

> **Example config will produce /artists/series directory:**
> `DEST_DIR_NAMING_SCHEME_AUDIOBOOK=(artist series)`

DEST_BOOKDIR_NAMING_SCHEME_AUDIOBOOK

Audiobook DIRECTORY naming scheme (**cannot be empty**).

This is the directory where the audiobook will be moved, it is created under DEST_DIR_NAMING_SCHEME_AUDIOBOOK.

Use `"%string"` to insert custom text in the file name example: `(series-part "% - " title)` will produce a directory named `1 - audiobook_title`.

Custom string parameters are ignored if they are in the start of the naming scheme, in previous example `(series-part "% - " title)` if 'series-part' is not defined for an audiobook then only 'title' will be used.

Keep the keys inside parentheses.

Allowed values:
- any combination of the available keys
- and custom user strings starting with '%' (example: '% -')

Example config:
```
DEST_BOOKDIR_NAMING_SCHEME_AUDIOBOOK=(series-part "% - " title)
```

DEST_BOOK_NAMING_SCHEME_AUDIOBOOK

Audiobook FILE naming scheme (**cannot be empty**).

Use `"%string"` to insert custom text in the file name example: `(series-part "% - " title)`.

The same rules apply here as in DEST_BOOKDIR_NAMING_SCHEME_AUDIOBOOK.

Allowed values:
- any combination of the available keys
- and custom user strings starting with '%' (example: '% -')

Example config:
```
DEST_BOOK_NAMING_SCHEME_AUDIOBOOK=(title)
```

DEST_DIR_OVERWRITE

Destination directory overwrite mode (**cannot be empty**)

Allowed values:
- **'true' or 'ignore'**: If audiobook destination directory exists then overwrite files in it, other files in directory are preserved
- **'remove'**: If audiobook destination directory exists then remove it and recreate it (all existing files in it will be deleted)
- **'keep'**: If audiobook destination directory exists then create a new one with an incremental suffix
- **'false' or any other value**: If audiobook destination directory exists then skip processing to next audiobook

Example config:
```
DEST_DIR_OVERWRITE=true
```

DEST_COPY_COVER

This flag allows the user to specify if cover image should be copied to audiobook destination directory. If the flag is set then a copy of the largest cover will be created in the audiobook destination directory. Does nothing if no covers are found.

Allowed values:
- 'true'
- 'false' or anything else

Example config:
```
DEST_COPY_COVER=true
```

DEST_COPY_PDF

This flag allows the user to specify if PDF should be copied to audiobook destination directory. Does nothing if no PDF exists.

Allowed values:
- 'true'
- 'false' or anything else

Example config:
```
DEST_COPY_PDF=true
```

DEST_COPY_CHAPTERS_FILE

This flag allows the user to specify if JSON chapters file should be copied to audiobook destination directory. Does nothing if no PDF exists.

Allowed values:
- 'true'
- 'false' or anything else

> **Example config:**
> `DEST_COPY_CHAPTERS_FILE=true`

DEST_COPY_ANNOT_FILE

This flag allows the user to specify if JSON annotations/bookmarks file should be copied to audiobook destination directory. Does nothing if no PDF exists.

Allowed values:
- 'true'
- 'false' or anything else

> **Example config:**
> `DEST_COPY_ANNOT_FILE=true`

CLEAN_TMPLOGS

Delete logs generated during the current run (old ones are kept) It is safe to keep to 'true'

Allowed values:
- 'true'
- 'false' or anything else

> **Example config:**
> `CLEAN_TMPLOGS=true`

KEEP_DOWNLOADS

This flag allows the script to delete downloaded files once they are converted and pushed to final target directory.

Only converted files may be deleted, not converted audiobook remaining are not deleted.

Allowed values:
- 'true' or anything else => Keep downloaded files.
- 'false' => Deleted only files that have been converted

> **Example config:**
> `KEEP_DOWNLOADS=true`

Debug settings

Parameters below are for debugging purposes (**default for all boolean parameters is 'false'**).

Keep all settings below to `false` for normal behavior.

Main debug flags

DEBUG Global debug flag, enable stepped updates and other debug features. Requires STATUS_FILE exists and populated.

Allowed values:
- 'true'
- 'false' or anything else

> **Example config:**
> `DEBUG=false`

DEBUG_DONT_UPDATE_LASTRUN Flag to tell if the script should update the STATUS_FILE. This is handy if you want to run a script multiple times for debugging purposes.

Allowed values:
- 'true'
- 'false' or anything else

Example config:
```
DEBUG_DONT_UPDATE_LASTRUN=false
```

DEBUG_STEP When DEBUG is enabled and STATUS_FILE contains an old date, this setting only increments the STATUS_FILE date with the specified period of time. This allows manual stepped runs over specified time periods.

Allowed values:
- '1 month'
- '2 week'
- '4 days'
- any time period understood by 'date' console command

Example config:
```
DEBUG_STEP="1 month"
```

DEBUG_SKIPDOWNLOADS This flag disables all Audible downloads for the current run. If other parts of the scripts are enabled then the scripts expect to find the correct download folder with all the required files in it.

Allowed values:
- 'true' => Disable ALL downloads
- 'false' or any other value => normal behavior

Example config:
```
DEBUG_SKIPDOWNLOADS=false
```

DEBUG_SKIPBOOKCONVERT This flag disables all audiobook conversions for the current run. If other parts of the scripts are enabled then the scripts expect to find the correct download folder with all the required files in it.

Allowed values:
- 'true' => disable audiobook conversion
- 'false' or any other value => normal behavior

Example config:
```
DEBUG_SKIPBOOKCONVERT=false
```

DEBUG_SKIPBOOKMETADATA This flag disables all audiobook metadata gathering for the current run. If other parts of the scripts are enabled then the scripts expect to find the correct download folder with all the required files in it.

Allowed values:
- 'true' => disable metadata gathering for audiobooks
- 'false' or any other value => normal behavior

Example config:
```
DEBUG_SKIPBOOKMETADATA=false
```

DEBUG_SKIPMOVEBOOKS This flag disables the last part of the script and no file will be moved to destination directory for the current run.

Allowed values:
- 'true' => disable moving audiobooks
- 'false' or any other value => normal behavior

Example config:
`DEBUG_SKIPMOVEBOOKS=false`

DEBUG_DONTEMBEDCOVER This flag, if enabled, prevents the art cover file to be embedded in the converted audiobook. It does not prevent the copy of the cover JPG file into destination directory (use this instead DEST_COPY_COVER).

Allowed values:
- 'true' => disable moving audiobooks
- 'false' or any other value => normal behavior

Example config:
`DEBUG_DONTEMBEDCOVER=false`

DEBUG_METADATA This flag allows more detailed debug output of metadata to be print during script execution. It also allows you to create a debug file with all the metadata extracted by `ffprobe` for each converted file.

Allowed values:
- 'true' => disable moving audiobooks
- 'false' or any other value => normal behavior

Example config:
`DEBUG_METADATA=false`

Audiobook debug samples

Debug samples are small AAX and AAXC files that will be used during audiobook conversions instead of using big original files. The metadata is still fetched from the original files and inserted in converted samples.

It is recommended to take the smallest AAX and the smallest AAXC files + voucher.

DEBUG_USEAAXSAMPLE Full path and file name of the AAX sample file. The file must come from your own Audible account or the script will not be able to process it.

Allowed values:
- `sample.aax` => file name (must be in BALD directory)
- 'false' => normal behavior

Example config:
`DEBUG_USEAAXSAMPLE=sample.aax`

Note: This setting is ignored when using the container. You MUST use volume mapping instead.

DEBUG_USEAAXCSAMPLE File name of the AAXC sample file. The file must come from your own Audible account or the script will not be able to process it. Do not forget to put aside the voucher file for it.

Allowed values:
- `sample.aaxc` => file name (must be in BALD directory)
- 'false' => normal behavior

Example config:
`DEBUG_USEAAXCSAMPLE=sample.aaxc`

Note: This setting is ignored when using the container. You MUST use volume mapping instead.

Anti-piracy notice

Note that this project does NOT 'crack' the DRM. It simply allows the user to use their own encryption key (fetched from Audible servers) to decrypt the audiobook in the same manner that the official audiobook playing software does.

Please only use this application for gaining full access to your own audiobooks for archiving/conversion/convenience. DeDRMed audiobooks should not be uploaded to open servers, torrents, or other methods of mass distribution. No help will be given to people doing such things. Authors, retailers, and publishers all need to make a living, so that they can continue to produce audiobooks for us to hear, and enjoy.

This blurb is borrowed from the https://apprenticealf.wordpress.com/ page.

License