audible

mkb79

Apr 29, 2024

TABLE OF CONTENTS

1 Installation Guide 3

2 Getting started 5

3 Marketplaces 7

4 Authorization (Login) 9

5 Authentication 13

6 Register a Audible device 15

7 Load and Save authentication data 17

8 Asynchron requests 21

9 Advanced Usage 23

10 Logging 29

11 External Audible API 31

12 Examples 47

13 Changelog 49

14 Modules Documentation 57

15 Indices and tables 93

Python Module Index 95

HTTP Routing Table 97

Index 99

Audible is a Python low-level interface to communicate with the non-publicly Audible API. It enables Python developers to create there own Audible services. Asynchronous communication with the Audible API is supported.

Note: For a basic command line interface take a look at my audible-cli package. This package supports:

- downloading audiobooks (aax/aaxc), cover, PDF and chapter file
- export library to csv files
- get activation bytes
- add own plugin commands

CHAPTER

ONE

INSTALLATION GUIDE

1.1 Requirements / Dependencies

Audible needs at least *Python 3.10*.

It depends on the following packages:

- beautifulsoup4
- httpx
- pbkdf2
- Pillow
- pyaes
- rsa

1.2 Installation

The easiest way to install the latest version from PyPI is by using pip:

```
pip install audible
```

You can also use Git to clone the repository from GitHub to install the latest development version:

```
git clone https://github.com/mkb79/audible.git
cd Audible
pip install .
```

Alternatively, install it directly from the GitHub repository:

```
pip install git+https://github.com/mkb79/audible.git
```

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

If you are new to Audible, this is the place to begin. The goal of this tutorial is to get you set-up and rolling with Audible. I won't go into too much detail here, just some important basics.

2.2 First Audible device

Before you can communicate with the non-publicly Audible Api, you need to authorize (login) yourself to Amazon (or Audible) and register a new "virtual" Audible device. Please make sure to select the correct Audible marketplace. An overview about all known Audible marketplaces and associated country codes be found at *Country Codes*.

```
import audible

# Authorize and register in one step
auth = audible.Authenticator.from_login(
    USERNAME,
    PASSWORD,
    locale=COUNTRY_CODE,
    with_username=False
)

# Save credentials to file
auth.to_file(FILENAME)
```

Important: Every device registration will be shown on the Amazon devices list. So only register once and reuse your authentication data or deregister the device with `auth.deregister_device()` before you close your session.

Note: If you have activated 2-factor-authentication for your Amazon account, you can append the current OTP to your password. This eliminates the need for a new OTP prompt.

Note: Set *with_username=True* to login with your pre-Amazon account (for US, UK or DE marketplace only).

Note: For security reasons in some cases you have to solve a Captcha and complete some extra steps. Please take a look at the *Authorization* section for more information.

2.3 Hello Library

After the device creation was successfully completed, you are ready to make your first API call. To fetch and print out all books from your Audible library (sorted by purchase date in descending order) you can do:

```
with audible.Client(auth=auth) as client:
    library = client.get(
        "1.0/library",
        num_results=1000,
        response_groups="product_desc, product_attrs",
        sort_by="-PurchaseDate"
    )
    for book in library["items"]:
        print(book)
```

Note: The information returned by the API depends on the requested *response_groups*. The response for the example above are very minimized. Please take a look at *GET /1.0/library* for all known *response_groups* and other parameter for the library endpoint.

2.4 Reuse authentication data

You can store your authentication data after an device registration with:

```
auth.to_file(FILENAME)
```

And load the data from file to reuse it with:

```
auth = audible.Authenticator.from_file(FILENAME)
```

CHAPTER

THREE

MARKETPLACES

3.1 General Information

Audible offers his service on 11 different marketplaces. You can read more about marketplaces here.

Note: Except website cookies, authentication data from device registration are valid for all marketplaces, no matter which marketplace are used.

Note: The Brazilian marketplace was added in mid-2023.

3.2 Country Codes

This app supports all marketplaces provided by Audible. For every marketplace a country code is associated.

Note: The country code of the selected marketplace is stored to file, when you save your authentication data. So, after loading this data from file, the stored country code is used by default.

Table 1: Marketplaces with country codes

Marketplace	Supported Countries	Country Code
Audible.com	US and all other countries not listed	us
Audible.ca	Canada	ca
Audible.co.uk	UK and Ireland	uk
Audible.co.au	Australia and New Zealand	au
Audible.fr	France, Belgium, Switzerland	fr
Audible.de	Germany, Austria, Switzerland	de
Audible.co.jp	Japan	jp
Audible.it	Italy	it
Audible.co.in	India	in
Audible.es	Spain	es
Audible.com.br	Brazil	br

3.3 The locale argument

The locale argument have the same meaning as the country code argument. Because of backward compatibility I didn't renamed the locale argument yet. So if you are asked for a *locale* than provide a country code from above.

Note: The country code for the Brazilian marketplace needs Audible > 0.8.2. How to use these marketplace with a previous version read this comment.

CHAPTER

FOUR

AUTHORIZATION (LOGIN)

4.1 Information

Clients are authorized using OpenID in Authorization Code Flow with PKCE. Once a client has successfully authorized to Amazon, they receive an *authorization code* for device registration to Audible/Amazon.

4.2 Authorization

For an example on authorization, please take a look at *Hello Library*.

4.2.1 CAPTCHA

Added in version v0.5.2: Init cookies added to login function to prevent CAPTCHAs in most cases.

Authorization requires answering a CAPTCHA in some cases. By default Pillow is used to show captcha and a user prompt will be provided to enter your answer, which looks like:

```
Answer for CAPTCHA:
```

A custom callback can be provided (for example submitting the CAPTCHA to an external service), like so:

```
def custom_captcha_callback(captcha_url):

    # Do some things with the captcha_url ...
    # maybe you can call webbrowser.open(captcha_url)
    # or simply print out the captcha_url

    return "My answer for CAPTCHA"

auth = audible.Authenticator.from_login(
    ...
    captcha_callback=custom_captcha_callback
)
```

4.2.2 2FA (OTP Code)

If two-factor authentication (2FA) is activated, a user prompt will be provided using *input* to enter your one time password (OTP), which looks like:

```
"OTP Code: "
```

A custom callback can be provided, like so:

```
def custom_otp_callback():

    # Do some things to insert otp code

    return "My answer for otp code"

auth = audible.Authenticator.from_login(
    ...
    otp_callback=custom_otp_callback
)
```

If you have to enter an OTP often and don't care about security, you can use the pyotp package with a custom callback like so:

```
from pyotp.totp import TOTP

def otp_callback():
    secret = "YOUR-AMAZON-OTP-SECRET"
    secret = secret.replace(" ", "")
    otp = TOTP(secret)
    return str(otp.now())
```

Another approach is to append the current OTP to the password.

4.2.3 CVF Code

If 2FA is deactivated and Amazon detects some security risks (too many logins in short times, etc.) you will be asked for a verify code (CVF). In that case, amazon sends you an email or SMS with a code, which you enter here:

```
"CVF Code: "
```

A custom callback can be provided, like so:

```
def custom_cvf_callback():

    # Do some things to insert cvf code

    return "My answer for cvf code"

auth = audible.Authenticator.from_login(
    ...
    cvf_callback=custom_cvf_callback
)
```

4.2.4 Approval Alert

Some users report that trying to authorize with audible gives them an approval alert and an email from amazon. Since audible v0.5 you will get a user prompt which looks like:

```
"Approval alert detected! Amazon sends you a mail."
"Please press enter when you approve the notification."
```

Please approve the email/SMS, and press any key to continue.

Added in version 0.5.1: Provide a custom callback with `approval_callback`

A custom callback can be provided, like so:

```
def custom_approval_callback():

    # You can let python check for the received Amazon mail and
    # open the approval link. The login function waits until
    # the callback function is executed. The returned value will be
    # ignored by the login function.


auth = audible.Authenticator.from_login(
    ...
    approval_callback=custom_approval_callback
    )
```

4.3 Authorization with external browser or program logic

Added in version v0.5.1: Login with external browser or program logic

To handle the login with a external browser or program logic you can do the following:

```
import audible

auth = audible.Authenticator.from_login_external(locale=COUNTRY_CODE)
```

By default, this code prints out the login url for the selected country code. Paste this url into a web browser or use it programatically to authorize yourself. You have to enter your credentials two times (because of missing init cookies). First time, the password can be a random one. Second time, you have to solve a captcha before you can submit the login form and you must use your correct password. After loggin in, you will end in an error page (not found). This is correct. Copy the url from the address bar from your browser and paste the url into the input field of the python code. It will look something like "https://www.amazon.{domain}/ap/maplanding?...&openid.oa2.authorization_code=..."

Note: If you have playwright installed and use the default `login_url_callback`, a new browser is opened, where you can authorize to your account.

Note: If you are using MacOS and have trouble insert the login result url, simply import the readline module in your script. See #34.

4.3.1 Custom callback

A custom callback can be provided (for example open the url in a webbrowser directly), like so:

```
def custom_login_url_callback(login_url):

    # Do some things with the login_url ...
    # maybe you can call webbrowser.open(login_url)
    # or simply print out the login_url

    return "The postlogin url"

auth = audible.Authenticator.from_login_external(
    ...
    login_url_callback=custom_login_url_callback
    )
```

CHAPTER

FIVE

AUTHENTICATION

5.1 API Authentication

Audible uses the *sign request* or the *bearer* method to authenticate the requests to the Audible API.

The authentication is done automatically when using the *audible.Authenticator*. Simply use the `Authenticator` with the *audible.Client* or *audible.AsyncClient* like so:

```
auth = audible.Authenticator.from_file(...)
client = audible.Client(auth=auth)
```

The Authenticator will try to use the sign request method if available. Otherwise the Authenticator will try the bearer method. If no method is available an exception is raised.

5.1.1 Sign request method

With the sign request method you gain unrestricted access to the Audible API. To use this method, you need the RSA private key and the adp_token from a *device registration*. This method is used by the Audible apps for iOS and Android too. A device registration is done automatically with *audible.Authenticator.from_login()* or *audible.Authenticator.from_login_external()*

Request signing is fairly straight-forward and uses a signed SHA256 digest. Headers look like:

```
x-adp-alg: SHA256withRSA:1.0
x-adp-signature: AAAAAAAA...:2019-02-16T00:00:01.000000000Z,
x-adp-token: {enc:...}
```

5.1.2 Bearer method

API requests with the bearer method have some restrictions. Some API call, like the *POST /1.0/content/(string:asin)/licenserequest*, doesn't work. To use the bearer method you need an access token and a client id. You receive the token after a device registration. Which values are valid for the client-id is unknown but 0 does work. An access token expires after 60 minutes. It can be renewed with a refresh token. A refresh token is obtained by a device registration only. Headers for the bearer method look like:

```
Authorization: Bearer Atna|...
client-id: 0
```

5.2 Website Authentication

To authenticate website requests you need the website cookies received from an authorization or device registration.

You can use the website cookies from an `Authenticator` with a `httpx.Client` or `httpx.AsyncClient` like so:

```
auth = audible.Authenticator.from_file(...)
with httpx.Client(cookies=auth.website_cookies) as client:
    resp = client.get("https://www.amazon.com/cpe/yourpayments/wallet?ref_=ya_d_c_pmt_mpo
↪")
    resp = client.get("https://www.audible.com")
```

Note: Website cookies are limited to the scope of a top level domain (e.g. com, de, ...). To set website cookies for another top level domain scope, you can call `auth.set_website_cookies_for_country(COUNTRY_CODE)`.

Warning: Set website cookies for another country will override the old ones. If you want to keep the new cookies, please make sure to save your authentication data.

5.3 Using Postman for authentication

Postman is a helpful utility to test API's.

To use Postman with the Audible API, every request needs to be authenticated. You can use the bearer method (with his limitions) with Postman out of the box.

Using the sign request method with Postman is possible, but needs some extra work.

HOWTO:

1. Install the postman_util_lib
2. Copy the content from the `pre-request-script` into the *Pre-request Scripts* Tab for the Collection or request
3. Create an Environment and define the variables *adp-token* and *private key* with the counterparts from the authentication data file

CHAPTER

SIX

REGISTER A AUDIBLE DEVICE

6.1 Register

Clients are obtaining additional authentication data and information after registration a "virtual" Audible device.

To authorize and register a new device you can do:

```
auth = audible.Authenticator.from_login(
    username,
    password,
    locale=country_code,
    with_username=False,
)
```

This will authorize you to your account and register an Audible device.

Important: Every device registration will be shown on the Amazon devices list. So only register once and save your credentials or deregister the device before you close your session.

6.2 Deregister

Authentication data obtained by a device registration are valid until deregister. Call `auth.deregister_device()` to deregister the current used device.

Call `auth.deregister_device(deregister_all=True)` to deregister **ALL** Audible devices. This function is helpful to remove hanging slots. This can happens if you registered a device and forgot to store the given authentication data or to deregister. This also deregister all other devices such as an Audible app on mobile devices. If you only want to remove one registration you can also open the amazon devices list on the the amazon website.

Important: Deregister needs an valid access token. The authentication data from a device registration contains a refresh token. With these token, an access token can be renewed with `auth.refresh_access_token()`.

CHAPTER

SEVEN

LOAD AND SAVE AUTHENTICATION DATA

7.1 Unencrypted Load/Save

Authentication data can be saved any time to file like so:

```
auth.to_file(FILENAME, encryption=False)
```

And can then be reused later like so:

```
auth = audible.Authenticator.from_file(FILENAME)
```

Note: The provided *FILENAME* is set as default when loading from or save to file. Simply run `auth.to_file()` to overwrite the previous loaded file.

Added in version v0.7.1: The *audible.Authenticator.to_dict()* and *audible.Authenticator.from_dict()* methods.

With *audible.Authenticator.to_dict()* you can get the authentication data as a dictionary. This enables you to implement your own save/load methods. Simply use the *audible.Authenticator.from_dict()* classmethod to load the data from the dictionary.

7.2 Encrypted Load/Save

This Client supports file encryption. The encryption algorithm used is symmetric AES in cipher-block chaining (CBC) mode. Currently json and bytes style output are supported.

Authentication data can be saved any time to encrypted file in json style like so:

```
auth.to_file(
    FILENAME,
    PASSWORD,
    encryption="json"
)
```

Or in bytes style like so:

```
auth.to_file(
    FILENAME,
    PASSWORD,
```

(continues on next page)

(continued from previous page)

```
    encryption="bytes"
)
```

When loading data from file, encryption style is auto detected. These files can be loaded like so:

```
auth = audible.Authenticator.from_file(
    FILENAME,
    PASSWORD
)
```

Note: Authenticator sets the last *FILENAME*, *PASSWORD* and *ENCRYPTION* as default when loading from or save to file. Simply run `auth.to_file()` to overwrite the previous loaded file with these settings.

7.3 Which data are saved?

The following data will be stored:

- website_cookies
- access_token
- locale_code (our country_code)
- access_token expiration timestamp
- refresh_token
- adp_token
- device_private_key
- store_authentication_cookie
- device_info data
- customer_info data
- activation_bytes
- with_username (`True` for pre-Amazon accounts else `False`)

Added in version 0.5.1: Stores `activation_bytes` to file (if they where fetched before).

Added in version v0.8.0: The `with_username` value

7.4 Advanced use of encryption/decryption

When saving authentication data, additional options can be provided with `auth.to_file(..., **kwargs)`. This data can be loaded with `auth = audible.Authenticator.from_file(..., **kwargs)`.

Following options are supported:

- key_size (default = 32)
- salt_marker (default = b"$")
- kdf_iterations (default = 1000)

- hashmod (default = Crypto.Hash.SHA256)

key_size may be 16, 24 or 32. The key is derived via the PBKDF2 key derivation function (KDF) from the password and a random salt of 16 bytes (the AES block size) minus the length of the salt header (see below).

The hash function used by PBKDF2 is SHA256 per default. You can pass a different hash function module via the *hashmod* argument. The module must adhere to the Python API for Cryptographic Hash Functions (PEP 247).

PBKDF2 uses a number of iterations of the hash function to derive the key, which can be set via the *kdf_iterations* keyword argumeent. The default number is 1000 and the maximum 65535.

The header and the salt are written to the first block of the encrypted output (bytes mode) or written as key/value pairs (dict mode). The header consist of the number of KDF iterations encoded as a big-endian word bytes wrapped by *salt_marker* on both sides. With the default value of *salt_marker* = *b'$'*, the header size is thus 4 and the salt 12 bytes. The salt marker must be a byte string of 1-6 bytes length. The last block of the encrypted output is padded with up to 16 bytes, all having the value of the length of the padding.

In json style all values are written as base64 encoded string.

7.5 Remove encryption

To remove encryption from file (or save as new file) simply load the encrypted file with *audible.Authenticator.from_file()* and save the data unencrypted. If the *Authenticator* can't load your data, you can try:

```
from audible.aescipher import remove_file_encryption

remove_file_encryption(
    encrypted_file=FILENAME,
    decrypted_file=FILENAME,
    password=PASSWORD_FOR_ENCRYPTED_FILE
)
```

CHAPTER

EIGHT

ASYNCHRON REQUESTS

This app supports asynchronous request using the httpx module. You can instantiate a async Client with:

```
async with audible.AsyncClient(auth=...) as client:
    ...
```

8.1 Example

```
import asyncio

import audible


# ASYNC FUNCTIONALITY
async def get_book_infos(client, asin):
    try:
        book = await client.get(
            path=f"library/{asin}",
            params={
                "response_groups": (
                    "contributors, media, price, reviews, product_attrs, "
                    "product_extended_attrs, product_desc, product_plan_details, "
                    "product_plans, rating, sample, sku, series, ws4v, origin, "
                    "relationships, review_attrs, categories, badge_types, "
                    "category_ladders, claim_code_url, is_downloaded, pdf_url, "
                    "is_returnable, origin_asin, percent_complete, provided_review"
                )
            },
        )
        return book
    except Exception as e:
        print(e)


async def main(auth):
    async with audible.AsyncClient(auth) as client:
        print(repr(client))

        library = await client.get(path="library", params={"num_results": 999})
```

(continues on next page)

(continued from previous page)

```
        asins = [book["asin"] for book in library["items"]]

        # books = await asyncio.gather(*(dl_book(asin) for asin in asins))
        tasks = []
        for asin in asins:
            tasks.append(asyncio.ensure_future(get_book_infos(client, asin)))
        books = await asyncio.gather(*tasks)

        for book in books:
            if book is not None:
                print(book["item"])
                print("\n", 40 * "-", "\n")


if __name__ == "__main__":
    # authenticate with login
    # don't stores any credentials on your system
    username = ""
    password = ""
    filename = ""

    auth = audible.Authenticator.from_login(username, password, locale="us")

    # store credentials to file
    auth.to_file(filename=filename, encryption="json", password=password)

    # save again
    auth.to_file()

    # load credentials from file
    auth = audible.Authenticator.from_file(filename=filename, password=password)
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main(auth))

    # deregister device
    auth.deregister_device()
```

CHAPTER

NINE

ADVANCED USAGE

9.1 Client classes

Here are some information about the `Client` and the `AsyncClient` classes.

9.1.1 Instantiate a client

A client needs at least an *audible.Authenticator* at instantiation. The following args and kwargs can be passed to the client instantiation:

- country_code (overrides the country code set in *audible.Authenticator*)
- headers (will be bypassed to the underlying httpx client)
- timeout (will be bypassed to the underlying httpx client)
- response_callback (custom response preparation - read more below)
- all other kwargs (will be bypassed to the underlying httpx client)

9.1.2 Make API requests

Both client classes have the following methods to send requests to the external API:

- get
- post
- delete
- put

The external Audible API offers currently two API versions, *0.0* and *1.0*. The Client use the *1.0* by default. So both terms are equal:

```
resp = client.get("library")
resp = client.get("1.0/library")
```

Each query parameter can be written as a separate keyword argument or you can merge them as a dict to the *params* keyword. So both terms are equal:

```
resp = client.get("library", response_groups="...", num_results=20)
resp = client.get(
    "library",
```

(continues on next page)

(continued from previous page)

```
    params={
        "response_groups"="...",
        "num_results"=20
    }
)
```

The external Audible API awaits a request body in JSON format. You have to provide the body as a dict to the Client. The Client converts and sends them in JSON style to the API. You can send them like so:

```
resp = client.post(
    "wishlist",
    body={"asin": ASIN_OF_BOOK_TO_ADD}
)
```

The Audible API responses are in JSON format. The client converts them to a and output them as a Python dict.

Note: For all known API endpoints take a look at *API Endpoints*.

9.1.3 Client responses

Added in version v0.8.0: The `response_callback` kwarg to client __init__, get, post, delete and put methods.

By default requesting the API with the client get, post, delete and put methods will call *audible.client.raise_for_status()* and try to convert the response with *audible.client.convert_response_content()* to a Python dict, which is finally returned.

If you want to implement your own response preparation, you can do:

```
def own_response_callback(resp):
    return resp

client = audible.Client(auth=..., response_callback=own_response_callback)
```

This will return the unprepared response (include headers).

9.1.4 Show/Change Marketplace

The currently selected marketplace can be shown with:

```
client.marketplace
```

The marketplace can be changed with:

```
client.switch_marketplace(COUNTRY_CODE)
```

9.1.5 Username/Userprofile

To get the profile for the user, which authentication data are used you can do this:

```
user_profile = client.get_user_profile()

# or from an Authenticator instance
auth.refresh_access_token()
user_profile = auth.user_profile()
```

To get the username only:

```
user_name = client.user_name
```

9.1.6 Switch User

If you work with multiple users you can do this:

```
# instantiate 1st user
auth = audible.Authenticator.from_file(FILENAME)

# instantiate 2nd user
auth2 = audible.Authenticator.from_file(FILENAME2)

# instantiate client with 1st user
client = audible.AudibleAPI(auth)
print(client.user_name)

# now change user with auth2
client.switch_user(auth2)
print(client.user_name)

# optional set default marketplace from 2nd user
client.switch_user(auth2, switch_to_default_marketplace=True)
```

9.1.7 Misc

The underlying Authenticator can be accessed via the *auth* attribute.

9.2 Authenticator classes

Deprecated since version v0.5.0: The `LoginAuthenticator` and the `FileAuthenticator`

Changed in version v0.6.0.

The `LoginAuthenticator` and the `FileAuthenticator` are removed from the Audible package.

Added in version v0.5.0: The `Authenticator` with the classmethods `from_file` and `from_login`

The `Authenticator.from_login()` classmethod is used to authorize an user and then authenticate requests with the received data. The `Authenticator.from_file()` classmethod is used to load previous saved authentication data.

With an Authenticator you can:

- Save credentials to file with `auth.to_file()`
- Deregister a previously registered device with `auth.deregister_device()`.
- Refresh an access token from a previously registered device with `auth.refresh_access_token()`.
- Get user profile with `auth.user_profile()`. Needs a valid access token.

To check if a access token is expired you can call:

```
auth.access_token_expired
```

Or to check the time left before token expires:

```
auth.access_token_expires
```

9.3 Activation Bytes

Added in version v0.4.0: Get activation bytes

Added in version v0.5.0: the `extract` param

To retrieve activation bytes an authentication `Authenticator` is needed.

The Activation bytes can be obtained like so:

```
activation_bytes = auth.get_activation_bytes()

# the whole activation blob can fetched with
auth.get_activation_bytes(extract=False)
```

The activation blob can be saved to file too:

```
activation_bytes = auth.get_activation_bytes(FILENAME)
```

Attention: Please only use this for gaining full access to your own audiobooks for archiving / converson / 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. Don't be a parasite.

9.4 PDF Url

PDF urls received by the Audible API don't work anymore. Authentication data are missing in the provided link. As a workaround you can do:

```
import audible
import httpx

asin = ASIN_FROM_BOOK
auth = audible.Authenticator.from_file(...)  # or Authenticator.from_login
tld = auth.locale.domain
```

(continues on next page)

(continued from previous page)

```
with httpx.Client(auth=auth) as client:
    resp = client.head(
        f"https://www.audible.{tld}/companion-file/{asin}"
    )
    url = resp.url
```

9.5 Decrypting license

Responses from the *POST /1.0/content/(string:asin)/licenserequest* endpoint contains the encrypted license (voucher).

To decrypt the license response you can do:

```
from audible.aescipher import decrypt_voucher_from_licenserequest

auth = YOUR_AUTH_INSTANCE
lr = RESPONSE_FROM_LICENSEREQUEST_ENPOINT
dlr = decrypt_voucher_from_licenserequest(auth, lr)
```

Attention: Please only use this for gaining full access to your own audiobooks for archiving / converson / 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. Don't be a parasite.

CHAPTER

TEN

LOGGING

You can use the Python logging module to log the ouput. You can get the logger with `logger = logging.getLogger("audible")`.

I implement a basic log helper, where you can set the basic behavior of logging. You can use it in this way:

```
from audible import log_helper

# set the log level for the audible package
log_helper.set_level(LEVEL)

# console logging
log_helper.set_console_logger(LEVEL)

# file logging
log_helper.set_file_logger(FILENAME, LEVEL)

# capture warnings
log_helper.capture_warnings()
```

The *LEVEL* argument for `set_console_logger()` and `set_file_logger()` are optional. If a *LEVEL* is provided, it must be equal or greater than the log level for the package. Otherwise console or file logger will log nothing.

Following levels are accepted:

- debug
- info
- warning
- error
- critical
- notset

You can use numeric levels too:

- 0 (notset)
- 10 (debug)
- 20 (info)
- 30 (warning)
- 40 (error)
- 50 (critical)

CHAPTER

ELEVEN

EXTERNAL AUDIBLE API

11.1 Documentation

There is currently no publicly available documentation about the Audible API.

There is a node client audible-api that has some endpoints documented, but does not provide information on authentication.

Luckily the Audible API is partially self-documenting, however the parameter names need to be known. Error responses will look like:

```
{
  "message": "1 validation error detected: Value 'some_random_string123' at 'numResults'␣
↪failed to satisfy constraint: Member must satisfy regular expression pattern: ^\\d+$"
}
```

Few endpoints have been fully documented, as a large amount of functionality is not testable from the app or functionality is unknown. Most calls need to be authenticated.

For *%s* substitutions the value is unknown or can be inferred from the endpoint. */1.0/catalog/products/%s* for example requires an *asin*, as in */1.0/catalog/products/B002V02KPU*.

Each bullet below refers to a parameter for the request with the specified method and URL.

Responses will often provide very little info without *response_groups* specified. Multiple response groups can be specified, for example: */1.0/catalog/products/B002V02KPU?response_groups=product_plan_details,media,review_attrs*. When providing an invalid response group, the server will return an error response but will not provide information on available response groups.

11.2 API Endpoints

GET /0.0/library/books

This API endpoint is deprecated. Please use *GET /1.0/library* instead.

Query Parameters

- **purchaseAfterDate** (*string*) – mm/dd/yyyy
- **sortByColumn** (*string*) – [SHORT_TITLE, strTitle, DOWNLOAD_STATUS, RUNNING_TIME, sortPublishDate, SHORT_AUTHOR, sortPurchDate, DATE_AVAILABLE]
- **sortInAscendingOrder** (*bool*) – [true, false]

11.2.1 Library

`GET /1.0/library`

The audible library of current user

Query Parameters

- **num_results** (*integer*) – (max: 1000)
- **page** (*integer*) – page
- **purchased_after** (*string*) – [RFC3339](https://tools.ietf.org/html/rfc3339) (e.g. *2000-01-01T00:00:00Z*)
- **title** (*string*) – a title
- **author** (*string*) – a author
- **response_groups** (*string*) – [contributors, customer_rights, media, price, product_attrs, product_desc, product_details, product_extended_attrs, product_plan_details, product_plans, rating, sample, sku, series, reviews, ws4v, origin, relationships, review_attrs, categories, badge_types, category_ladders, claim_code_url, in_wishlist, is_archived, is_downloaded, is_finished, is_playable, is_removable, is_returnable, is_visible, listening_status, order_details, origin_asin, pdf_url, percent_complete, periodicals, provided_review]
- **image_sizes** (*string*) – [1215,408,360,882,315,570,252,558,900,500]
- **sort_by** (*string*) – [-Author, -Length, -Narrator, -PurchaseDate, -Title, Author, Length, Narrator, PurchaseDate, Title]
- **status** (*string*) – [Active, Revoked] ('Active' is the default, 'Revoked' returns audiobooks the user has returned for a refund.)
- **parent_asin** (*string*) – asin
- **include_pending** (*string*) – [true, false]
- **marketplace** (*string*) – [e.g. AN7V1F1VY261K]
- **state_token** (*string*)

`GET /1.0/library/(string: asin)`

Parameters

- **asin** (*string*) – The asin of the book

Query Parameters

- **response_groups** (*string*) – [contributors, media, price, product_attrs, product_desc, product_details, product_extended_attrs, product_plan_details, product_plans, rating, sample, sku, series, reviews, ws4v, origin, relationships, review_attrs, categories, badge_types, category_ladders, claim_code_url, is_downloaded, is_finished, is_returnable, origin_asin, pdf_url, percent_complete, periodicals, provided_review]

`POST /1.0/library/item`

Request JSON Object

- **asin** (*string*) – The asin of the book

`POST /1.0/library/item`

Request JSON Object

- `asin`

`PUT /1.0/library/item`

Add an (AYCL) item to the library

Request JSON Object

- `asin`

`POST /1.0/library/item/`(*param1*)`/`
param2

Parameters

- `param1`
- `param2`

Request JSON Object

- `unknown`

`POST /1.0/library/collections/`(*param1*)`/channels/`
param2

Parameters

- `param1`
- `param2`

Request JSON Object

- `customer_id`
- `marketplace`

`POST /1.0/library/collections/`(*param1*)`/products/`
param2

Parameters

- `param1`
- `param2`

Request JSON Object

- `channel_id`

`GET /1.0/library/collections`

Query Parameters

- `customer_id`
- `marketplace`

`POST /1.0/library/collections`

Request JSON Object

- `collection_type`

`GET /1.0/library/collections/`(*param1*)

Parameters

- **param1**

Query Parameters

- **customer_id**
- **marketplace**
- **page_size**
- **continuation_token**

`GET /1.0/library/collections/`(*param1*)`/products`

Parameters

- **param1**

Query Parameters

- **customer_id**
- **marketplace**
- **page_size**
- **continuation_token**
- **image_sizes**

11.2.2 Catalog

Categories

`GET /1.0/catalog/categories`

Query Parameters

- **response_groups** – [category_metadata, products]
- **products_plan** – [Enterprise, RodizioFreeBasic, AyceRomance, AllYouCanEat, US Minerva, Universal, AmazonEnglish, ComplimentaryOriginalMemberBenefit, Radio, SpecialBenefit, Rodizio]
- **products_in_plan_timestamp**
- **products_num_results**
- **runtime_length_min**
- **content_level**
- **content_type**
- **categories_num_levels** (*int*) – (greater than or equal to 1)
- **ids** – \d+(,\d+)*
- **root** – [InstitutionsHpMarketing, ChannelsConfigurator, AEReadster, ShortsPrime, ExploreBy, RodizioBuckets, EditorsPicks, ClientContent, RodizioGenres, AmazonEnglishProducts, ShortsSandbox, Genres, Curated, ShortsIntroOutroRemoval, Shorts, RodizioEpisodesAndSeries, ShortsCurated]

GET /1.0/catalog/categories/(*category_id*)

Parameters

- **category_id**

Query Parameters

- **image_dpi** (*int*)
- **image_sizes**
- **image_variants**
- **products_in_plan_timestamp**
- **products_num_results** (*int*)
- **products_plan** – [Enterprise, RodizioFreeBasic, AyceRomance, AllYouCanEat, AmazonEnglish, ComplimentaryOriginalMemberBenefit, Radio, SpecialBenefit, Rodizio]
- **products_sort_by** – [-ReleaseDate, ContentLevel, -Title, AmazonEnglish, AvgRating, BestSellers, -RuntimeLength, ReleaseDate, ProductSiteLaunchDate, -ContentLevel, Title, Relevance, RuntimeLength]
- **reviews_num_results** (*int*)
- **reviews_sort_by** – [MostHelpful, MostRecent]

Quers products_not_in_plan_timestamp

Products

GET /1.0/catalog/products/(string: *asin*)

Parameters

- **asin** (*string*) – The asin of the book

Query Parameters

- **image_dpi**
- **image_sizes**
- **response_groups** – [contributors, media, price, product_attrs, product_desc, product_details, product_extended_attrs, product_plan_details, product_plans, rating, sample, sku, series, reviews, relationships, review_attrs, category_ladders, claim_code_url, provided_review, rights, customer_rights]
- **reviews_num_results** – \d+ (max: 10)
- **reviews_sort_by** – [MostHelpful, MostRecent]
- **asins**

GET /1.0/catalog/products

Query Parameters

- **asins**
- **image_sizes** – [1215,408,360,882,315,570,252,558,900]
- **response_groups** – [sku,product_attrs,rating,product_extended_attrs,media,sample,product_plans,product_plan_

`GET /1.0/catalog/products/(string: asin)/reviews`

Parameters

- **asin** (*string*) – The asin of the book

Query Parameters

- **sort_by** – [MostHelpful, MostRecent]
- **num_results** (*int*) – (max: 50)
- **page** (*int*)

`GET /1.0/catalog/products`

Query Parameters

- **author**
- **browse_type**
- **category_id** (*int*) – \d+(,\d+)*
- **disjunctive_category_ids**
- **image_dpi** (*int*)
- **image_sizes**
- **in_plan_timestamp**
- **keywords**
- **narrator**
- **not_in_plan_timestamp**
- **num_most_recent**
- **num_results** (*int*) – (max: 50)
- **page** (*int*)
- **plan** – [Enterprise, RodizioFreeBasic, AyceRomance, AllYouCanEat, AmazonEnglish, ComplimentaryOriginalMemberBenefit, Radio, SpecialBenefit, Rodizio]
- **products_since_timestamp**
- **products_sort_by** – [-ReleaseDate, ContentLevel, -Title, AmazonEnglish, AvgRating, BestSellers, -RuntimeLength, ReleaseDate, ProductSiteLaunchDate, -ContentLevel, Title, Relevance, RuntimeLength]
- **publisher**
- **response_groups** – [contributors, media, price, product_attrs, product_desc, product_extended_attrs, product_plan_details, product_plans, rating, review_attrs, reviews, sample, series, sku]
- **reviews_num_results** (*int*) – (max: 10)
- **reviews_sort_by** – [MostHelpful, MostRecent]
- **title**

`GET /1.0/catalog/products/(string: asin)/sims`

Parameters

- **asin** (*string*) – The asin of the book

Query Parameters

- **category_image_variants**
- **image_dpi**
- **image_sizes**
- **in_plan_timestamp**
- **language**
- **not_in_plan_timestamp**
- **num_results** (*int*) – (max: 50)
- **plan** – [Enterprise, RodizioFreeBasic, AyceRomance, AllYouCanEat, AmazonEnglish, ComplimentaryOriginalMemberBenefit, Radio, SpecialBenefit, Rodizio]
- **response_groups** – [contributors, media, price, product_attrs, product_desc, product_extended_attrs, product_plans, rating, review_attrs, reviews, sample, sku]
- **reviews_num_results** (*int*) – (max: 10)
- **reviews_sort_by** – [MostHelpful, MostRecent]
- **similarity_type** – [InTheSameSeries, ByTheSameNarrator, RawSimilarities, ByTheSameAuthor, NextInSameSeries]

11.2.3 Collections

`GET /1.0/collections`

Query Parameters

- **state_token** – [ey...]
- **visibility_types** – [Private, Discoverable]

`POST /1.0/collections`

Create a new collection

Request JSON Object

- **name**
- **asins** – []
- **description**

Response JSON Object

- **collection_id**
- **creation_date**
- **customer_id**
- **marketplace**

`GET /1.0/collections/`(*collection_id*)

Parameters

- **collection_id**

`PUT /1.0/collections/`(*collection_id*)

Modify a collection

Parameters

- **collection_id**

Request JSON Object

- **state_token**
- **collection_id**
- **name**
- **description**

Response JSON Object

- **state_token**
- **collection_id**
- **name**
- **description**

`GET /1.0/collections/`(*collection_id*)`/items`

Parameters

- **collection_id** – e.g __FAVORITES

Query Parameters

- **response_groups** – [always-returned]

`POST /1.0/collections/`(*collection_id*)`/items`

Add item(s) to a collection

Parameters

- **collection_id**

Request JSON Object

- **collection_id**
- **asins** – []

Response JSON Object

- **description**
- **name**
- **num_items_added** (*int*)
- **state_token**

11.2.4 Orders

`GET /1.0/orders`

Returns order history from at least the past 6 months. Supports pagination.

Query Parameters

- **unknown**

`POST /1.0/orders`

Request JSON Object

- **asin** (*string*)
- **audiblecreditapplied** (*boolean*) – will specify whether to use available credits or default payment method.

Example request body

```
{
  "asin": "B002V1CB2Q",
  "audiblecreditapplied": "false"
}
```

11.2.5 Wishlist

`GET /1.0/wishlist`

Query Parameters

- **num_results** (*int*) – (max: 50)
- **page** (*int*) – (wishlist start at page 0)
- **locale** (*string*) – e.g. de-DE
- **response_groups** – [contributors, media, price, product_attrs, product_desc, product_extended_attrs, product_plan_details, product_plans, rating, sample, sku, customer_rights, relationships]
- **sort_by** – [-Author, -DateAdded, -Price, -Rating, -Title, Author, DateAdded, Price, Rating, Title]

`POST /1.0/wishlist`

Request JSON Object

- **asin** (*string*) – The asin of the book to add

Status Codes

- 201 Created – Returns the *Location* to the resource.

Example request body

```
{
  "asin": "B002V02KPU"
}
```

`DELETE /1.0/wishlist/(string: asin)`

Parameters

- **asin** (*string*) – The asin of the book

Status Codes

- 204 No Content – Removes the item from the wishlist using the given *asin*.

`GET /1.0/badges/progress`

Query Parameters

- **locale** – en_US
- **response_groups** – brag_message
- **store** – [AudibleForInstitutions, Audible, AmazonEnglish, Rodizio]

11.2.6 Badges

`GET /1.0/badges/metadata`

Query Parameters

- **locale** – en_US
- **response_groups** – all_levels_metadata

11.2.7 Content

`POST /1.0/content/(string: asin)/licenserequest`

Parameters

- **asin** (*string*) – The asin of the book

Request JSON Object

- **use_adaptive_bit_rate** (*boolean*) – [true, false]
- **quality** (*string*) – [High, Normal]
- **chapter_titles_type** (*string*) – [Tree, Flat]
- **response_groups** (*string*) – [chapter_info, content_reference, last_position_heard, pdf_url, ad_insertion, certificate]
- **consumption_type** (*string*) – [Streaming, Offline, Download]
- **spatial** (*boolean*) – [true, false]
- **supported_media_features** (*dict*) – [codecs, drm_types]
- **codecs** (*list*) – [mp4a.40.2, mp4a.40.42, ec+3, ac-4]
- **drm_types** (*list*) – [Mpeg, PlayReady, Hls, Dash, Adrm, FairPlay, Widevine, HlsCmaf]
- **num_active_offline_licenses** (*integer*) – (max: 10)

Example request body

```
{
    "quality": "High",
    "response_groups": "chapter_info,content_reference,last_position_heard,pdf_url,␣
↪ad_insertion, certificate",
    "consumption_type": "Download",
    "supported_media_features":
        {
            "codecs": [
                "mp4a.40.2",
                "mp4a.40.42",
                "ec+3",
                "ac-4"
            ],
            "drm_types": [
                "Mpeg",
                "PlayReady",
                "Hls",
                "Dash",
                "Adrm",
                "FairPlay",
                "Widevine",
                "HlsCmaf",
            ]
        },
    "spatial": false
}
```

For a succesful request, returns JSON body with *content_url*.

GET /1.0/content/(string: *asin*)/metadata

Parameters

- **asin** (*string*) – the asin of the book

Query Parameters

- **response_groups** – [chapter_info, always-returned, content_reference, content_url]
- **acr**
- **quality** – [High, Normal]
- **chapter_titles_type** – [Tree, Flat]
- **drm_type** – [Mpeg, PlayReady, Hls, Dash, FairPlay, Widevine, HlsCmaf, Adrm]

POST /1.0/content/(string: *asin*)/drmlicense

Parameters

- **asin** (*string*) – The asin of the book

Request JSON Object

- **licenseChallenge** (*string*) – The license challenge
- **asin** (*string*) – The asin of the book
- **consumption_type** (*string*) – “Download”

- **drm_type** (*string*) – "FairPlay"

Response JSON Object

- **license** (*string*) – The encrypted license

`GET 1.0/content/FairPlay/certificate`

Response JSON Object

- **certificate** (*string*) – The base64 encoded FairPlay certificate

11.2.8 Account

`GET /1.0/account/information`

Query Parameters

- **response_groups** – [delinquency_status, customer_benefits, customer_segments, subscription_details_payment_instrument, plan_summary, subscription_details, directed_ids]
- **source** – [Credit, Enterprise, RodizioFreeBasic, AyceRomance, AllYouCanEat, AmazonEnglish, ComplimentaryOriginalMemberBenefit, Radio, SpecialBenefit, Rodizio]

11.2.9 Customer

`GET /1.0/customer/information`

Query Parameters

- **response_groups** – [migration_details, subscription_details_rodizio, subscription_details_premium, customer_segment, subscription_details_channels]

`GET /1.0/customer/status`

Query Parameters

- **response_groups** – [benefits_status, member_giving_status, prime_benefits_status, prospect_benefits_status]

`GET /1.0/customer/freetrial/eligibility`

11.2.10 Stats

`GET /1.0/stats/aggregates`

Query Parameters

- **daily_listening_interval_duration** – ([012]?[0-9])|(30) (0 to 30, inclusive)
- **daily_listening_interval_start_date** – YYYY-MM-DD (e.g. *2019-06-16*)
- **locale** – en_US
- **monthly_listening_interval_duration** – 0?[0-9]|1[012] (0 to 12, inclusive)
- **monthly_listening_interval_start_date** – YYYY-MM (e.g. *2019-02*)
- **response_groups** – [total_listening_stats]
- **store** – [AudibleForInstitutions, Audible, AmazonEnglish, Rodizio]

`GET /1.0/stats/status/finished`

Query Parameters

- **asin** – asin
- **start_date** – [RFC3339](https://tools.ietf.org/html/rfc3339) (e.g. *2000-01-01T00:00:00Z*)

`POST /1.0/stats/status/finished`

Request JSON Object

- **start_date**
- **status**
- **continuation_token**

`PUT /1.0/stats/events`

Request JSON Object

- **stats**

Example request body

```
{
    "stats" : [
        {
            "download_start" : {
                "country_code" : "de",
                "download_host" : "xxxxx.cloudfront.net",
                "user_agent" : "Audible, iPhone, 3.35.1 (644), iPhone XS (iPhone11,
↪2), 238 GB, iOS, 14.1, Wifi",
                "request_id" : "xxxxxxxxxxxx",
                "codec" : "AAX_44_128",
                "source" : "audible_iPhone"
            },
            "social_network_site" : "Unknown",
            "event_type" : "DownloadStart",
            "listening_mode" : "Offline",
            "local_timezone" : "Europe\/Berlin",
            "asin_owned" : false,
            "playing_immersion_reading" : false,
            "audio_type" : "FullTitle",
            "event_timestamp" : "2020-10-23T21:29:06.985Z",
            "asin" : "xxxxxxx",
            "store" : "Audible",
            "delivery_type" : "Download"
        }
    ]
}
```

11.2.11 Misc

`GET /1.0/annotations/lastpositions`

Query Parameters

- **asins** – asin (comma-separated), e.g. ?asins=B01LWUJKQ7,B01LWUJKQ7,B01LWUJKQ7

`PUT /1.0/lastpositions/(string: asin)`

Parameters

- **asin** (*string*) – the asin of the book

Request JSON Object

- **acr** – obtained by *POST /1.0/content/(string:asin)/licenserequest*
- **asin**
- **position_ms**

`GET /1.0/pages/(string: param1)`

Parameters

- **param1** (*string*) – [ios-app-home]

Query Parameters

- **image_dpi** (*int*) – [489]
- **local_time** – [2022-01-01T12:00:00+01:00]
- **locale** – en-US
- **os** – [15.2]
- **reviews_num_results**
- **reviews_sort_by**
- **response_groups** – [media, product_plans, view, product_attrs, contributors, product_desc, sample]
- **session_id** – [123-1234567-1234567]
- **surface** – [iOS]

`GET /1.0/recommendations`

Query Parameters

- **category_image_variants**
- **category_image_variants**
- **image_dpi**
- **image_sizes**
- **in_plan_timestamp**
- **language**
- **not_in_plan_timestamp**
- **num_results** (*int*) – (max: 50)

- **plan** – [Enterprise, RodizioFreeBasic, AyceRomance, AllYouCanEat, AmazonEnglish, ComplimentaryOriginalMemberBenefit, Radio, SpecialBenefit, Rodizio]
- **response_groups** – [contributors, media, price, product_attrs, product_desc, product_extended_attrs, product_plan_details, product_plans, rating, sample, sku]
- **reviews_num_results** (*int*) – (max: 10)
- **reviews_sort_by** – [MostHelpful, MostRecent]

GET /1.0/user/settings

Query Parameters

- **setting_name** (*string*) – [captionsEnabled]

GET /1.0/app/upgradestatus

Query Parameters

- **version** – [3.68]
- **app_id** – [A2CZJZGLK2JJVM]
- **operating_system** – [iOS15.4]

GET https://cde-ta-g7g.amazon.com/FionaCDEServiceEngine/sidecar

Returns the clips, notes and bookmarks of a book

Query Parameters

- **type** (*string*) – ["AUDI"]
- **key** (*string*) – asin of the book

CHAPTER

TWELVE

EXAMPLES

Here are some examples and ideas how to use this app. Everyone who will provide some examples are welcome.

Print number of books for every marketplace:

```
import audible

auth = audible.Authenticator.from_file(filename)
client = audible.Client(auth)
country_codes = ["de", "us", "ca", "uk", "au", "fr", "jp", "it", "in"]

for country in country_codes:
    client.switch_marketplace(country)
    library = client.get("library", num_results=1000)
    asins = [book["asin"] for book in library["items"]]
    print(f"Country: {client.marketplace.upper()} | Number of books: {len(asins)}")
    print(34* "-")
```

Get listening statistics aggragated month-over-month from 2021-03 to 2021-06:

```
import audible

auth = audible.Authenticator.from_file(filename)
client = audible.Client(auth)
with audible.Client(auth=auth) as client:
     stats = client.get(
         "1.0/stats/aggregates",
         monthly_listening_interval_duration="3", #number of months to aggragate for
         monthly_listening_interval_start_date="2021-03", #start month for aggragation
         store="Audible")
```

CHAPTER

THIRTEEN

CHANGELOG

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog.

13.1 Unreleased

13.1.1 Misc

- Drop support for Python 3.8 and 3.9
- Add support for Python 3.12

13.2 [0.9.1] - 2023-09-27

13.3 Bugfix

- Fix login issues on brazilian marketplace.
- Fix a `RecursionError` which occurs when checking the length of an `Authenticator` instance.

13.4 [0.9.0] - 2023-09-27

13.5 Bugfix

- Multiple fixes for XXTEA encryption/decryption in metadata module.

13.5.1 Added

- Add brazilian marketplace.
- Login function now checks for a `verification-code-form` tag in login HTML page.

13.5.2 Changes

- Drop support for Python 3.7.

13.5.3 Misc

- First step to refactor code.
- Switch project to poetry.
- Using nox and ruff for tests and linting.

13.6 [0.8.2] - 2022-05-25

13.6.1 Changed

- Allow httpx v0.23.x to fix a bug in httpx

13.7 [0.8.1] - 2022-04-20

13.7.1 Bugfix

- fix a bug in `Client.delete` and `AsyncClient.delete` method

13.8 [0.8.0] - 2022-04-11

13.8.1 Added

- full support of pre-Amazon accounts (e.g. refresh access token, deregister device)
- `Client` and `AsynClient` now accepts session kwargs which are bypassed to the underlying httpx Client
- a `respone_callback` can now be set to `Client` and `AsyncClient` class to allow custom preparation of response output
- An absolut url (e.g. https://cde-ta-g7g.amazon.com/FionaCDEServiceEngine/sidecar) can now be passed to a client `get`, `post`, `delete` and `put` method as the `path` arg. So in most cases the client `raw_request` method is not needed anymore.

13.8.2 Changed

- rename (and rework) `Client._split_kwargs` to `Client._prepare_params`

13.9 [0.7.2] - 2022-03-27

13.9.1 Bugfix

- fix a bug in registration url

13.10 [0.7.1] - 2022-03-27

13.10.1 Added

- `Authenticator.from_dict` to instantiate an `Authenticator` from dict and `Authenticator.to_dict` to get authentication data as dict

13.10.2 Bugfix

- register a new device with `with_username=True` results in a server error due to wrong registration domain

13.11 [0.7.0] - 2021-10-25

13.11.1 Bugfix

- make sure activation bytes has 8 bytes, otherwise append ‚0‘ in front until 8 bytes are reached
- make sure metadata1 has 8 bytes, otherwise append ‚0‘ in front until 8 bytes are reached
- If installed, use playwright to login with external browser. Please read here how to install playwright. Then use `audible.Authenticator.from_login_external(COUNTRY_CODE)` for login.
- fix login issues

13.12 [0.6.0] - 2021-10-21

13.12.1 Bugfix

- Fix a bug when searching for „resend-approval-link“ in login page

13.12.2 Changed

- switched to `auth_code_flow` when login (gives an auth code instead of an access token for security purposes)
- `Authenticator.from_login` and `Authenticator.from_login_external` now always register a new device
- `Authenticator` now refreshes `access_token` (when needed) before deregister the device
- now simulate Audible app version 3.56.2 under iOS version 15.0.0
- login process now auto-detect next request method and url

13.12.3 Misc

- Correct documentation
- Update example download_books_aaxc.py
- Bump httpx to `v0.20.*`

13.12.4 Remove

- `LoginAuthenticator` and `FileAuthenticator`
- `Authenticator.register_device`, `Authenticator.re_login` and `Authenticator.re_login_external`

13.13 [0.5.5] - 2021-07-22

13.13.1 Misc

- switch from httpx 0.16.x to 0.18.x

13.13.2 Added

- logging error messages during login

13.13.3 Changed

- extend allowed chars by email check during login
- instead of raising an exception, invalid email will now be logged as warning

13.13.4 Misc

- Add description to the docs, to handling 2FA

13.14 [0.5.4] - 2021-02-28

13.14.1 Added

- Provide a custom serial when login
- Login with Audible username instead of Amazon account for US, UK and DE markteplace

13.14.2 Bugfix

- register a device on Australian marketplace

13.14.3 Misc

- Redesign Module documentation
- Rework description of audible-cli package in documentation

13.15 [0.5.3] - 2021-01-25

13.15.1 Added

- function `activation_bytes.fetch_activation_sign_auth`
- Spain marketplace

13.15.2 Changed

- `activation_bytes.get_activation_bytes` uses the new `fetch_activation_sign_auth` function, if `signing` auth method is available. Otherwise activation bytes will be fetched the old way with a `player_token`.

13.16 [0.5.2] - 2021-01-08

13.16.1 Added

- Add initial cookies to login function to prevent captcha requests in most cases.

13.17 [0.5.1] - 2021-01-05

13.17.1 Added

- Fetched activation bytes (with `extract=True` argument) will be stored to `activation_bytes` attribute of Authenticator class instance for now. Ignore existing activation bytes and force refresh with `auth.get_activation_bytes(force_refresh=True)`
- `activation_bytes` will be loaded from and save to file. Saved auth files are **not backward compatible** to previous audible versions so keep old files save.
- Add `Client.raw_request` and `AsyncClient.raw_request` method.
- Provide a custom Callback with `approval_callback` keyword argument when login.
- Add classmethod `Authenticator.from_login_external` and method `Authenticator.re_login_external`.
- Add `login_external` function to login.py

13.17.2 Misc

- Add description how to use pyotp with custom otp callback to docs
- Add description how to use login external to docs

13.18 [0.5.0] - 2020-12-07

13.18.1 Added

- Added support to output the whole activation blob instead of the extracted activation bytes with `get_activation_bytes(extract=False, ...)`.
- Added support to fetch website cookies for another country with `Authenticator.set_website_cookies_for_country`.
- Added `Client.put` and `AsyncClient.put`.
- Added support to solve approval alerts during login

13.18.2 Changed

- The `FileAuthenticator` has been deprecated, use classmethod `Authenticator.from_file` instead.
- The `Authenticator` don't inherit from MutableMapping anymore
- The `Authenticator` sets allowed instance attributes at creation to `None`, not allowed attributes will raise an Exception
- The `LoginAuthenticator` has been deprecated, use classmethod `Authenticator.from_login` instead.
- Changed internal code base for encryption and decryption metadata. Moved the related code to `metadata.py`.

13.18.3 Remove

- deprecated `AudibleAPI`

13.18.4 Misc

- Added more docstrings and type hints to code base
- Added support to install Sphinx documentation dependencies with `pip install audible[docs]`.
- Added a guide to use authentication with Postman.
- Rework documentation.
- Added `.readthedocs.yml` config file
- Added module description (autodoc) to docs
- Uses `httpx` 0.16.* for now

13.19 [0.4.4] - 2020-10-25

13.19.1 Bugfix

- Set `padding=„none“` when decrypting license voucher

CHAPTER

FOURTEEN

MODULES DOCUMENTATION

14.1 audible package

class audible.**AsyncClient**(*auth*, *country_code=None*, *headers=None*, *timeout=10*, *response_callback=None*, ***session_kwargs*)

Bases: *BaseClient*[AsyncClient]

Parameters

- **auth** (*Authenticator*)
- **country_code** (str | None)
- **headers** (Union[Headers, Mapping[str, str], Mapping[bytes, bytes], Sequence[Tuple[str, str]], Sequence[Tuple[bytes, bytes]], None])
- **timeout** (int)
- **response_callback** (Callable[[Response], Any] | None)
- **session_kwargs** (Any)

async **close**()

Return type
None

async **delete**(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

async **get**(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

async post(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

async put(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

class audible.**Authenticator**

Bases: Auth

Audible Authenticator class.

Note: A new class instance have to be instantiated with *Authenticator.from_login()* or *Authenticator.from_file()*.

Added in version v0.8: The with_username attribute.

Note: Auth data saved with v0.8 or later can not be loaded with versions less than v0.8! If an auth file for a pre-Amazon account (with_username=True) was created with v0.7.1 or v0.7.2 set *auth.with_username* to *True* and save the data again. After this deregistration, refreshing access tokens and requesting cookies for another domain will work for pre-Amazon accounts.

---

access_token: str | None = None

property access_token_expired: bool

property access_token_expires: timedelta

activation_bytes: str | None = None

adp_token: str | None = None

auth_flow(*request*)

Auth flow to be executed on every request by `httpx`.

Parameters
request (`Request`) – The request made by `httpx`.

Yields
The next request

Raises
AuthFlowError – If no auth flow is available.

Return type
`Generator[Request, Response, None]`

property available_auth_modes: list[str]

crypter: *AESCipher* | None = None

customer_info: dict[str, Any] | None = None

deregister_device(*deregister_all=False*)

Parameters
deregister_all (`bool`)

Return type
`Any`

device_info: dict[str, Any] | None = None

device_private_key: str | None = None

encryption: str | bool | None = None

expires: float | None = None

filename: Optional[Path] = None

classmethod from_dict(*data*, *locale=None*)

Instantiate an Authenticator from authentication file.

Added in version v0.7.1.

Parameters

- **data** (`dict[str, Any]`) – A dictionary with the authentication data
- **locale** (`Union[str, Locale, None]`) – The country code of the Audible marketplace to interact with. If `None` the country code from file is used.

Return type
Authenticator

Returns
A new Authenticator instance.

classmethod from_file(*filename*, *password=None*, *locale=None*, *encryption=None*, ***kwargs*)

Instantiate an Authenticator from authentication file.

Added in version v0.5.0.

Parameters

- **filename** (Union[str, Path]) – The name of the file with the authentication data.
- **password** (str | None) – The password of the authentication file.
- **locale** (Union[str, *Locale*, None]) – The country code of the Audible marketplace to interact with. If None the country code from file is used.
- **encryption** (bool | str | None) – The encryption style to use. Can be json or bytes. If None, encryption will be auto detected.
- ****kwargs** (Any) – Keyword arguments are passed to the *AESCipher* class. See below.

Keyword Arguments

- **key_size** (*int, Optional*)
- **salt_marker** (*Optional[bytes]*)
- **kdf_iterations** (int, optional)
- **hashmod**
- **mac**

Return type
Authenticator

Returns
A new Authenticator instance.

Raises
FileEncryptionError – If file ist encrypted without providing a password

classmethod from_login(*username*, *password*, *locale*, *serial=None*, *with_username=False*, *captcha_callback=None*, *otp_callback=None*, *cvf_callback=None*, *approval_callback=None*)

Instantiate a new Authenticator with authentication data from login.

Added in version v0.5.0.

Added in version v0.5.4: The serial argument The with_username argument

Parameters

- **username** (str) – The Amazon email address.
- **password** (str) – The Amazon password.
- **locale** (Union[str, *Locale*]) – The country_code or *audible.localization.Locale* instance for the marketplace to login.
- **serial** (str | None) – The device serial. If None a custom one will be created.
- **with_username** (bool) – If True login with Audible username instead of Amazon account.
- **captcha_callback** (Callable[[str], str] | None) – A custom callback to handle captcha requests during login.
- **otp_callback** (Callable[[], str] | None) – A custom callback to handle one-time password requests during login.
- **cvf_callback** (Callable[[], str] | None) – A custom callback to handle verify code requests during login.
- **approval_callback** (Callable[[], Any] | None) – A custom Callable for handling approval alerts.

Return type
Authenticator

Returns
An *Authenticator* instance.

classmethod from_login_external(*locale*, *serial=None*, *with_username=False*, *login_url_callback=None*)

Instantiate a new Authenticator from login with external browser.

Added in version v0.5.1.

Added in version v0.5.4: The serial argument The with_username argument

Parameters

- **locale** (Union[str, *Locale*]) – The country_code or *audible.localization.Locale* instance for the marketplace to login.
- **serial** (str | None) – The device serial. If None a custom one will be created.
- **with_username** (bool) – If True login with Audible username instead of Amazon account.
- **login_url_callback** (Callable[[str], str] | None) – A custom Callable for handling login with external browsers.

Return type
Authenticator

Returns
An *Authenticator* instance.

get_activation_bytes(*filename=None*, *extract=True*, *force_refresh=False*)

Get Activation bytes from Audible.

Parameters

- **filename** (Union[Path, str, None]) – [Optional] filename to save the activation blob
- **extract** (Literal[True, False]) – [Optional] if True, returns the extracted activation bytes otherwise the whole activation blob
- **force_refresh** (bool) – [Optional] if True, existing activation bytes in auth file will be ignored and new activation bytes will be requested from server.

Return type
str | bytes

Returns
The activation bytes

Added in version 0.5.1: The force_refresh argument. Fetched activation bytes are now stored to *Authentenicator.activation_bytes*.

locale: Optional[*Locale*] = None

refresh_access_token(*force=False*)

Parameters
force (bool)

Return type
None

refresh_token: str | None = None

requires_request_body: bool = True

set_website_cookies_for_country(*country_code*)

Parameters
country_code (str)

Return type
None

sign_request(*request*)

Sign a request.

Deprecated since version 0.5.0: Use `self._apply_signing_auth_flow()` instead.

Parameters
request (Request)

Return type
None

store_authentication_cookie: dict[str, Any] | None = None

to_dict()

Returns authentication data as dict. :rtype: dict[str, Any]

Added in version 0.7.1.

Added in version v0.8: The returned dict now contains the *with_username* attribute

to_file(*filename=None*, *password=None*, *encryption='default'*, *indent=4*, *set_default=True*, ***kwargs*)

Save authentication data to file.

Added in version 0.5.1: Save activation bytes to auth file

Added in version v0.8: The saved file now contains the *with_username* attribute

Parameters

- **filename** (Union[Path, str, None])
- **password** (str | None)
- **encryption** (bool | str)
- **indent** (int)
- **set_default** (bool)
- **kwargs** (Any)

Return type
None

user_profile()

Return type
dict[str, Any]

website_cookies: dict[str, Any] | None = None

with_username: bool | None = False

class audible.**Client**(*auth*, *country_code=None*, *headers=None*, *timeout=10*, *response_callback=None*, ***session_kwargs*)

Bases: *BaseClient*[Client]

Parameters

- **auth** (*Authenticator*)
- **country_code** (str | None)
- **headers** (Union[Headers, Mapping[str, str], Mapping[bytes, bytes], Sequence[Tuple[str, str]], Sequence[Tuple[bytes, bytes]], None])
- **timeout** (int)
- **response_callback** (Callable[[Response], Any] | None)
- **session_kwargs** (Any)

close()

Return type
None

delete(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

get(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

post(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

put(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

14.1.1 audible.activation_bytes module

audible.activation_bytes.**extract_activation_bytes**(*data*)

Extracts the activation bytes from activation blob.

Parameters
data (bytes) – A activation blob returned by fetch_activation function

Return type
str

Returns
The extracted activation bytes.

Raises
ValueError – If *data* is not a valid activation blob.

audible.activation_bytes.**fetch_activation**(*player_token*)

Fetches the activation blob with player token from Audible server.

Parameters
player_token (str) – A player token returned by get_player_token function.

Return type
bytes

Returns
The activation blob.

audible.activation_bytes.**fetch_activation_sign_auth**(*auth*)

Fetches the activation blob with sign authentication from Audible server.

Parameters
auth (*Authenticator*) – A Authenticator instance with valid adp_token and device_private_cert.

Return type
bytes

Returns
The activation blob.

Raises
AuthFlowError – If no valid auth method is available.

audible.activation_bytes.**get_activation_bytes**(*auth*, *filename=None*, *extract=True*)

Fetches the activation blob from Audible and extracts the bytes.

Parameters

- **auth** (*Authenticator*) – The Authenticator.
- **filename** (str | Path | None) – The filename to save the activation blob (Default: None).
- **extract** (Literal[True, False]) – If True, returns the extracted activation bytes otherwise the whole activation blob (Default: True).

Return type
str | bytes

Returns
The activation bytes or activation blob.

Raises
AuthFlowError – If no valid auth method is available.

audible.activation_bytes.**get_player_id**()

Build a software player Id.

Return type
str

audible.activation_bytes.**get_player_token**(*auth*)

Fetches a player token for further authentication.

Parameters
auth (*Authenticator*) – The Authenticator.

Return type
str

Returns
The player token.

Raises
Exception – If *playerToken* not found in response url query.

14.1.2 audible.aescipher module

class audible.aescipher.**AESCipher**(*password*, ***, *key_size=32*, *salt_marker=b'$'*, *kdf_iterations=1000*, *hashmod=<built-in function openssl_sha256>*, *mac=<module 'hmac' from '/home/docs/.asdf/installs/python/3.12.0/lib/python3.12/hmac.py'>*)

Bases: object

Encrypt/Decrypt data using password to generate key.

The encryption algorithm used is symmetric AES in cipher-block chaining (CBC) mode.

The key is derived via the PBKDF2 key derivation function (KDF) from the password and a random salt of 16 bytes (the AES block size) minus the length of the salt header (see below). The hash function used by PBKDF2 is SHA256 per default. You can pass a different hash function module via the hashmod argument. The module must adhere to the Python API for Cryptographic Hash Functions (PEP 247). PBKDF2 uses a number of iterations of the hash function to derive the key, which can be set via the kdf_iterations keyword argument. The default number is 1000 and the maximum 65535. The header and the salt are written to the first block of the encrypted output (bytes mode) or written as key/value pairs (dict mode). The header consist of the number of

KDF iterations encoded as a big-endian word bytes wrapped by `salt_marker` on both sides. With the default value of `salt_marker = b'$'`, the header size is thus 4 and the salt 12 bytes. The salt marker must be a byte string of 1-6 bytes length. The last block of the encrypted output is padded with up to 16 bytes, all having the value of the length of the padding. All values in dict mode are written as base64 encoded string.

password

The password for encryption/decryption.

key_size

The size of the key. Can be `16`, `24` or `32` (Default: 32).

salt_marker

The salt marker with max. length of 6 bytes (Default: $).

kdf_iterations

The number of iterations of the hash function to derive the key (Default: 1000).

hashmod

The hash method to use (Default: sha256).

mac

The mac module to use (Default: hmac).

Parameters

- **password** (`str`) – The password for encryption/decryption.
- **key_size** (`int`) – The size of the key. Can be `16`, `24` or `32` (Default: 32).
- **salt_marker** (`bytes`) – The salt marker with max. length of 6 bytes (Default: $).
- **kdf_iterations** (`int`) – The number of iterations of the hash function to derive the key (Default: 1000).
- **hashmod** – The hash method to use (Default: sha256).
- **mac** – The mac module to use (Default: hmac).

Raises

- **ValueError** – If *salt_marker* is not one to six bytes long.
- **ValueError** – If *kdf_iterations* is greater than 65535.
- **TypeError** – If type of *salt_marker* is not bytes.

from_bytes(*data*)

Decrypts data previously encrypted with *AESCipher.to_bytes()*.

Parameters
data (`bytes`) – The encrypted data in bytes style.

Return type
`str`

Returns
The decrypted data.

from_dict(*data*)

Decrypts data previously encrypted with *AESCipher.to_dict()*.

Parameters
data (`dict[str, str]`) – The encrypted data in json style.

Return type
str

Returns
The decrypted data.

from_file(*filename*, *encryption='json'*)

Loads and decrypts data from given file.

Parameters

- **filename** (Path) – The name of the file to load the data from.
- **encryption** (str) – The encryption style which where used. Can be json or bytes (Default: json).

Return type
str

Returns
The decrypted data.

Raises
ValueError – If *encryption* is not json or bytes.

to_bytes(*data*)

Encrypts data in bytes style.

The output bytes contains the (packed) salt, iv and ciphertext.

Parameters
data (str) – The data to encrypt.

Return type
bytes

Returns
The encrypted data in dict style.

to_dict(*data*)

Encrypts data in dict style.

The output dict contains the base64 encoded (packed) salt, iv and ciphertext key/value pairs and an info key/value pair with additional encryption information.

Parameters
data (str) – The data to encrypt.

Return type
dict[str, str]

Returns
The encrypted data in dict style.

to_file(*data*, *filename*, *encryption='json'*, *indent=4*)

Encrypts and saves data to given file.

Parameters

- **data** (str) – The data to encrypt.
- **filename** (Path) – The name of the file to save the data to.
- **encryption** (str) – The encryption style to use. Can be json or bytes (Default: json).

- **indent** (int) – The indention level when saving in json style (Default: 4).

Raises
ValueError – If *encryption* is not json or bytes.

Return type
None

audible.aescipher.**aes_cbc_decrypt**(*key*, *iv*, *encrypted_data*, *padding='default'*)

Decrypts data encrypted in cipher block chaining mode of operation.

Parameters

- **key** (bytes) – The AES key used at encryption.
- **iv** (bytes) – The initialization vector used at encryption.
- **encrypted_data** (bytes) – The encrypted data to decrypt.
- **padding** (str) – Can be default or none (Default: default)

Return type
str

Returns
The decrypted data.

audible.aescipher.**aes_cbc_encrypt**(*key*, *iv*, *data*, *padding='default'*)

Encrypts data in cipher block chaining mode of operation.

Parameters

- **key** (bytes) – The AES key.
- **iv** (bytes) – The initialization vector.
- **data** (str) – The data to encrypt.
- **padding** (str) – Can be default or none (Default: default)

Return type
bytes

Returns
The encrypted data.

audible.aescipher.**create_salt**(*salt_marker*, *kdf_iterations*)

Creates the header and salt for the *derive_from_pbkdf2()* function.

The header consist of the number of KDF iterations encoded as a big-endian word bytes wrapped by salt_marker on both sides. The random salt has a length of 16 bytes (the AES block size) minus the length of the salt header.

Parameters

- **salt_marker** (bytes)
- **kdf_iterations** (int)

Return type
tuple[bytes, bytes]

audible.aescipher.**decrypt_voucher_from_licenserequest**(*auth*, *license_response*)

Decrypt the voucher from license request response.

Parameters

- **auth** (*Authenticator*) – The Authenticator.
- **license_response** (dict[str, Any]) – The response content from a *POST /1.0/content/(string:asin)/licenserequest* request.

Return type
dict[str, Any]

Returns
The decrypted license voucher with needed key and iv.

Raises
Exception – If device info or customer info data is missing.

Note: A device registration is needed to use the auth instance for a license request and to obtain the needed device data

---

audible.aescipher.**derive_from_pbkdf2**(*password*, ***, *key_size*, *salt*, *kdf_iterations*, *hashmod*, *mac*)

Creates an AES key with the PBKDF2 key derivation class.

Parameters

- **password** (str)
- **key_size** (int)
- **salt** (bytes)
- **kdf_iterations** (int)

Return type
bytes

audible.aescipher.**detect_file_encryption**(*filename*)

Detect the encryption format from an authentication file.

Parameters
filename (Path) – The name for the authentication file.

Return type
Optional[Literal[False, 'json', 'bytes']]

Returns
False if file is not encrypted otherwise the encryption format.

audible.aescipher.**pack_salt**(*header*, *salt*)

Combines the header and salt created by *create_salt()* function.

Parameters

- **header** (bytes)
- **salt** (bytes)

Return type
bytes

audible.aescipher.**remove_file_encryption**(*source*, *target*, *password*, ***kwargs*)

Removes the encryption from an authentication file.

Please try to load the authentication file with *audible.Authenticator.from_file()* and save the authentication data as a unencrypted file first. Use this function as fallback if you ran into any error.

Parameters

- **source** (str | Path) – The encrypted authentication file.
- **target** (str | Path) – The filename for the decrypted file.
- **password** (str) – The password for the encrypted authentication file.
- ****kwargs** (Any) – keyword args supported by *AESCipher*

Raises
ValueError – If source is not encrypted.

Return type
None

audible.aescipher.**unpack_salt**(*packed_salt*, *salt_marker*)

Unpack salt and kdf_iterations from previous created and packed salt.

Parameters

- **packed_salt** (bytes)
- **salt_marker** (bytes)

Return type
tuple[bytes, int]

14.1.3 audible.auth module

class audible.auth.**Authenticator**

Bases: Auth

Audible Authenticator class.

Note: A new class instance have to be instantiated with *Authenticator.from_login()* or *Authenticator.from_file()*.

Added in version v0.8: The with_username attribute.

Note: Auth data saved with v0.8 or later can not be loaded with versions less than v0.8! If an auth file for a pre-Amazon account (with_username=True) was created with v0.7.1 or v0.7.2 set *auth.with_username* to *True* and save the data again. After this deregistration, refreshing access tokens and requesting cookies for another domain will work for pre-Amazon accounts.

---

access_token: str | None = None

property access_token_expired: bool

property access_token_expires: timedelta

activation_bytes: str | None = None

adp_token: str | None = None

auth_flow(*request*)

Auth flow to be executed on every request by `httpx`.

Parameters
request (`Request`) – The request made by `httpx`.

Yields
The next request

Raises
AuthFlowError – If no auth flow is available.

Return type
`Generator[Request, Response, None]`

property available_auth_modes: list[str]

crypter: *AESCipher* | None = None

customer_info: dict[str, Any] | None = None

deregister_device(*deregister_all=False*)

Parameters
deregister_all (`bool`)

Return type
`Any`

device_info: dict[str, Any] | None = None

device_private_key: str | None = None

encryption: str | bool | None = None

expires: float | None = None

filename: Optional[Path] = None

classmethod from_dict(*data*, *locale=None*)

Instantiate an Authenticator from authentication file.

Added in version v0.7.1.

Parameters

- **data** (`dict[str, Any]`) – A dictionary with the authentication data
- **locale** (`Union[str, Locale, None]`) – The country code of the Audible marketplace to interact with. If `None` the country code from file is used.

Return type
Authenticator

Returns
A new Authenticator instance.

classmethod from_file(*filename*, *password=None*, *locale=None*, *encryption=None*, ***kwargs*)

Instantiate an Authenticator from authentication file.

Added in version v0.5.0.

Parameters

- **filename** (Union[str, Path]) – The name of the file with the authentication data.
- **password** (str | None) – The password of the authentication file.
- **locale** (Union[str, *Locale*, None]) – The country code of the Audible marketplace to interact with. If None the country code from file is used.
- **encryption** (bool | str | None) – The encryption style to use. Can be json or bytes. If None, encryption will be auto detected.
- ****kwargs** (Any) – Keyword arguments are passed to the *AESCipher* class. See below.

Keyword Arguments

- **key_size** (*int, Optional*)
- **salt_marker** (*Optional[bytes]*)
- **kdf_iterations** (int, optional)
- **hashmod**
- **mac**

Return type
Authenticator

Returns
A new Authenticator instance.

Raises
FileEncryptionError – If file ist encrypted without providing a password

classmethod from_login(*username*, *password*, *locale*, *serial=None*, *with_username=False*, *captcha_callback=None*, *otp_callback=None*, *cvf_callback=None*, *approval_callback=None*)

Instantiate a new Authenticator with authentication data from login.

Added in version v0.5.0.

Added in version v0.5.4: The serial argument The with_username argument

Parameters

- **username** (str) – The Amazon email address.
- **password** (str) – The Amazon password.
- **locale** (Union[str, *Locale*]) – The country_code or *audible.localization.Locale* instance for the marketplace to login.
- **serial** (str | None) – The device serial. If None a custom one will be created.
- **with_username** (bool) – If True login with Audible username instead of Amazon account.
- **captcha_callback** (Callable[[str], str] | None) – A custom callback to handle captcha requests during login.
- **otp_callback** (Callable[[], str] | None) – A custom callback to handle one-time password requests during login.
- **cvf_callback** (Callable[[], str] | None) – A custom callback to handle verify code requests during login.
- **approval_callback** (Callable[[], Any] | None) – A custom Callable for handling approval alerts.

Return type
Authenticator

Returns
An *Authenticator* instance.

classmethod from_login_external(*locale*, *serial=None*, *with_username=False*, *login_url_callback=None*)

Instantiate a new Authenticator from login with external browser.

Added in version v0.5.1.

Added in version v0.5.4: The serial argument The with_username argument

Parameters

- **locale** (Union[str, *Locale*]) – The country_code or *audible.localization.Locale* instance for the marketplace to login.
- **serial** (str | None) – The device serial. If None a custom one will be created.
- **with_username** (bool) – If True login with Audible username instead of Amazon account.
- **login_url_callback** (Callable[[str], str] | None) – A custom Callable for handling login with external browsers.

Return type
Authenticator

Returns
An *Authenticator* instance.

get_activation_bytes(*filename=None*, *extract=True*, *force_refresh=False*)

Get Activation bytes from Audible.

Parameters

- **filename** (Union[Path, str, None]) – [Optional] filename to save the activation blob
- **extract** (Literal[True, False]) – [Optional] if True, returns the extracted activation bytes otherwise the whole activation blob
- **force_refresh** (bool) – [Optional] if True, existing activation bytes in auth file will be ignored and new activation bytes will be requested from server.

Return type
str | bytes

Returns
The activation bytes

Added in version 0.5.1: The force_refresh argument. Fetched activation bytes are now stored to *Authentenicator.activation_bytes*.

locale: Optional[*Locale*] = None

refresh_access_token(*force=False*)

Parameters
force (bool)

Return type
None

refresh_token: str | None = None

requires_request_body: bool = True

set_website_cookies_for_country(*country_code*)

Parameters
country_code (str)

Return type
None

sign_request(*request*)

Sign a request.

Deprecated since version 0.5.0: Use `self._apply_signing_auth_flow()` instead.

Parameters
request (Request)

Return type
None

store_authentication_cookie: dict[str, Any] | None = None

to_dict()

Returns authentication data as dict. :rtype: dict[str, Any]

Added in version 0.7.1.

Added in version v0.8: The returned dict now contains the *with_username* attribute

to_file(*filename=None*, *password=None*, *encryption='default'*, *indent=4*, *set_default=True*, ***kwargs*)

Save authentication data to file.

Added in version 0.5.1: Save activation bytes to auth file

Added in version v0.8: The saved file now contains the *with_username* attribute

Parameters

- **filename** (Union[Path, str, None])
- **password** (str | None)
- **encryption** (bool | str)
- **indent** (int)
- **set_default** (bool)
- **kwargs** (Any)

Return type
None

user_profile()

Return type
dict[str, Any]

website_cookies: dict[str, Any] | None = None

with_username: bool | None = False

audible.auth.**refresh_access_token**(*refresh_token*, *domain*, *with_username=False*)

Refreshes an access token.

Parameters

- **refresh_token** (str) – The refresh token obtained after a device registration.
- **domain** (str) – The top level domain of the requested Amazon server (e.g. com).
- **with_username** (bool) – If True uses *audible* domain instead of *amazon*.

Return type
dict[str, Any]

Returns
A dict with the new access token and expiration timestamp.

Note: The new access token is valid for 60 minutes.

Added in version v0.8: The with_username argument

audible.auth.**refresh_website_cookies**(*refresh_token*, *domain*, *cookies_domain*, *with_username=False*)

Fetches website cookies for a specific domain.

Parameters

- **refresh_token** (str) – The refresh token obtained after a device registration.
- **domain** (str) – The top level domain of the requested Amazon server (e.g. com, de, fr).
- **cookies_domain** (str) – The top level domain scope for the cookies (e.g. com, de, fr).
- **with_username** (bool) – If True uses *audible* domain instead of *amazon*.

Return type
dict[str, str]

Returns
The requested cookies for the Amazon and Audible website for the given *cookies_domain* scope.

Added in version v0.8: The with_username argument

audible.auth.**sign_request**(*method*, *path*, *body*, *adp_token*, *private_key*)

Helper function who creates signed headers for http requests.

Parameters

- **path** (str) – The requested http url path and query.
- **method** (str) – The http request method (GET, POST, DELETE, ...).
- **body** (bytes) – The http message body.
- **adp_token** (str) – The adp token obtained after a device registration.
- **private_key** (str) – The rsa key obtained after device registration.

Return type
dict[str, str]

Returns
A dict with the signed headers.

audible.auth.**user_profile**(*access_token*, *domain*)

Returns user profile from Amazon.

Parameters

- **access_token** (str) – The valid access token for authentication.
- **domain** (str) – The top level domain of the requested Amazon server (e.g. com, de, fr).

Return type
dict[str, Any]

Returns
The Amazon user profile for the authenticated user.

Raises
Exception – If the user profile is malformed

audible.auth.**user_profile_audible**(*access_token*, *domain*)

Returns user profile from Audible.

Parameters

- **access_token** (str) – The valid access token for authentication.
- **domain** (str) – The top level domain of the requested Audible server (e.g. com, de, fr).

Return type
dict[str, Any]

Returns
The Audible user profile for the authenticated user.

Raises
Exception – If the user profile is malformed

14.1.4 audible.client module

class audible.client.**AsyncClient**(*auth*, *country_code=None*, *headers=None*, *timeout=10*, *response_callback=None*, ***session_kwargs*)

Bases: *BaseClient*[AsyncClient]

Parameters

- **auth** (*Authenticator*)
- **country_code** (str | None)
- **headers** (Union[Headers, Mapping[str, str], Mapping[bytes, bytes], Sequence[Tuple[str, str]], Sequence[Tuple[bytes, bytes]], None])
- **timeout** (int)
- **response_callback** (Callable[[Response], Any] | None)
- **session_kwargs** (Any)

async close()

Return type
None

async delete(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

async get(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

async post(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

async put(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

class audible.client.**BaseClient**(*auth*, *country_code=None*, *headers=None*, *timeout=10*, *response_callback=None*, ***session_kwargs*)

Bases: Generic[ClientT]

Parameters

- **auth** (*Authenticator*)
- **country_code** (str | None)
- **headers** (Union[Headers, Mapping[str, str], Mapping[bytes, bytes], Sequence[Tuple[str, str]], Sequence[Tuple[bytes, bytes]], None])

- **timeout** (int)
- **response_callback** (Callable[[Response], Any] | None)
- **session_kwargs** (Any)

property auth: *Authenticator*

abstract delete(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

abstract get(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

get_user_profile()

Return type
dict[str, Any]

property marketplace: str

abstract post(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

abstract put(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

raw_request(*method*, *url*, *, *stream=False*, *apply_auth_flow=False*, *apply_cookies=False*, ***kwargs*)

Sends a raw request with the underlying httpx Client.

This method ignores a set api_url and allows send request to custom hosts. The raw httpx response will be returned.

Parameters

- **method** (str) – The http request method.
- **url** (str) – The url to make requests to.
- **stream** (Literal[True, False]) – If *True*, streams the response
- **apply_auth_flow** (bool) – If *True*, the Authenticator.auth_flow() will be applied to the request.
- **apply_cookies** (bool) – If *True*, website cookies from Authenticator.website_cookies will be added to request headers.
- ****kwargs** (Any) – keyword args supported by httpx.AsyncClient.stream, httpx.Client.stream, httpx.AsyncClient.request, httpx.Client.request.

Return type
Response | Coroutine[Any, Any, Response] | AbstractContextManager[Response] | AbstractAsyncContextManager[Response]

Returns
An unprepared httpx Response object.

Added in version v0.5.1.

switch_marketplace(*country_code*)

Parameters
country_code (str)

Return type
None

switch_user(*auth*, *switch_to_default_marketplace=False*)

Parameters

- **auth** (*Authenticator*)
- **switch_to_default_marketplace** (bool)

Return type
None

property user_name: str

class audible.client.**Client**(*auth*, *country_code=None*, *headers=None*, *timeout=10*, *response_callback=None*, ***session_kwargs*)

Bases: *BaseClient*[Client]

Parameters

- **auth** (*Authenticator*)
- **country_code** (str | None)

- **headers** (Union[Headers, Mapping[str, str], Mapping[bytes, bytes], Sequence[Tuple[str, str]], Sequence[Tuple[bytes, bytes]], None])
- **timeout** (int)
- **response_callback** (Callable[[Response], Any] | None)
- **session_kwargs** (Any)

close()

Return type
None

delete(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

get(*path*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

post(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

put(*path*, *body*, *response_callback=None*, ***kwargs*)

Parameters

- **path** (str)
- **body** (Any)
- **response_callback** (Callable[[Response], Any] | None)
- **kwargs** (dict[str, Any])

Return type
Any

audible.client.**convert_response_content**(*resp*)

Parameters
resp (Response)

Return type
Any

audible.client.**default_response_callback**(*resp*)

Parameters
resp (Response)

Return type
Any

audible.client.**raise_for_status**(*resp*)

Parameters
resp (Response)

Return type
None

14.1.5 audible.exceptions module

exception audible.exceptions.**AuthFlowError**

Bases: Exception

Raised if no auth method available.

exception audible.exceptions.**BadRequest**(*resp*, *data*)

Bases: *StatusError*

Raised when status code 400 is returned.

Typically, when at least one search parameter was not provided.

Parameters

- **resp** (Response)
- **data** (Any)

exception audible.exceptions.**FileEncryptionError**

Bases: Exception

Raised if something is wrong with file encryption.

exception audible.exceptions.**NetworkError**

Bases: *RequestError*

Raised if there is an issue with the network (i.e. requests.ConnectionError).

exception audible.exceptions.**NoRefreshToken**

Bases: Exception

Raised if refresh token is needed but not provided.

exception audible.exceptions.**NotFoundError**(*resp*, *data*)

Bases: *StatusError*

Raised if no result is found.

Parameters

- **resp** (Response)
- **data** (Any)

exception audible.exceptions.**NotResponding**

Bases: *RequestError*

Raised if the API request timed out.

exception audible.exceptions.**RatelimitError**(*resp*, *data*)

Bases: *StatusError*

Raised if ratelimit is hit.

Parameters

- **resp** (Response)
- **data** (Any)

exception audible.exceptions.**RequestError**

Bases: Exception

Base class for all errors.

exception audible.exceptions.**ServerError**(*resp*, *data*)

Bases: *StatusError*

Raised if the api service is having issues.

Parameters

- **resp** (Response)
- **data** (Any)

exception audible.exceptions.**StatusError**(*resp*, *data*)

Bases: *RequestError*

Base class for all errors except NotResponding and RatelimitDetectedError.

Parameters

- **resp** (Response)
- **data** (Any)

exception audible.exceptions.**Unauthorized**(*resp*, *data*)

Bases: *StatusError*

Raised if you passed invalid credentials.

Parameters

- **resp** (Response)
- **data** (Any)

exception audible.exceptions.**UnexpectedError**(*resp*, *data*)

Bases: *StatusError*

Raised if the error was not caught.

Parameters

- **resp** (Response)
- **data** (Any)

14.1.6 audible.localization module

class audible.localization.**Locale**(*country_code=None*, *domain=None*, *market_place_id=None*)

Bases: object

Adjustments for the different marketplaces who are provided by Audible.

You can try to autodetect_locale if your marketplace is not in templates`.

Parameters

- **country_code** (str | None)
- **domain** (str | None)
- **market_place_id** (str | None)

property **country_code: str**

property **domain: str**

property **market_place_id: str**

to_dict()

Return type
dict[str, str]

audible.localization.**autodetect_locale**(*domain*)

Try to automatically detect correct settings for marketplace.

Needs the top level domain of the audible page to continue with (e.g. co.uk, co.jp) and returns results found.

Parameters
domain (str) – The top level domain for the Audible marketplace to detect settings for (e.g. com).

Return type
dict[str, str]

Returns
The settings for the found Audible marketplace.

Raises

- **ConnectError** – If site does not exist or network error raises.
- **Exception** – If marketplace or country code can't be found.

audible.localization.**search_template**(*key*, *value*)

Parameters

- **key** (str)
- **value** (str)

Return type
dict[str, str] | None

14.1.7 audible.login module

audible.login.**build_client_id**(*serial*)

Parameters
serial (str)

Return type
str

audible.login.**build_device_serial**()

Return type
str

audible.login.**build_init_cookies**()

Build initial cookies to prevent captcha in most cases.

Return type
dict[str, str]

audible.login.**build_oauth_url**(*country_code*, *domain*, *market_place_id*, *code_verifier*, *serial=None*, *with_username=False*)

Builds the url to login to Amazon as an Audible device.

Parameters

- **country_code** (str)
- **domain** (str)
- **market_place_id** (str)
- **code_verifier** (bytes)
- **serial** (str | None)
- **with_username** (bool)

Return type
tuple[str, str]

audible.login.**check_for_approval_alert**(*soup*)

Checks a Amazon login page for an approval alert.

Parameters
soup (BeautifulSoup)

Return type
bool

audible.login.**check_for_captcha**(*soup*)

Checks a Amazon login page for a captcha form.

Parameters
soup (BeautifulSoup)

Return type
bool

audible.login.**check_for_choice_mfa**(*soup*)

Checks a Amazon login page for a MFA selection form.

Parameters
soup (BeautifulSoup)

Return type
bool

audible.login.**check_for_cvf**(*soup*)

Parameters
soup (BeautifulSoup)

Return type
bool

audible.login.**check_for_mfa**(*soup*)

Checks a Amazon login page for a multi-factor authentication form.

Parameters
soup (BeautifulSoup)

Return type
bool

audible.login.**create_code_verifier**(*length=32*)

Parameters
length (int)

Return type
bytes

audible.login.**create_s256_code_challenge**(*verifier*)

Parameters
verifier (bytes)

Return type
bytes

audible.login.**default_approval_alert_callback**()

Helper function for handling approval alerts.

Return type
None

audible.login.**default_captcha_callback**(*captcha_url*)

Helper function for handling captcha.

Parameters
captcha_url (str)

Return type
str

audible.login.**default_cvf_callback**()

Helper function for handling cvf verifys.

Amazon sends a verify code via Mail or SMS.

Return type
str

audible.login.**default_login_url_callback**(*url*)

Helper function for login with external browsers.

Parameters
url (str)

Return type
str

audible.login.**default_otp_callback**()

Helper function for handling 2-factor authentication.

Return type
str

audible.login.**external_login**(*country_code*, *domain*, *market_place_id*, *serial=None*, *with_username=False*, *login_url_callback=None*)

Gives the url to login with external browser and prompt for result.

Note: If you are using MacOS and have trouble insert the login result url simply import the readline module in your script. See #34.

Parameters

- **country_code** (str) – The country code for the Audible marketplace to login.
- **domain** (str) – The top level domain for the Audible marketplace to login.
- **market_place_id** (str) – The id for the Audible marketplace to login.
- **serial** (str | None) – The device serial. If None a custom one will be created.
- **with_username** (bool) – If True login with Audible username instead of Amazon account.
- **login_url_callback** (Callable[[str], str] | None) – A custom Callable for handling login with external browsers. If None *default_login_url_callback()* is used.

Return type
dict[str, Any]

Returns
An authorization_code, a code_verifier and the device serial from the authorized Client.

audible.login.**extract_captcha_url**(*soup*)

Returns the captcha url from a Amazon login page.

Parameters
soup (BeautifulSoup)

Return type
str

audible.login.**extract_code_from_url**(*url*)

Extracts the access token from url query after login.

Parameters
url (URL)

Return type
str

audible.login.**get_inputs_from_soup**(*soup*, *search_field=None*)

Extracts hidden form input fields from a Amazon login page.

Parameters

- **soup** (BeautifulSoup)
- **search_field** (dict[str, str] | None)

Return type
dict[str, str]

audible.login.**get_next_action_from_soup**(*soup*, *search_field=None*)

Parameters

- **soup** (BeautifulSoup)
- **search_field** (dict[str, str] | None)

Return type
tuple[str, str]

audible.login.**get_soup**(*resp*, *log_errors=True*)

Parameters

- **resp** (Response)
- **log_errors** (bool)

Return type
BeautifulSoup

audible.login.**is_valid_email**(*obj*)

Parameters
obj (str)

Return type
bool

audible.login.**login**(*username*, *password*, *country_code*, *domain*, *market_place_id*, *serial=None*, *with_username=False*, *captcha_callback=None*, *otp_callback=None*, *cvf_callback=None*, *approval_callback=None*)

Login to Audible by simulating an Audible App for iOS.

Parameters

- **username** (str) – The Amazon email address.
- **password** (str) – The Amazon password.

- **country_code** (str) – The country code for the Audible marketplace to login.
- **domain** (str) – domain: The top level domain for the Audible marketplace to login.
- **market_place_id** (str) – The id for the Audible marketplace to login.
- **serial** (str | None) – The device serial. If None a custom one will be created.
- **with_username** (bool) – If True login with Audible username instead of Amazon account.
- **captcha_callback** (Callable[[str], str] | None) – A custom Callable for handling captcha requests. If None *default_captcha_callback()* is used.
- **otp_callback** (Callable[[], str] | None) – A custom Callable for providing one-time passwords. If None *default_otp_callback()* is used.
- **cvf_callback** (Callable[[], str] | None) – A custom Callable for providing the answer for a CVF code. If None *default_cvf_callback()* is used.
- **approval_callback** (Callable[[], Any] | None) – A custom Callable for handling approval alerts. If None *default_approval_alert_callback()* is used.

Return type
dict[str, Any]

Returns
An authorization_code, a code_verifier and the device serial from the authorized Client.

Raises
Exception – If authorization_code is not in response url.

audible.login.**playwright_external_login_url_callback**(*url*)

Helper function for login using playwright.

Parameters
url (str)

Return type
str

14.1.8 audible.metadata module

class audible.metadata.**XXTEA**(*key*)

Bases: object

XXTEA wrapper class.

Easy to use and compatible (by duck typing) with the Blowfish class.

Note: Partial copied from https://github.com/andersekbom/prycut and ported from PY2 to PY3

Parameters
key (str | bytes)

decrypt(*data*)

Decrypts and returns a block of data.

Parameters
data (str | bytes)

Return type
bytes

encrypt(*data*)

Encrypts and returns a block of data.

Parameters
data (str | bytes)

Return type
bytes

exception audible.metadata.**XXTEAException**

Bases: Exception

audible.metadata.**decrypt_metadata**(*encrypted_metadata*)

Decrypts metadata for testing purposes only.

Parameters
encrypted_metadata (str)

Return type
str

audible.metadata.**encrypt_metadata**(*metadata*)

Encrypts metadata to be used to log in to Amazon.

Parameters
metadata (str)

Return type
str

audible.metadata.**meta_audible_app**(*user_agent*, *oauth_url*)

Returns json-formatted metadata to simulate sign-in from iOS audible app.

Parameters

- **user_agent** (str)
- **oauth_url** (str)

Return type
str

audible.metadata.**now_to_unix_ms**()

Return type
int

audible.metadata.**raw_xxtea**(*v*, *n*, *k*)

Parameters

- **v** (list[int])
- **n** (int)
- **k** (list[int] | tuple[int, ...])

Return type
int

14.1.9 audible.register module

audible.register.**deregister**(*access_token*, *domain*, *deregister_all=False*, *with_username=False*)

Deregister a previous registered Audible device.

Note: Except of the `access_token`, all authentication data will lose validation immediately.

Parameters

- **access_token** (`str`) – The access token from the previous registered device which you want to deregister.
- **domain** (`str`) – The top level domain of the requested Amazon server (e.g. com).
- **deregister_all** (`bool`) – If `True`, deregister all Audible devices on Amazon.
- **with_username** (`bool`) – If `True` uses *audible* domain instead of *amazon*.

Return type
`Any`

Returns
The response for the deregister request. Contains errors, if some occurs.

Raises
Exception – If response status code is not 200.

Added in version v0.8: The with_username argument

audible.register.**register**(*authorization_code*, *code_verifier*, *domain*, *serial*, *with_username=False*)

Registers a dummy Audible device.

Parameters

- **authorization_code** (`str`) – The code given after a successful authorization
- **code_verifier** (`bytes`) – The verifier code from authorization
- **domain** (`str`) – The top level domain of the requested Amazon server (e.g. com).
- **serial** (`str`) – The device serial
- **with_username** (`bool`) – If `True` uses *audible* domain instead of *amazon*.

Return type
`dict[str, Any]`

Returns
Additional authentication data needed for access Audible API.

Raises
Exception – If response status code is not 200.

Added in version v0.7.1: The with_username argument

14.1.10 audible.utils module

class audible.utils.**ElapsedTime**

Bases: object

audible.utils.**test_convert**(*key*, *value*)

Helper function to check and convert values for specific keys.

Parameters

- **key** (str)
- **value** (Any)

Return type
Any

CHAPTER

FIFTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

audible, 57
audible.activation_bytes, 64
audible.aescipher, 65
audible.auth, 70
audible.client, 76
audible.exceptions, 81
audible.localization, 83
audible.login, 84
audible.metadata, 88
audible.register, 90
audible.utils, 91

HTTP ROUTING TABLE

/0.0

GET /0.0/library/books, 31

/1.0

GET /1.0/account/information, 42
GET /1.0/annotations/lastpositions, 44
GET /1.0/app/upgradestatus, 45
GET /1.0/badges/metadata, 40
GET /1.0/badges/progress, 40
GET /1.0/catalog/categories, 34
GET /1.0/catalog/categories/(category_id), 35
GET /1.0/catalog/products, 36
GET /1.0/catalog/products/(string:asin), 35
GET /1.0/catalog/products/(string:asin)/reviews, 35
GET /1.0/catalog/products/(string:asin)/sims, 36
GET /1.0/collections, 37
GET /1.0/collections/(collection_id), 37
GET /1.0/collections/(collection_id)/items, 38
GET /1.0/content/(string:asin)/metadata, 41
GET /1.0/customer/freetrial/eligibility, 42
GET /1.0/customer/information, 42
GET /1.0/customer/status, 42
GET /1.0/library, 32
GET /1.0/library/(string:asin), 32
GET /1.0/library/collections, 33
GET /1.0/library/collections/(param1), 33
GET /1.0/library/collections/(param1)/products, 34
GET /1.0/orders, 39
GET /1.0/pages/(string:param1), 44
GET /1.0/recommendations, 44
GET /1.0/stats/aggregates, 42
GET /1.0/stats/status/finished, 43
GET /1.0/user/settings, 45
GET /1.0/wishlist, 39
GET 1.0/content/FairPlay/certificate, 42
POST /1.0/collections, 37
POST /1.0/collections/(collection_id)/items, 38
POST /1.0/content/(string:asin)/drmlicense, 41
POST /1.0/content/(string:asin)/licenserequest, 40
POST /1.0/library/collections, 33
POST /1.0/library/collections/(param1)/channels/(param2), 33
POST /1.0/library/collections/(param1)/products/(param2), 33
POST /1.0/library/item, 32
POST /1.0/library/item/(param1)/(param2), 33
POST /1.0/orders, 39
POST /1.0/stats/status/finished, 43
POST /1.0/wishlist, 39
PUT /1.0/collections/(collection_id), 38
PUT /1.0/lastpositions/(string:asin), 44
PUT /1.0/library/item, 33
PUT /1.0/stats/events, 43
DELETE /1.0/wishlist/(string:asin), 39

/https:

GET https://cde-ta-g7g.amazon.com/FionaCDEServiceEngine/si 45

INDEX

A

access_token (*audible.auth.Authenticator attribute*), 70
access_token (*audible.Authenticator attribute*), 58
access_token_expired (*audible.auth.Authenticator property*), 70
access_token_expired (*audible.Authenticator property*), 58
access_token_expires (*audible.auth.Authenticator property*), 70
access_token_expires (*audible.Authenticator property*), 58
activation_bytes (*audible.auth.Authenticator attribute*), 70
activation_bytes (*audible.Authenticator attribute*), 58
adp_token (*audible.auth.Authenticator attribute*), 70
adp_token (*audible.Authenticator attribute*), 58
aes_cbc_decrypt() (*in module audible.aescipher*), 68
aes_cbc_encrypt() (*in module audible.aescipher*), 68
AESCipher (*class in audible.aescipher*), 65
AsyncClient (*class in audible*), 57
AsyncClient (*class in audible.client*), 76
audible
    module, 57
audible.activation_bytes
    module, 64
audible.aescipher
    module, 65
audible.auth
    module, 70
audible.client
    module, 76
audible.exceptions
    module, 81
audible.localization
    module, 83
audible.login
    module, 84
audible.metadata
    module, 88
audible.register
    module, 90
audible.utils
    module, 91
auth (*audible.client.BaseClient property*), 78
auth_flow() (*audible.auth.Authenticator method*), 70
auth_flow() (*audible.Authenticator method*), 58
Authenticator (*class in audible*), 58
Authenticator (*class in audible.auth*), 70
AuthFlowError, 81
autodetect_locale() (*in module audible.localization*), 83
available_auth_modes (*audible.auth.Authenticator property*), 71
available_auth_modes (*audible.Authenticator property*), 59

B

BadRequest, 81
BaseClient (*class in audible.client*), 77
build_client_id() (*in module audible.login*), 84
build_device_serial() (*in module audible.login*), 84
build_init_cookies() (*in module audible.login*), 84
build_oauth_url() (*in module audible.login*), 84

C

check_for_approval_alert() (*in module audible.login*), 84
check_for_captcha() (*in module audible.login*), 84
check_for_choice_mfa() (*in module audible.login*), 85
check_for_cvf() (*in module audible.login*), 85
check_for_mfa() (*in module audible.login*), 85
Client (*class in audible*), 62
Client (*class in audible.client*), 79
close() (*audible.AsyncClient method*), 57
close() (*audible.Client method*), 63
close() (*audible.client.AsyncClient method*), 76
close() (*audible.client.Client method*), 80
convert_response_content() (*in module audible.client*), 81
country_code (*audible.localization.Locale property*), 83

create_code_verifier() (*in module audible.login*), 85
create_s256_code_challenge() (*in module audible.login*), 85
create_salt() (*in module audible.aescipher*), 68
crypter (*audible.auth.Authenticator attribute*), 71
crypter (*audible.Authenticator attribute*), 59
customer_info (*audible.auth.Authenticator attribute*), 71
customer_info (*audible.Authenticator attribute*), 59

D

decrypt() (*audible.metadata.XXTEA method*), 88
decrypt_metadata() (*in module audible.metadata*), 89
decrypt_voucher_from_licenserequest() (*in module audible.aescipher*), 68
default_approval_alert_callback() (*in module audible.login*), 85
default_captcha_callback() (*in module audible.login*), 85
default_cvf_callback() (*in module audible.login*), 86
default_login_url_callback() (*in module audible.login*), 86
default_otp_callback() (*in module audible.login*), 86
default_response_callback() (*in module audible.client*), 81
delete() (*audible.AsyncClient method*), 57
delete() (*audible.Client method*), 63
delete() (*audible.client.AsyncClient method*), 76
delete() (*audible.client.BaseClient method*), 78
delete() (*audible.client.Client method*), 80
deregister() (*in module audible.register*), 90
deregister_device() (*audible.auth.Authenticator method*), 71
deregister_device() (*audible.Authenticator method*), 59
derive_from_pbkdf2() (*in module audible.aescipher*), 69
detect_file_encryption() (*in module audible.aescipher*), 69
device_info (*audible.auth.Authenticator attribute*), 71
device_info (*audible.Authenticator attribute*), 59
device_private_key (*audible.auth.Authenticator attribute*), 71
device_private_key (*audible.Authenticator attribute*), 59
domain (*audible.localization.Locale property*), 83

E

ElapsedTime (*class in audible.utils*), 91
encrypt() (*audible.metadata.XXTEA method*), 89
encrypt_metadata() (*in module audible.metadata*), 89
encryption (*audible.auth.Authenticator attribute*), 71
encryption (*audible.Authenticator attribute*), 59
expires (*audible.auth.Authenticator attribute*), 71
expires (*audible.Authenticator attribute*), 59
external_login() (*in module audible.login*), 86
extract_activation_bytes() (*in module audible.activation_bytes*), 64
extract_captcha_url() (*in module audible.login*), 86
extract_code_from_url() (*in module audible.login*), 87

F

fetch_activation() (*in module audible.activation_bytes*), 64
fetch_activation_sign_auth() (*in module audible.activation_bytes*), 64
FileEncryptionError, 81
filename (*audible.auth.Authenticator attribute*), 71
filename (*audible.Authenticator attribute*), 59
from_bytes() (*audible.aescipher.AESCipher method*), 66
from_dict() (*audible.aescipher.AESCipher method*), 66
from_dict() (*audible.auth.Authenticator class method*), 71
from_dict() (*audible.Authenticator class method*), 59
from_file() (*audible.aescipher.AESCipher method*), 67
from_file() (*audible.auth.Authenticator class method*), 71
from_file() (*audible.Authenticator class method*), 59
from_login() (*audible.auth.Authenticator class method*), 72
from_login() (*audible.Authenticator class method*), 60
from_login_external() (*audible.auth.Authenticator class method*), 73
from_login_external() (*audible.Authenticator class method*), 61

G

get() (*audible.AsyncClient method*), 57
get() (*audible.Client method*), 63
get() (*audible.client.AsyncClient method*), 77
get() (*audible.client.BaseClient method*), 78
get() (*audible.client.Client method*), 80
get_activation_bytes() (*audible.auth.Authenticator method*), 73
get_activation_bytes() (*audible.Authenticator method*), 61
get_activation_bytes() (*in module audible.activation_bytes*), 64
get_inputs_from_soup() (*in module audible.login*), 87

get_next_action_from_soup() (*in module audible.login*), 87
get_player_id() (*in module audible.activation_bytes*), 65
get_player_token() (*in module audible.activation_bytes*), 65
get_soup() (*in module audible.login*), 87
get_user_profile() (*audible.client.BaseClient method*), 78

H

hashmod (*audible.aescipher.AESCipher attribute*), 66

I

is_valid_email() (*in module audible.login*), 87

K

kdf_iterations (*audible.aescipher.AESCipher attribute*), 66
key_size (*audible.aescipher.AESCipher attribute*), 66

L

locale (*audible.auth.Authenticator attribute*), 73
locale (*audible.Authenticator attribute*), 61
Locale (*class in audible.localization*), 83
login() (*in module audible.login*), 87

M

mac (*audible.aescipher.AESCipher attribute*), 66
market_place_id (*audible.localization.Locale property*), 83
marketplace (*audible.client.BaseClient property*), 78
meta_audible_app() (*in module audible.metadata*), 89
module
    audible, 57
    audible.activation_bytes, 64
    audible.aescipher, 65
    audible.auth, 70
    audible.client, 76
    audible.exceptions, 81
    audible.localization, 83
    audible.login, 84
    audible.metadata, 88
    audible.register, 90
    audible.utils, 91

N

NetworkError, 81
NoRefreshToken, 81
NotFoundError, 81
NotResponding, 82
now_to_unix_ms() (*in module audible.metadata*), 89

P

pack_salt() (*in module audible.aescipher*), 69
password (*audible.aescipher.AESCipher attribute*), 66
playwright_external_login_url_callback() (*in module audible.login*), 88
post() (*audible.AsyncClient method*), 58
post() (*audible.Client method*), 63
post() (*audible.client.AsyncClient method*), 77
post() (*audible.client.BaseClient method*), 78
post() (*audible.client.Client method*), 80
put() (*audible.AsyncClient method*), 58
put() (*audible.Client method*), 63
put() (*audible.client.AsyncClient method*), 77
put() (*audible.client.BaseClient method*), 78
put() (*audible.client.Client method*), 80

R

raise_for_status() (*in module audible.client*), 81
RatelimitError, 82
raw_request() (*audible.client.BaseClient method*), 79
raw_xxtea() (*in module audible.metadata*), 89
refresh_access_token() (*audible.auth.Authenticator method*), 73
refresh_access_token() (*audible.Authenticator method*), 61
refresh_access_token() (*in module audible.auth*), 74
refresh_token (*audible.auth.Authenticator attribute*), 73
refresh_token (*audible.Authenticator attribute*), 61
refresh_website_cookies() (*in module audible.auth*), 75
register() (*in module audible.register*), 90
remove_file_encryption() (*in module audible.aescipher*), 69
RequestError, 82
requires_request_body (*audible.auth.Authenticator attribute*), 74
requires_request_body (*audible.Authenticator attribute*), 62

S

salt_marker (*audible.aescipher.AESCipher attribute*), 66
search_template() (*in module audible.localization*), 83
ServerError, 82
set_website_cookies_for_country() (*audible.auth.Authenticator method*), 74
set_website_cookies_for_country() (*audible.Authenticator method*), 62
sign_request() (*audible.auth.Authenticator method*), 74
sign_request() (*audible.Authenticator method*), 62

sign_request() (*in module audible.auth*), 75
StatusError, 82
store_authentication_cookie (*audible.auth.Authenticator attribute*), 74
store_authentication_cookie (*audible.Authenticator attribute*), 62
switch_marketplace() (*audible.client.BaseClient method*), 79
switch_user() (*audible.client.BaseClient method*), 79

T

test_convert() (*in module audible.utils*), 91
to_bytes() (*audible.aescipher.AESCipher method*), 67
to_dict() (*audible.aescipher.AESCipher method*), 67
to_dict() (*audible.auth.Authenticator method*), 74
to_dict() (*audible.Authenticator method*), 62
to_dict() (*audible.localization.Locale method*), 83
to_file() (*audible.aescipher.AESCipher method*), 67
to_file() (*audible.auth.Authenticator method*), 74
to_file() (*audible.Authenticator method*), 62

U

Unauthorized, 82
UnexpectedError, 82
unpack_salt() (*in module audible.aescipher*), 70
user_name (*audible.client.BaseClient property*), 79
user_profile() (*audible.auth.Authenticator method*), 74
user_profile() (*audible.Authenticator method*), 62
user_profile() (*in module audible.auth*), 75
user_profile_audible() (*in module audible.auth*), 76

W

website_cookies (*audible.auth.Authenticator attribute*), 74
website_cookies (*audible.Authenticator attribute*), 62
with_username (*audible.auth.Authenticator attribute*), 74
with_username (*audible.Authenticator attribute*), 62

X

XXTEA (*class in audible.metadata*), 88
XXTEAException, 89