ArchiveBox

Release 0.9.74

Nick Sweeting

Oct 08, 2026

CONTENTS

1 Documentation **3**

1.1 Contents . . . 3
1.2 Overview . . . 3
1.3 Getting Started . . . 15
1.4 Guides . . . 58
1.5 Architecture . . . 91
1.6 API Reference . . . 92
1.7 Meta . . . 379

Python Module Index **401**

Index **403**

Welcome to ArchiveBox!

Just getting started?
Check out the Quickstart guide.

Need help with something?
Open an issue on Github or chat on Zulip.

Want to join the community?
See our Community Wiki page.

ARCHIVEBOX

"The open-source self-hosted internet archive."

Website | Github | Source | Bug Tracker

```
mkdir my-archive; cd my-archive/
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox>=0.9.0rc0,<0.10
↪'

archivebox init
archivebox install
archivebox add https://example.com
archivebox status
```

CHAPTER

ONE

DOCUMENTATION

1.1 Contents

1.2 Overview

1.2.1 ArchiveBox Documentation

Use the sidebar on the right to browse documentation topics

(Expand the `Pages` section to Search for a specific term)

If you need help or have a question, you can:

- Chat with us on our Public Zulip Server
- Open an issue or feature request in our bug tracker
- Reach out to me on Twitter @ArchiveBoxApp or @theSquashSH

Readme | Demo | Quickstart | Usage | Community

Donate to support open-source development

Quickstart | Demo | GitHub | Documentation | Info & Motivation | Community

ArchiveBox is a self-hosted app that lets you preserve content from websites in a variety of formats.

We aim to make your data immediately useful, and kept in formats that other programs can read directly. As output, we save standard HTML, PNG, PDF, TXT, JSON, WARC, SQLite, all guaranteed to be readable for decades to come. ArchiveBox also has a CLI, REST API, and webhooks so you can set up integrations with other services.

Without active preservation effort, everything on the internet eventually disappears or degrades.

ArchiveBox is an open source tool that lets organizations & individuals archive both public & private web content while retaining control over their data. It can be used to save copies of bookmarks, preserve evidence for legal cases, backup photos from FB/Insta/Flickr or media from YT/Soundcloud/etc., save research papers, and more...

> Get ArchiveBox with `pip install archivebox>=0.9.0rc0'` on *Linux*/*macOS*, or via ***Docker*** on Linux/macOS.

Once installed, you can interact with it through the: Browser Extension, CLI, self-hosted web interface, Python API, *or* filesystem.

You can feed ArchiveBox URLs one at a time, or schedule regular imports from your bookmarks or history, social media feeds or RSS, link-saving services like Pocket/Pinboard, our Browser Extension, and more.
See Input Formats for a full list of supported input formats…

It saves snapshots of the URLs you feed it in several redundant formats.
It also detects any content featured *inside* pages & extracts it out into a folder:

- **HTML/Any websites** `original HTML+CSS+JS`, `singlefile HTML`, `screenshot PNG`, `PDF`, `WARC`, `title`, `article text`, `favicon`, `headers`, …
- **Social Media/News** `post content TXT`, `comments`, `title`, `author`, `images`, …
- **YouTube/SoundCloud**/etc. `MP3/MP4s`, `subtitles`, `metadata`, `thumbnail`, …
- **Github/Gitlab**/etc. links `clone of GIT source code`, `README`, `images`, …
- *and more, see Output Formats below…*

You can run ArchiveBox as a Docker web app to manage these snapshots, or continue accessing the same collection using the uv-installed CLI, Python API, and SQLite3 APIs. All the ways of using it are equivalent, and provide matching features like adding tags, scheduling regular crawls, viewing logs, and more…

ArchiveBox uses *standard tools* like Chrome, `wget`, & `yt-dlp`, and stores data in *ordinary files & folders*.
(no complex proprietary formats, all data is readable without needing to run ArchiveBox)

The goal is to sleep soundly knowing the part of the internet you care about will be automatically preserved in durable, easily accessible formats *for decades* after it goes down.

Install ArchiveBox using your preferred method: `docker` / `uv` / `brew` / `apt` / etc. (*see full Quickstart below*).

1.2.2 Key Features

- **Free & open source**, own your own data & maintain your privacy by self-hosting
- ***Powerful CLI*** with *modular dependencies* and *support for Google Drive/NFS/SMB/S3/B2/etc.*
- **Comprehensive documentation**, *active development*, and *rich community*
- **Extracts a wide variety of content out-of-the-box**: media (yt-dlp), articles (readability), code (git), etc.
- ***Supports scheduled/realtime importing*** from many types of sources
- **Uses standard, durable, long-term formats** like HTML, JSON, PDF, PNG, MP4, TXT, and WARC
- ***Powerful CLI***, ***self-hosted web UI***, Python API (BETA), REST API (ALPHA), or desktop app
- **Saves all pages to archive.org as well** by default for redundancy (can be disabled for local-only mode)
- Support for archiving content requiring login/paywall/cookies (see wiki security caveats!)
- Support for running JS during archiving to adblock, infiniscroll, modal-hide, thread-expand

1.2.3 Quickstart

***Supported OSs*:** Ubuntu, macOS, Docker **CPUs:** `amd64` (`x86_64`), `arm64`

Easy Setup

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive.

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive.

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive. See setup.sh for the source code of the uv install shortcut. See "Against curl | sh as an install method" blog post for my thoughts on the shortcomings of this install method.

Package Manager Setup

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive.

See the uv tool documentation for more details about this installation method.

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive. See the homebrew-archivebox repo for more details about this distribution.

> *Warning: These are contributed by external volunteers and may lag behind the official* `uv` *and Docker channels.*

Other Options

> *Warning: These are contributed by external volunteers and may lag behind the official Docker and* `uv` *channels.*

For more discussion on third-party hosting options see here: Issue #531.

Next Steps

- Import URLs from some of the supported Input Formats or view the supported Output Formats…
- (Optional) Import browser cookies and settings into a persona.
- Tweak your UI or archiving behavior *Configuration*, read about some of the *Caveats*, or *Troubleshoot*
- Read about the *Dependencies* used for archiving, the *Upgrading Process*, or the *Archive Layout* on disk…
- Or check out our full *Documentation* or *Community Wiki*…

Usage

CLI Usage

ArchiveBox commands can be run in a terminal *directly on your host*, or via *Docker*/*Docker Compose*.
(depending on how you chose to install it above)

```
mkdir -p ~/archivebox/data   # create a new data dir anywhere
cd ~/archivebox/data         # IMPORTANT: cd into the directory

# archivebox [subcommand] [--help]
archivebox version
archivebox help

# equivalent: docker compose run --rm archivebox [subcommand] [--help]
docker compose run --rm archivebox help

# equivalent: docker run -it -v $PWD:/data archivebox/archivebox:dev [subcommand] [--
↪help]
docker run -it -v $PWD:/data archivebox/archivebox:dev help
```

ArchiveBox Subcommands

- `archivebox help/version` to see the list of available subcommands / currently installed version info
- `archivebox setup/init/config/status/shell/manage` to administer your collection
- `archivebox add/schedule` to pull in fresh URLs from bookmarks/history/RSS/etc.
- `archivebox list/update/remove` to manage existing Snapshots in your collection

Open http://admin.archivebox.localhost:5797/admin/ to create the first admin and finish web setup. Use http://web.archivebox.localhost:5797 for the public UI. Advanced: archivebox manage createsuperuser remains available for creating accounts from the CLI. Set BASE_URL to change the public base domain. The default auto mode uses web. and admin. subdomains on *.localhost, but one host for ordinary DNS names. BIND_ADDR only controls the local listen address.

For more info, see our Usage: Web UI wiki.

Optional: Change permissions to allow non-logged-in users

Snapshot caching follows each snapshot's saved `PERMISSIONS`, including inherited defaults and snapshot overrides. Public content can be cached by browsers and shared proxies/CDNs; private and unlisted content uses `Cache-Control: private` so only the user's browser can cache it. Both retain normal artifact cache lifetimes and validators for fast repeat viewing. Authentication handoffs are not stored. Changing permissions does not revoke copies already cached under the previous policy; purge any affected shared cache when needed.

- ArchiveBox iOS Mobile App
- ArchiveBox.app macOS Desktop App
- ArchiveBox Browser Extension
- Linux/Windows Desktop App (Requires Docker Desktop)

  [!TIP] Whether in Docker or not, ArchiveBox commands work the same way, and can be used to access the same data on-disk. For example, you could run the Web UI in Docker Compose, and run one-off commands with `uv`-installed ArchiveBox.

For more info, see our Docker wiki.

1.2.4 Overview

Input Formats: How to pass URLs into ArchiveBox for saving

- From the official ArchiveBox Browser Extension
  Provides realtime archiving of browsing history or selected pages from Chrome/Chromium/Firefox/Safari browsers.
- From the official ArchiveBox macOS Desktop App or iOS Mobile App Save directly from the Share menu or Siri/Shortcuts to your server.
- From manual imports of URLs from RSS, JSON, CSV, TXT, SQL, HTML, Markdown, etc. files
  ArchiveBox supports injesting URLs in *any text-based format*.
- From manually exported *browser history* or *browser bookmarks* (in Netscape format)
  Instructions: Chrome, Firefox, Safari, IE, Opera, and more...
- From URLs visited through a MITM Proxy with `archivebox-proxy`
  Provides realtime archiving of all traffic from any device going through the proxy.

- From bookmarking services or social media (e.g. Twitter bookmarks, Reddit saved posts, etc.)
  Instructions: Pocket, Pinboard, Instapaper, Shaarli, Delicious, Reddit Saved, Wallabag, Unmark.it, OneTab, Firefox Sync, and more...

```
# archivebox add --help
archivebox add 'https://example.com/some/page'
archivebox add --depth=1 --plugins=parse_rss_urls "file://$HOME/Downloads/some_feed.xml"
archivebox add --depth=1 'https://news.ycombinator.com#2020-12-12'
echo 'http://example.com' | archivebox add
echo 'any text with <a href="https://example.com">urls</a> in it' | archivebox add

# if using Docker, add -i when piping stdin:
# echo 'https://example.com' | docker run -v $PWD:/data -i archivebox/archivebox:dev add
# if using Docker Compose, add -T when piping stdin / stdout:
# echo 'https://example.com' | docker compose run --rm -T archivebox add
```

See the *Usage: CLI* page for documentation and examples.

It also includes a built-in scheduled import feature with `archivebox schedule`, handled by the same orchestrator that powers `archivebox server`, so you can pull in URLs from RSS feeds and websites regularly without a separate cron container.

Output Formats: What ArchiveBox saves for each URL

For each web page added, ArchiveBox creates a Snapshot folder and preserves its content as ordinary files inside the folder (e.g. HTML, PDF, PNG, JSON, etc.).

It uses all available methods out-of-the-box, but you can disable extractors and fine-tune the *configuration* as-needed.

data/archive/users/{username}/snapshots/{YYYYMMDD}/{domain}/{Snapshot.id}/

Configuration

ArchiveBox can be configured via environment variables, by using the `archivebox config` CLI, or by editing `./ArchiveBox.conf`.

The configuration is documented here: ***Configuration Wiki***, and loaded from: `archivebox/config/`.

Dependencies

To achieve high-fidelity archives in as many situations as possible, ArchiveBox depends on a variety of 3rd-party libraries and tools that specialize in extracting different types of content.

> Under-the-hood, ArchiveBox uses Django to power its *Web UI*, Django Ninja for the REST API, and SQlite + the filesystem to provide fast & durable metadata storage w/ deterministic upgrades.

ArchiveBox bundles industry-standard tools like *Google Chrome*, *`wget`*, *`yt-dlp`*, *`readability`*, *etc.* internally, and its operation can be *tuned, secured, and extended* as-needed for many different applications.

These optional subdependencies used for archiving sites include:

You don't need to install every dependency by hand. ArchiveBox resolves every extractor dependency through abxpkg: it uses a compatible host installation when one is already available, and otherwise installs and manages the dependency for you.

Installing directly on Windows without Docker or WSL/WSL2/Cygwin is not officially supported (I cannot respond to Windows support tickets), but some advanced users have reported getting it working.

Archive Layout

All of ArchiveBox's state (SQLite DB, content, config, logs, etc.) is stored in a single folder per collection.

Data folders can be created anywhere (`~/archivebox/data` or `$PWD/data` as seen in our examples), and you can create as many data folders as you want to hold different collections. All archivebox CLI commands are designed to be run from inside an ArchiveBox data folder, starting with archivebox init to initialize a new collection inside an empty directory.

The on-disk layout is optimized to be easy to browse by hand and durable long-term. The main index is a standard index.sqlite3 database in the root of the data folder (it can also be exported as static JSON/HTML). Snapshot data is organized by user, date, domain, and UUID under data/archive/users/.

Each snapshot subfolder includes static metadata and plain extractor output files. Current releases do not create top-level data/archive/TIMESTAMP projections; legacy timestamp directories are migrated into the user-scoped tree by archivebox update –migrate-only.

Static Archive Exporting

You can export your index as static HTML using `archivebox list` (so you can view it without an ArchiveBox server).

The paths in the static exports are relative, make sure to keep them next to your `./archive` folder when backing them up or viewing them.

Caveats

Archiving Private Content

If you're importing pages with private content or URLs containing secret tokens you don't want public (e.g Google Docs, paywalled content, unlisted videos, etc.), **you may want to disable some of the extractor methods to avoid leaking that content to 3rd party APIs or the public**.

Security Risks of Viewing Archived JS

Archived JavaScript is untrusted content. The default SERVER_SECURITY_MODE=auto uses isolated subdomains with full replay on *.localhost, and a one-domain no-JS replay policy on ordinary public or LAN hostnames. Choose safe-subdomains-fullreplay only when wildcard DNS and TLS are configured. See the *Security Overview* and Issue #239 for details.

Working Around Sites that Block Archiving

For various reasons, many large sites (Reddit, Twitter, Cloudflare, etc.) actively block archiving or bots in general. There are a number of approaches to work around this.

Saving Multiple Snapshots of a Single URL

ArchiveBox can preserve multiple snapshots of the same URL. The default ONLY_NEW=True skips URLs already in the collection; use –no-only-new when you intentionally want another snapshot.

Each re-archive creates a distinct Snapshot row for the same URL:

The button in the Admin UI performs the same explicit re-archive.

Storage Requirements

Because ArchiveBox is designed to ingest a large volume of URLs with multiple copies of each URL stored by different 3rd-party tools, it can be quite disk-space intensive. There are also some special requirements when using filesystems like NFS/SMB/FUSE.

Screenshots

1.2.5 Background & Motivation

ArchiveBox aims to enable more of the internet to be saved from deterioration by empowering people to self-host their own archives. The intent is for all the web content you care about to be viewable with common software in 50 - 100 years without needing to run ArchiveBox or other specialized software to replay it.

Vast treasure troves of knowledge are lost every day on the internet to link rot. As a society, we have an imperative to preserve some important parts of that treasure, just like we preserve our books, paintings, and music in physical libraries long after the originals go out of print or fade into obscurity.

Whether it's to resist censorship by saving news articles before they get taken down or edited, or just to save a collection of early 2010's flash games you loved to play, having the tools to archive internet content enables to you save the stuff you care most about before it disappears.

The balance between the permanence and ephemeral nature of content on the internet is part of what makes it beautiful. I don't think everything should be preserved in an automated fashion–making all content permanent and never removable, but I do think people should be able to decide for themselves and effectively archive specific content that they care about, just like libraries do. Without the work of archivists saving physical books, manuscrips, and paintings we wouldn't have any knowledge of our ancestors' history. I believe archiving the web is just as important to provide the same benefit to future generations.

ArchiveBox's stance is that duplication of other people's content is only ethical if it:

- A. doesn't deprive the original creators of revenue and
- B. is responsibly curated by an individual/institution.

In the U.S., libraries, researchers, and archivists are allowed to duplicate copyrighted materials under "fair use" for private study, scholarship, or research. Archive.org's non-profit preservation work is covered under fair use in the US, and they properly handle unethical content/DMCA/GDPR removal requests to maintain good standing in the eyes of the law.

As long as you A. don't try to profit off pirating copyrighted content and B. have processes in place to respond to removal requests, many countries allow you to use software like ArchiveBox to ethically and responsibly archive any web content you can view. That being said, ArchiveBox is not liable for how you choose to operate the software. You must research your own local laws and regulations, and get proper legal counsel if you plan to host a public instance (start by putting your DMCA/GDPR contact info in FOOTER_INFO and placing branding overrides in your collection's fixed custom_templates/ directory).

Comparison to Other Projects

Check out our *community wiki* for a list of alternative web archiving tools and orgs.

ArchiveBox gained momentum in the internet archiving industry because it uniquely combines 3 things:

- **it's distributed:** users own their data instead of entrusting it to one big central provider
- **it's future-proof:** saving in *multiple formats* and extracting out raw TXT, PNG, PDF, MP4, etc. files
- **it's extensible:** with powerful APIs, flexible storage, and a big community adding new extractors regularly

ArchiveBox tries to be a robust, set-and-forget archiving solution suitable for archiving RSS feeds, bookmarks, or your entire browsing history (beware, it may be too big to store), including private/authenticated content that you wouldn't otherwise share with a centralized service like Archive.org.

Not all content is suitable to be archived on a centralized, publicly accessible platform. Archive.org doesn't offer the ability to save things behind login walls for good reason, as the content may not have been intended for a public audience. ArchiveBox exists to fill that gap by letting everyone save what they have access to on an individual basis, and to encourage decentralized archiving that's less succeptible to censorship or natural disasters.

By having users store their content locally or within their organizations, we can also save much larger portions of the internet than a centralized service has the disk capacity to handle. The eventual goal is to work towards federated archiving where users can share portions of their collections with each other, and with central archives on a case-by-case basis.

ArchiveBox differentiates itself from *similar self-hosted projects* by providing both a comprehensive CLI interface for managing your archive, a Web UI that can be used either independently or together with the CLI, and a simple on-disk data format that can be used without either.

If you want better fidelity for very complex interactive pages with heavy JS/streams/API requests, check out ArchiveWeb.page and ReplayWeb.page.

If you want more bookmark categorization and note-taking features, check out Memex, Hoarder, LinkWarden, Archivy, or LinkAce.

If you need more advanced recursive spider/crawling ability beyond `--depth=1`, check out Browsertrix, Photon, or Scrapy and pipe the outputted URLs into ArchiveBox.

For more alternatives, see our *list here*...

ArchiveBox is neither the highest fidelity nor the simplest tool available for self-hosted archiving, rather it's a jack-of-all-trades that tries to do most things well by default. We encourage you to try these other tools made by our friends if ArchiveBox isn't suited to your needs.

Internet Archiving Ecosystem

- *Community Wiki*
  - *Web Archiving Software*
    List of ArchiveBox alternatives and open source projects in the internet archiving space.
  - *Awesome-Web-Archiving Lists*
    Community-maintained indexes of archiving tools and institutions like `iipc/awesome-web-archiving`.
  - *Reading List*
    Articles, posts, and blogs relevant to ArchiveBox and web archiving in general.
  - *Communities*
    A collection of the most active internet archiving communities and initiatives.
- Check out the ArchiveBox *Roadmap* and *Changelog*
- Learn why archiving the internet is important by reading the "On the Importance of Web Archiving" blog post.
- Reach out to me for questions and comments via @ArchiveBoxApp or @theSquashSH on Twitter

1.2.6 Documentation

We use the ArchiveBox GitHub Wiki for documentation.

There is also a mirror available on Read the Docs (though it's sometimes outdated).

> You can submit docs changes & suggestions in our dedicated repo `ArchiveBox/docs`.

Getting Started

- *Quickstart*
- *Install*
- *Docker*
- *Usage*
- *Configuration*
- *Supported Sources*
- Supported Outputs
- *Scheduled Archiving*

Advanced

- *Security Overview*
- *Cookies & Sessions Setup* (archiving sites that require logins)
- *Setting up the Search Backends* (choosing ripgrep, Sonic, or FTS5)
- *Setting up Local/Remote Storages* (S3/B2/Google Drive/SMB/NFS/etc.)
- *Setting up Authentication & Permissions* (SSO/LDAP/OAuth/API Keys/etc.)
- *Publishing Your Archive* (sharing your archive server with others)
- *Chromium Install Options* (installing and configuring ArchiveBox's Chrome)
- *Upgrading or Merging Archives*
- *Troubleshooting*

Developers

- Developer Documentation
- Python API
- REST API (alpha)

More Info

- Bug Tracker
- *Roadmap*
- Changelog
- *Donations*
- Background & Motivation
- *Web Archiving Community*

1.2.7 ArchiveBox Development

All contributions to ArchiveBox are welcomed! Check our issues and *Roadmap* for things to work on, and please open an issue to discuss your proposed implementation before working on things! Otherwise we may have to close your PR if it doesn't align with our roadmap.

For low hanging fruit / easy first tickets, see: ArchiveBox/Issues `#good first ticket` `#help wanted`.

Python API Documentation: https://docs.archivebox.io/dev/apidocs/

Internal Architecture Diagrams: https://github.com/ArchiveBox/ArchiveBox/wiki/ArchiveBox-Architecture-Diagrams

Setup the dev environment

1. Setup the monorepo

First make sure you have `uv` installed: https://docs.astral.sh/uv/getting-started/installation/

```
git clone https://github.com/ArchiveBox/monorepo
cd monorepo
./bin/setup.sh

# activate the monorepo venv, then cd into archivebox
source .venv/bin/activate
cd archivebox
```

Repos included in monorepo setup:

- https://github.com/ArchiveBox/abxbus
- https://github.com/ArchiveBox/abxpkg
- https://github.com/ArchiveBox/abx-plugins
- https://github.com/ArchiveBox/abx-dl
- https://github.com/ArchiveBox/ArchiveBox

2. Option A: Install the Python, JS, and system dependencies directly on your machine

```
# Install ArchiveBox runtime dependencies
mkdir -p data && cd data
archivebox init
archivebox install         # detect and install all extractor dependencies

# Run the development server w/ autoreloading (but no bg workers)
archivebox server --debug --reload 0.0.0.0:5797

# Run the production server (with bg workers but no autoreloading)
archivebox server 0.0.0.0:5797
```

2. Option B: Build the docker container and use that for development instead

```
# Optional: develop via docker by mounting the code dir into the container
# if you edit e.g. ./archivebox/core/models.py on the docker host, runserver
# inside the container will reload and pick up your changes
```

(continues on next page)

(continued from previous page)

```
./bin/build_docker.sh dev

# Run the development server w/ autoreloading (but no bg workers)
docker run -it -v $PWD/data:/data -v $PWD/archivebox:/app/archivebox -p 5797:5797␣
↪archivebox/archivebox:dev server --init --debug --reload 0.0.0.0:5797

# Run the production server (with bg workers but no autoreloading)
docker run -it -v $PWD/data:/data -v $PWD/archivebox:/app/archivebox -p 5797:5797␣
↪archivebox/archivebox:dev server --init

# (remove the --reload flag and add the --nothreading flag when profiling with the␣
↪django debug toolbar)
# When using --reload, make sure any files you create can be read by the user in the␣
↪Docker container, eg with 'chmod a+rX'.
```

Common development tasks

See the `./bin/` folder and read the source of the bash scripts within. You can also run all these in Docker. For more examples see the GitHub Actions CI/CD tests that are run: `.github/workflows/*.yaml`.

Run in DEBUG mode

```
# set up persistent DEBUG=True for all runs
archivebox config --set DEBUG=True

# OR you can run a dev server with DEBUG=True in a few ways:
archivebox server --debug --reload 0.0.0.0:5797
# or
archivebox server --debug 0.0.0.0:5797
# or
env DEBUG=True daphne -b 0.0.0.0 -p 5797 archivebox.core.asgi:application
```

https://stackoverflow.com/questions/1074212/how-can-i-see-the-raw-sql-queries-django-is-running

Install and run a specific GitHub branch

Use a Pre-Built Image

If you're looking for the latest `dev` Docker image, it's often available pre-built on Docker Hub, simply pull and use `archivebox/archivebox:dev`.

```
docker pull archivebox/archivebox:dev
docker run archivebox/archivebox:dev version
# verify the BUILD_TIME and COMMIT_HASH in the output are recent
```

Build Branch from Source

You can also build and run any branch yourself from source, for example to build & use `dev` locally:

```
# docker-compose.yml:
services:
    archivebox:
```

(continues on next page)

(continued from previous page)

```
        image: archivebox/archivebox:dev
        build: 'https://github.com/ArchiveBox/ArchiveBox.git#dev'
        ...

# or with plain Docker:
docker build -t archivebox:dev https://github.com/ArchiveBox/ArchiveBox.git#dev
docker run -it -v $PWD:/data -p 5797:5797 archivebox:dev

# or with uv:
uv tool install --python 3.13 --upgrade 'git+https://github.com/ArchiveBox/ArchiveBox.
↪git@dev'
archivebox install
```

Run the linters / tests

```
./bin/lint.sh
./bin/test.sh
```

(uses `ruff`, `pyright`, `ty`, and `pytest -s`)

Make DB migrations, enter Django shell, other dev helper commands

```
# generate the database migrations after changes to models.py
archivebox manage makemigrations

# enter a python shell or a SQL shell
cd path/to/test/data/
archivebox shell
archivebox manage dbshell

# generate a graph of the ORM models
brew install graphviz
uv run --with pydot --with graphviz archivebox manage graph_models -a -o orm.png
open orm.png

# list all models with field db info and methods
archivebox manage list_model_info --all --signature --db-type --field-class

# print all django settings
archivebox manage print_settings
uv run --with pyyaml archivebox manage print_settings --format=yaml

# autogenerate an admin.py from given app models
archivebox manage admin_generator core > core/admin.py

# dump db data to a script that re-populates it
archivebox manage dumpscript core > scripts/testdata.py
archivebox manage reset core
archivebox manage runscript testdata

# resetdb and clear all data!
```

(continues on next page)

(continued from previous page)

```
archivebox manage reset_db

# use django-tui to interactively explore commands
# ensure django-tui is in INSTALLED_APPS: core/settings.py
uv run --with django-tui archivebox manage tui
```

- https://django-extensions.readthedocs.io/en/latest/command_extensions.html
- https://stackoverflow.com/questions/1074212/how-can-i-see-the-raw-sql-queries-django-is-running
- https://github.com/anze3db/django-tui (explore `manage.py` commands as TUI)
- https://github.com/bloomberg/memray (advanced python profiler)
- https://github.com/laixintao/flameshow (display flamegraphs in terminal)
- https://github.com/taliraj/django-migrations-tui (explore migrations as TUI)

Contributing a new extractor

Extractors are maintained in a separate repo here: https://github.com/ArchiveBox/abx-plugins (included in monorepo setup).

Copy a similar plugin as a template to modify, then open a new PR to add it in that repo.

Build the docs, pip package, and docker image

(Normally CI takes care of this, but these scripts can be run to do it manually)

```
./bin/build.sh

# or individually:
./bin/build_docs.sh
./bin/build_pip.sh
./bin/build_docker.sh
```

Further Reading

- ArchiveBox.io Website / ArchiveBox Github (Source Code) / ArchiveBox Demo Server
- Documentation (Github Wiki) / API Reference Docs (ReadTheDocs) / *Roadmap* / Changelog
- Bug Tracker (Github Issues) / Discussions (Github Discussions) / Community Chat Forum (Zulip)
- Find us on social media: Twitter `@ArchiveBoxApp`, LinkedIn, YouTube, SaaSHub, Alternative.to, Reddit

---

1.3 Getting Started

1.3.1 Quickstart

It only takes about 5 minutes to get up and running with ArchiveBox.

ArchiveBox *officially supports* **macOS** and **Ubuntu** on `amd64` or `arm64`, plus **Docker** on Linux and macOS. Other operating systems are not tested for this release.

For more detailed Docker and Docker Compose-specific instructions, see the *Docker* page.

---

1. Set up ArchiveBox

Follow the README Instructions for your platform to get archivebox set up.

2. Get your list of URLs to archive

Follow the links here to find instructions for exporting a list of URLs from each service.

- Pocket
- Pinboard
- Instapaper
- Reddit Saved Posts
- Shaarli
- Unmark.it
- Wallabag
- Chrome Bookmarks
- Firefox Bookmarks
- Safari Bookmarks
- Opera Bookmarks
- Internet Explorer Bookmarks
- Chrome History: `bash ./bin/export_browser_history.sh --chrome`
- Firefox History: `bash ./bin/export_browser_history.sh --firefox`
- Safari History: `bash ./bin/export_browser_history.sh --safari`
- Other File or URL: (e.g. RSS feed url, text file path) pass as second argument in the next step

(If any of these links are broken, please submit an issue and I'll fix it)

3. Add your URLs to the archive

Pass in URLs directly, import a list of links from a file, or import from a feed URL. All via stdin:

```
archivebox add < your_urls.txt

# or if using plain Docker
docker run -v $PWD:/data -i archivebox/archivebox:dev add < your_urls.txt

# or if using Docker Compose
docker compose run --rm -T archivebox add < your_urls.txt

# any text containing URLs can ingested via stdin or as args
curl -fsSL 'https://getpocket.com/users/YOURUSERNAME/feed/all' | archivebox add
archivebox add 'https://example.com'
```

Done!

Open `./archive` to view your archive data in the filesystem.

You can also use the interactive Web UI to view/manage/add links to your archive:

```
# with plain Docker:
docker run -v $PWD:/data -it -p 5797:5797 archivebox/archivebox:dev

# with Docker Compose:
docker compose up -d

# or without Docker:
archivebox server

# Open /admin/ on the hostname or IP used to reach ArchiveBox.
# Local example: http://admin.archivebox.localhost:5797/admin/
# If BASE_URL is not configured yet, continue through the web setup wizard.
```

Next Steps:

```
archivebox help   # see info about all the available commands
```

- Read *Usage* to learn about the various CLI and web UI functions
- Read *Configuration* to learn about the various archive method options
- Read *Scheduled Archiving* to learn how to set up automatic daily archiving
- Read *Publishing Your Archive* if you want to host your archive for others to access online
- Read *Troubleshooting* if you encounter any problems

1.3.2 Install

ArchiveBox is primarily distributed as a Python package installed with `uv`, but it also depends on some system packages that can be installed manually or automatically with Docker. It usually takes less than ~10min to get ArchiveBox set up and running.

- Supported Systems
- Install Instructions
  - ***Option A. Docker / Docker Compose***
  - *Option B. uv Install Shortcut*
  - *Option C. System Package Manager Setup*
    - Upgrading ArchiveBox to a new version
- Next Steps

Supported Systems

CPU Architectures: `amd64` (`x86_64`), `arm64` (`aarch64`) *(Including 64-bit Intel/AMD and Apple Silicon Macs)*

Memory: 1GB RAM minimum; 2GB+ is recommended. On a 1GB VPS, configure at least 4GB of swap before running full default crawls.

- ***macOS:*** >=13 on Intel or Apple Silicon (with `uv` or Homebrew)

- *Ubuntu:* on amd64 or arm64 (with apt, uv, or Homebrew)
- **Docker:** on amd64 or arm64 Linux/macOS

Other operating systems are not tested or supported for this release.

You can reduce crawl-time memory pressure by disabling all Chrome-based archiving methods with CHROME_ENABLED=False (or its USE_CHROME alias).

It's also recommended to use a filesystem with compression and/or deduplication (e.g. ZFS or BTRFS) for maximum efficiency.

Option A. Docker / Docker Compose Setup

Docker Compose is the recommended way to get ArchiveBox, as it includes all the extras out-of-the-box and provides the best security and upgrade UX.

1. If you don't already have Docker installed, follow the official instructions for Linux or macOS: https://docs.docker.com/install/#supported-platforms
2. Then follow the Quickstart guide and read the *Docker* wiki page for next steps.

   You can also run Dockerized ArchiveBox using UNRAID/TrueNAS/Proxmox/etc. or Kubernetes.

More info:

- Dockerfile
- docker-compose.yml
- ArchiveBox Docker Quickstart + *Usage* + *Configuration* + *Upgrading* documentation

---

Option B. uv Install Shortcut

On macOS, Linux, or BSD, get.archivebox.io is a shortcut for the uv install method.

(or scroll further down for manual install instructions)

```
curl -fsSL 'https://get.archivebox.io' | bash
# shortcut to run https://raw.githubusercontent.com/ArchiveBox/ArchiveBox/dev/bin/setup.
↪sh
```

The script installs uv when needed, then runs the same uv tool install command documented below. It does not initialize a collection, install runtime dependencies, or start a server; continue with the Quickstart after it finishes.

After running the setup script, continue with the Quickstart guide…

> *See here for our thoughts on the inherent limitations of curl | sh as an install method…*

Option C. Bare Metal Setup

If you'd rather not use Docker or our uv install shortcut, you can follow these manual setup instructions to install ArchiveBox and its dependencies using uv, apt, or Homebrew.

See our Dependencies documentation to see the full list of dependencies and how they're used. Not all the dependencies are required for all modes. If you disable some archive methods you can skip installing those dependencies — for

example, if you set `MEDIA_ENABLED=False` you don't need to install `yt-dlp`, and if you set `PDF_ENABLED=False`, `SCREENSHOT_ENABLED=False`, and `DOM_ENABLED=False` you don't need `chromium`.

More info:

- For help installing these, see the *Manual Setup*, *Troubleshooting* and *Chromium Install* pages.
- For per-plugin binary and enable/disable options (CHROME_BINARY, RIPGREP_BINARY, `<plugin>_ENABLED`, etc.) see the abx-plugins config reference.

1. Install uv or the ArchiveBox OS package

ArchiveBox itself is the only tool you need to bootstrap manually. After that, `archivebox install` resolves every runtime dependency through `abxpkg`, preferring compatible host binaries and installing managed ones only when needed.

macOS

Make sure you have Homebrew installed first.

```
# install ArchiveBox directly with Homebrew
brew tap archivebox/archivebox
brew trust archivebox/archivebox
brew install archivebox
```

Or use Homebrew to install `uv`, then install ArchiveBox with `uv`:

```
brew install uv
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox>=0.9.0rc0,<0.10
↪'
```

Ubuntu/Debian-based Systems

Use the third-party ArchiveBox apt repo for the simplest bare-metal install:

```
echo 'deb [trusted=yes] https://archivebox.github.io/debian-archivebox dev main' | sudo␣
↪tee /etc/apt/sources.list.d/archivebox.list
sudo apt update
sudo apt install archivebox

mkdir -p ~/archivebox/data
cd ~/archivebox/data
archivebox init
sudo archivebox install
archivebox add 'https://example.com'
```

The apt package is a thin dev-channel wrapper around the normal Python install flow. Runtime extractor dependencies such as Chromium, yt-dlp, SingleFile, and other plugin-managed tools are installed by `sudo archivebox install`, which uses apt for missing system dependencies while preserving ownership of the user-owned collection.

Alternatively, install with `uv`:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
source "$HOME/.local/bin/env"
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox>=0.9.0rc0,<0.10
↪'
```

Or install with Homebrew on Linux. Run Homebrew as your normal non-root user; Homebrew does not support `sudo brew` or root installs.

```
brew tap archivebox/archivebox
brew trust archivebox/archivebox
brew install archivebox
```

2. Install ArchiveBox using `uv`

If you are not using the apt or Homebrew packages above, install ArchiveBox with `uv`.

```
# get the dev version of ArchiveBox
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox>=0.9.0rc0,<0.10
↪'

# if the optional ldap extra must compile locally on Debian/Ubuntu, install its headers␣
↪and retry
# sudo apt install build-essential libldap2-dev libsasl2-dev
```

3. Install runtime dependencies using `archivebox install`

Finish installing runtime dependencies for the enabled ArchiveBox plugins.

```
# create a new empty folder anywhere to hold your collection, and cd into it
mkdir -p ~/archivebox/data && cd ~/archivebox/data

# instantiate the directory as an archivebox collection dir
archivebox init

# auto-install runtime dependencies such as Chromium, yt-dlp, SingleFile, etc.
archivebox install

# archive a first URL
archivebox add 'https://example.com'

#  see a final detailed breakdown of all the installed dependencies and commands␣
↪available
archivebox version
archivebox help
```

Troubleshooting

For `uv` installs, make sure the `uv`-installed version of `archivebox` is available in your `$PATH`.

```
uv tool list             # show info about uv-installed tools

echo $PATH               # show the directories your system is searching for binaries
type -a archivebox       # show all installed archivebox binaries available

cd ~/archivebox/data
archivebox version       #  show lots of useful info about installed dependencies and␣
↪more
```

(continues on next page)

(continued from previous page)

```
archivebox status
archivebox help
```

(ensure the version shown is the most recent available from Releases)

ArchiveBox can be launched as `root` by setup and package-manager flows; it creates or selects the `archivebox` service account and drops privileges before writing collection data. Run collection commands such as `init`, `install`, `add`, and `status` inside the data directory. Informational commands such as `version` and `help` can run anywhere.

If you have issues getting Chromium / Google Chrome or other dependencies working with ArchiveBox, see the *Chromium Install* and *Troubleshooting* pages for more detailed instructions.

Next Steps: Add some URLs to archive and try out CLI / Web UI

For guides on how to import URLs from different sources into ArchiveBox, check out Input Formats and *Preparing URLs*.

```
cd ~/archivebox/data
```

```
# feed in your URLs to start archiving!
archivebox add --help
archivebox add < ~/Downloads/bookmarks_export.html
```

```
# inspect the newly added Snapshots via the CLI
archivebox list
archivebox status
```

```
# OR start the webserver and view them in the Web UI
archivebox server 0.0.0.0:5797
# Visit http://admin.archivebox.localhost:5797/admin/ in a browser to finish setup
```

See our *Usage* Wiki documentation page for more examples.

Next Steps: *Upgrading Archivebox to a new version*

Stop any running ArchiveBox processes and back up the entire collection first. Upgrade ArchiveBox with the same package manager you originally used; `archivebox install` will then re-resolve compatible host binaries and update any managed runtime dependencies.

```
# back up the full collection before upgrading
cd ~/archivebox
tar -czf "archivebox-data-$(date +%s).tar.gz" data/

# get the dev version of ArchiveBox
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox>=0.9.0rc0,<0.10
↪'
# or: sudo apt update && sudo apt install --only-upgrade archivebox
# or: brew update && brew upgrade archivebox

# run init inside any data directories to migrate the index to the latest version
cd ~/archivebox/data
archivebox init          # update collection index & apply any migrations
archivebox install       # update runtime dependencies to latest versions
```

(continues on next page)

(continued from previous page)

```
archivebox update --migrate-only  # migrate/reconcile Snapshot files and metadata
archivebox status        # check collection health after the upgrade
```

Check our more detailed *Upgrading* documentation and Release Notes if you run into any problems.

Further Reading

- Read *Usage* to learn how to use the ArchiveBox CLI and HTML output
- Read *Configuration* to learn about the various archive method options
- Read *Scheduled Archiving* to learn how to set up automatic daily archiving
- Read *Publishing Your Archive* if you want to host your archive for others to access online
- Read *Troubleshooting* if you encounter any problems

1.3.3 Docker

For desktop browser cookies and bookmarks, see *Chrome profile setup*.

Overview

Running ArchiveBox with Docker allows you to manage it in a container without exposing it to the rest of your system. ArchiveBox generally works the same in Docker as it does outside Docker. You can even use uv-installed ArchiveBox and Docker ArchiveBox in tandem, as they both share the same data directory format.

- *Overview*
- *Docker Compose* (recommended)
  - *Setup*
  - *Upgrading*
  - *Usage*
  - *Accessing the data*
  - *Configuration*
- *Plain Docker*
  - *Setup*
  - *Upgrading*
  - *Usage*
  - *Accessing the data*
  - *Configuration*

Official Docker Hub image: `hub.docker.com/r/archivebox/archivebox`

```
docker pull archivebox/archivebox:dev
```

- `Dockerfile`
- `docker-compose.yml`

Published Docker tags:

- `:dev` for unstable alpha builds (breaks often, only for developers and willing beta testers)
- `:x.xrcN` and `:x.x.xrcN` for specific RC versions
- `:sha-xxxxxxx` for builds of specific git commits (to test or pin specific PRs or commits)

[!IMPORTANT] *Make sure Docker is **installed** and up-to-date before following any instructions below!* Check both commands before continuing: `docker --version` and `docker compose version` (Compose v2 is required).

Docker Compose

Setup

A full `docker-compose.yml` file is provided with all the extras included.
You can uncomment sections within it to enable extra features, or run the basic version as-is.

```
# create a folder to store your data (can be anywhere)
mkdir -p ~/archivebox/data && cd ~/archivebox

# download the compose file into the directory
curl -fsSL 'https://docker-compose.archivebox.io' > docker-compose.yml
# (shortcut for getting https://raw.githubusercontent.com/ArchiveBox/ArchiveBox/dev/
↪docker-compose.yml)

# pull and start the current image
# (the server initializes a new collection automatically)
docker compose pull
docker compose up -d --wait
```

Open http://admin.archivebox.localhost:5797 and follow the setup wizard to create the first admin and configure web access. Existing `BASE_URL` and security settings are used as-is, so configured servers skip the web-access wizard.

ArchiveBox installs and enables both ripgrep and Sonic. Sonic is selected by default in the UI, while ripgrep remains available as the fallback. To select ripgrep explicitly:

```
docker compose exec archivebox archivebox config --set SEARCH_BACKEND_ENGINE=ripgrep
```

Upgrading

See the wiki page on *Upgrading or Merging Archives: Upgrading with Docker Compose* for instructions.

Usage

With the server running from the setup steps above, use `docker compose exec archivebox archivebox [subcommand]` just like the non-Docker `archivebox [subcommand]` CLI. If the server is stopped, use `docker compose run --rm archivebox [subcommand]` instead.

First, make sure you're `cd`'ed into the same folder as your `docker-compose.yml` file (e.g. `~/archivebox`):

```
docker compose exec archivebox archivebox help
```

To add an individual URL, pass it in as an arg or via stdin:

```
docker compose exec archivebox archivebox add 'https://example.com'
# OR
echo 'https://example.com' | docker compose exec -T archivebox archivebox add
```

To add multiple URLs at once, pipe them in via stdin, or place them in a file inside `./data/sources` so that ArchiveBox can access it from within the container:

```
# pipe URLs in from a file outside Docker
docker compose exec -T archivebox archivebox add < ~/Downloads/example_urls.txt

# OR ingest URLs from a file mounted inside Docker
docker compose exec archivebox archivebox add --depth=1 /data/sources/example_urls.txt

# OR pipe in URLs from a remote source
curl 'https://example.com/some/rss/feed.xml' | docker compose exec -T archivebox␣
↪archivebox add
docker compose exec archivebox archivebox add --depth=1 'https://example.com/some/rss/
↪feed.xml'
```

The `--depth=1` flag tells ArchiveBox to look inside the provided source and archive all the URLs within:

```
# this archives just the RSS file itself (probably not what you want)
docker compose exec archivebox archivebox add 'https://example.com/some/feed.rss'

# this archives the RSS feed file + all the URLs mentioned inside of it
docker compose exec archivebox archivebox add --depth=1 'https://example.com/some/feed.
↪rss'
```

Accessing the data

The outputted archive data is stored in `data/` (relative to the project root), or whatever folder path you specified in the `docker-compose.yml` `volumes:` section. The mounted directory must be writable by its current owner; by default the entrypoint detects that non-root owner and runs ArchiveBox with matching permissions.

Docker `PUID` / `PGID`

`PUID` and `PGID` are optional Docker-entrypoint environment variables for mounts with a fixed numeric identity, such as NFS/CIFS/FUSE. They override owner autodetection and are not ArchiveBox config keys: do not put them in `ArchiveBox.conf` or use `archivebox config --set`. Both must be numeric. `PUID=0` is ignored in favor of the detected/default non-root UID so ArchiveBox and Chrome never run as root; `PGID=0` is permitted for a group-writable mount.

```
PUID=1000 PGID=1000 docker compose up -d --wait
```

Leave them unset for normal local bind mounts. When you do set them, match the effective numeric owner/group or ACL configured by the storage server; container root cannot override server-side permission enforcement such as NFS `root_squash`.

To access a result directly via the filesystem, follow its backwards-compatible `./data/archive/<timestamp>` symlink, or browse the canonical `./data/archive/users/<user>/snapshots/<date>/<domain>/<uuid>/` tree.

Alternatively, to use the web UI, start the server with:

```
docker compose up          # add -d to run in the background
```

Then open `http://web.archivebox.localhost:5797` for the public UI or `http://admin.archivebox.localhost:5797` for the admin UI.

Use port **5797** for new deployments. The Docker container also listens on the old port **8000** so existing port mappings remain compatible.

Configuration

ArchiveBox running with `docker compose` accepts all the same config options as other ArchiveBox distributions, see the full list of options available on the *Configuration* page.

The recommended way configure ArchiveBox in Docker Compose is using `archivebox config --set ...` or by editing `ArchiveBox.conf`.

```
docker compose exec archivebox archivebox config --set TIMEOUT=120
# OR edit ./data/ArchiveBox.conf and add this under its existing [ARCHIVING_CONFIG]␣
↪section:
TIMEOUT=120

# plugin-specific options work the same way (see https://plugins.archivebox.io/)
docker compose exec archivebox archivebox config --set YTDLP_MAX_SIZE=750m
```

This will apply the config to all containers or archivebox instances that access the collection.

If you're only running one container, or if you want to scope config options to only apply to a particular container, you can set them in that container's `environment:` section:

```
...

services:
    archivebox:
        ...
        environment:
            - USE_COLOR=False
            - SHOW_PROGRESS=False
            - CHECK_SSL_VALIDITY=False
            - RESOLUTION=1900,1820
            - MEDIA_TIMEOUT=512000
        ...
```

For public HTTPS, start the default stack with `docker compose up -d`, use port `5797` only as the temporary setup/upstream endpoint, and follow the first-run wizard. It gives the DNS, upstream, and certificate settings to enter in Cloudflare, Nginx Proxy Manager, Caddy, Traefik, Tailscale, or your hosting platform's ingress UI, then verifies the public HTTPS URLs before saving `BASE_URL` and `SERVER_SECURITY_MODE`.

Use exactly one of these certificate layouts:

- **Single-domain mode:** one certificate for the `BASE_URL` hostname, proxied to ArchiveBox port `5797`.
- **Isolated-subdomain mode:** one certificate covering both the `BASE_URL` hostname and `*.BASE_URL`, normally obtained through DNS-01.

Never enable on-demand TLS or request individual certificates for `snap-*` hostnames.

Docker

Setup

Fetch and run the ArchiveBox Docker image. Starting the server creates the initial archive automatically.

```
docker pull archivebox/archivebox:dev

mkdir -p ~/archivebox/data && cd ~/archivebox/data
docker run -d --name archivebox -v "$PWD:/data" -p 5797:5797 archivebox/archivebox:dev
```

Then open `/admin/` on the hostname or IP used to reach ArchiveBox (local example: http://admin.archivebox.localhost:5797/admin/) to create the first admin. If `BASE_URL` is not configured yet, continue through the web setup wizard.

(You can create a collection in any directory you want, `~/archivebox/data` is just used as an example here)

If you encounter permissions issues, make sure the mounted data directory is writable by its intended owner. Docker startup uses explicit *PUID/PGID* values when supplied, otherwise the first non-root owner detected from the existing collection or the default `archivebox` user when the data directory is root-owned.

Upgrading

See the wiki page on *Upgrading or Merging Archives: Upgrading with plain Docker* for instructions.

Usage

The Docker CLI `docker run ... archivebox/archivebox:dev [subcommand]` works just like the non-Docker `archivebox [subcommand]` CLI.

First, make sure you're `cd`'ed into your collection data folder (e.g. `~/archivebox/data`).

```
docker run -it -v $PWD:/data archivebox/archivebox:dev help
```

To add a single URL, pass it as an arg or pipe it in via stdin:

```
docker run -it -v $PWD:/data archivebox/archivebox:dev add 'https://example.com'
# OR
echo 'https://example.com' | docker run -i -v $PWD:/data archivebox/archivebox:dev add
```

To archive multiple URLs at once, pass text containing URLs in via stdin:

```
docker run -i -v $PWD:/data archivebox/archivebox:dev add < urls.txt
# OR
curl 'https://example.com/some/rss/feed.xml' | docker run -i -v $PWD:/data archivebox/
↪archivebox:dev add
```

You can also use the `--depth=1` flag to tell ArchiveBox to recursively archive the URLs within a provided source.

```
docker run -it -v $PWD:/data archivebox/archivebox:dev add --depth=1 'https://example.
↪com/some/rss/feed.xml'
```

Accessing the data

The `docker run -v /path/on/host:/path/inside/container` flag specifies where your data dir lives on the host.

For example to use a folder on an external USB drive (instead of the current directory `$PWD` or `~/archivebox/data`):

```
docker run -it -v /media/USB-DRIVE/archivebox/data:/data archivebox/archivebox:dev ...
```

Then to view your data, you can look in the folder on the host `/media/USB-DRIVE/archivebox/data`, or use the Web UI:

```
docker run -it -v /media/USB-DRIVE/archivebox/data:/data -p 5797:5797 archivebox/
↪archivebox:dev
# then open http://web.archivebox.localhost:5797
```

Configuration

The easiest way is to use `archivebox config --set KEY=value` or edit `./ArchiveBox.conf` (in your collection dir).

For example, this sets `TIMEOUT=120` as a persistent setting for the collection:

```
docker run -it -v $PWD:/data archivebox/archivebox:dev config --set TIMEOUT=120
# OR edit ./ArchiveBox.conf and add this under its existing [ARCHIVING_CONFIG] section:
TIMEOUT=120
```

ArchiveBox in Docker also accepts config as environment variables, see more on the *Configuration* page (and the abx-plugins config reference for per-plugin options).

For example, this disables the screenshot extractor for a single run (without persisting for other runs):

```
docker run -it -v $PWD:/data -e SCREENSHOT_ENABLED=False archivebox/archivebox:dev add
↪'https://example.com'
# OR
echo 'SCREENSHOT_ENABLED=False' >> ./.env
docker run ... --env-file=./.env archivebox/archivebox:dev ...
```

1.3.4 Configuration

Configuration of ArchiveBox is done by using the `archivebox config` command, modifying the `ArchiveBox.conf` file in the data folder, or by setting environment variables as process defaults. All three methods work in Docker as well.

Some equivalent examples of setting some configuration options:

```
archivebox config --set TIMEOUT=120
# OR edit ArchiveBox.conf and add this under its existing [ARCHIVING_CONFIG] section:
TIMEOUT=120
# OR
env TIMEOUT=120 archivebox add ~/Downloads/bookmarks_export.html
```

Environment variables seed process-level defaults. Persisted Machine, Persona, Crawl, and Snapshot settings can override them depending on scope, and existing Crawl config is not silently overwritten by later environment changes. Runtime-derived values like crawl/snapshot output dirs are resolved fresh for each run instead of being stored in frozen crawl config. For more examples see *Usage: Configuration*...

Available Configuration Options:

- *General Settings:* Archiving process, output format, crawl limits, and retention.
- *Server Settings:* Web UI, authentication, subdomain routing, and reverse proxy options.
- *Storage Settings:* File layout, permissions, and temp/lib directories.
- *Database Settings:* SQLite tuning and lock-retry behavior.
- *Search Settings:* Full-text search backend selection.
- *Shell Options:* Format & behavior of CLI output.
- *Plugin Configuration:* Per-plugin options (now documented separately).

General Settings

General options around the archiving process, output format, retention, and concurrency limits.

ONLY_NEW

Possible Values: [True]/False Controls what happens when you add a URL that **already has a Snapshot** in your index.

- **True** (default) — skip the URL entirely. No new Snapshot is created, no extractors run. The existing Snapshot is left exactly as-is.
- **False** — create a **new** Snapshot for the URL (separate UUID, separate output directory) and run every enabled extractor on it. The previously archived Snapshot is preserved untouched; you end up with two side-by-side captures of the same URL.

Equivalent to the --only-new / --no-only-new flag on archivebox add:

```
archivebox add https://example.com                    # honors ONLY_NEW (default True)
archivebox add --no-only-new https://example.com      # force a re-archive even if␣
↪already in the index
```

> [!NOTE] Setting ONLY_NEW=False (or --no-only-new) is the supported way to **re-capture a page that has changed since your last archive** — for example, archiving a news article, then re-archiving it later after the article was edited. Each re-archive becomes its own Snapshot row with its own timestamp.
>
> [!NOTE] Within a single crawl, URLs are deduplicated regardless of ONLY_NEW — submitting the same URL twice in one add invocation still produces only one Snapshot. ONLY_NEW only governs deduplication *against the existing index*.

Related options: *DEFAULT_PERSONA*, *URL_DENYLIST*, *URL_ALLOWLIST*

TIMEOUT

Possible Values: [60]/120/... Maximum allowed runtime **per-extractor, per-Snapshot** in seconds. If you have a slow network connection or are seeing frequent timeout errors, you can raise this value.

This is a *plugin-shared* setting — each individual extractor can override it with its own <EXTRACTOR>_TIMEOUT (e.g. WGET_TIMEOUT, CHROME_TIMEOUT, YTDLP_TIMEOUT). See the per-plugin docs for the full list.

[!NOTE] `TIMEOUT` only caps a single extractor invocation. To bound the *total* wall-clock runtime of an entire crawl, use *CRAWL_TIMEOUT* instead.

[!WARNING] Do not set this to anything less than `5` seconds — Chrome will hang indefinitely and many sites will fail completely. Anywhere between `30` and `3000` is the recommended range.

Related options: CRAWL_TIMEOUT, CRAWL_MAX_URLS, SNAPSHOT_MAX_SIZE

RESOLUTION

Possible Values: [`1440,2000`]/`1024,768`/... Default screenshot/PDF viewport resolution in `width,height` pixels. Used as the fallback for `SCREENSHOT_RESOLUTION`, `PDF_RESOLUTION`, and `CHROME_RESOLUTION`.

This is a *plugin-shared* setting — individual extractors override it via `<EXTRACTOR>_RESOLUTION` (e.g. `SCREENSHOT_RESOLUTION`, `PDF_RESOLUTION`, `CHROME_RESOLUTION`). See the per-plugin docs for plugin-specific overrides.

CHECK_SSL_VALIDITY

Possible Values: [`True`]/`False` Whether to enforce HTTPS certificate validity and HSTS chain of trust when archiving sites. Set this to `False` if you want to archive pages even if they have expired or invalid certificates.

This is a *plugin-shared* setting — every HTTP-fetching extractor (`wget`, `yt-dlp`, `gallery-dl`, `chrome`, etc.) honors it, and individual extractors can override with `<EXTRACTOR>_CHECK_SSL_VALIDITY`. See the per-plugin docs.

[!WARNING] When `False`, ArchiveBox cannot guarantee that the captured content matches the real site — a man-in-the-middle could substitute responses. Only disable for trusted networks or for archiving legacy/internal sites with expired certs.

USER_AGENT

Possible Values: [`Mozilla/5.0 ... ArchiveBox/{VERSION} ...`]/`"Mozilla/5.0 ..."`/... The default `User-Agent` string sent during archiving. The built-in default identifies ArchiveBox and links back to the GitHub repo so site operators can identify and contact archivers if needed.

This is a *plugin-shared* setting — each extractor (`wget`, `chrome`, `yt-dlp`, `singlefile`, ...) can override it with its own `<EXTRACTOR>_USER_AGENT`, otherwise it falls back to this value. See the per-plugin docs for per-extractor specifics.

[!NOTE] Some sites block requests that look like bots or that don't match a real browser. If you're getting 403s or empty responses, try setting this to a current Chrome/Firefox UA string.

COOKIES_FILE

Possible Values: [`None`]/`/path/to/cookies.txt`/...

[!TIP] **Prefer *personas* over `COOKIES_FILE` for authentication.** A persona bundles a `cookies.txt`, a Chrome user-data-dir, a user-agent, and any other per-identity state into one named profile that's swappable per-crawl and automatically scoped across every extractor. `COOKIES_FILE` (and the per-extractor `<EXTRACTOR>_COOKIES_FILE` overrides) is a low-level escape hatch for when you specifically need to point at a hand-rolled cookies file outside the persona system — most users should ignore it and configure auth through `archivebox persona create` instead.

Path to a Netscape-format `cookies.txt` file passed to `wget`, `curl`, `yt-dlp`, and other non-Chrome extractors for authentication. Required when archiving sites behind a login (paywalls, social media feeds, members-only forums, etc.) **if you're not using a persona**.

This is a *plugin-shared* setting — each extractor can override it with `<EXTRACTOR>_COOKIES_FILE` (e.g. `WGET_COOKIES_FILE`, `YTDLP_COOKIES_FILE`, `GALLERYDL_COOKIES_FILE`). Chrome-based extractors instead read auth state from the persona's `CHROME_USER_DATA_DIR`. See the per-plugin docs for per-extractor variants.

You can generate a `cookies.txt` using a browser extension, or with `wget --save-cookies + --user=... --password=...`.

The recommended path is to create a persona and let it manage cookies + Chrome profile state for you:

```
archivebox persona create --import=chrome personal
archivebox add --persona=personal https://members.example.com/feed
```

> [!WARNING] **Use separate burner credentials dedicated to archiving** — don't re-use your normal daily Facebook/Instagram/Youtube/etc. account cookies as server responses often contain your name/email/PII and session tokens, which then get preserved in your snapshots forever!

Related options: *DEFAULT_PERSONA*, CHROME_USER_DATA_DIR

DEFAULT_PERSONA

Possible Values: [Default]/personal/work/... The persona profile used when no explicit persona is selected for a new crawl. The selected persona is stored on the Crawl row; `DEFAULT_PERSONA` is not duplicated into `Crawl.config`. Personas bundle a Chrome user-data-dir, a `cookies.txt`, auth state, a user-agent, and any other per-identity config into a single named profile, letting you swap between archiving contexts (logged-out vs. signed-into-work-account vs. signed-into-personal-account) without manually juggling files.

ArchiveBox auto-creates the named persona on disk if it doesn't already exist. See the *Chromium profile setup guide* for profile setup instructions.

Related options: *COOKIES_FILE*

URL_DENYLIST

Possible Values: [\.(css|js|otf|ttf|woff|woff2|gstatic\.com|googleapis\.com/css)(\?.*)?$]/.+\.exe$/...

Regex pattern matched against every URL discovered during a crawl. Any matching URL is **excluded** from archiving — useful for blocking tracking pixels, ad networks, CDN-hosted CSS/fonts, or arbitrary file extensions you don't want to capture.

The default skips common static assets (CSS, fonts, Google Fonts CDN) so they aren't re-fetched as separate Snapshots during recursive crawls — the parent page's `singlefile/dom` output already inlines them. It also excludes Hacker News vote and hide actions using `^https?://news\.ycombinator\.com/(?:vote|hide)/?(?:[?#]|$)`, while preserving stories, comments, profiles, and domain listings. Existing crawls retain their saved filter configuration.

Note: This option is also recognized under its legacy alias URL_BLACKLIST.

Related options: *URL_ALLOWLIST*

URL_ALLOWLIST

Possible Values: [None]/^http(s)?:\/\/(.+)?example\.com\/?.*$/...

Regex pattern matched against every URL discovered during a crawl. When set, any URL that does **not** match is excluded. Useful for recursive crawling scoped to a single domain or path prefix (e.g. only follow links within `docs.example.com/v2/`).

When both are set, `URL_DENYLIST` takes precedence over `URL_ALLOWLIST`.

Note: This option is also recognized under its legacy alias `URL_WHITELIST`.

Related options: URL_DENYLIST

TAG_SEPARATOR_PATTERN

Possible Values: [[,]]/[,;]/[,;\s]/... Regex character class used to split tag strings (e.g. `news,politics;longform`) into individual tags when importing URLs. The default splits on commas only; widen it if you paste in tags separated by semicolons, spaces, or other delimiters.

CRAWL_MAX_URLS

Possible Values: [0]/50/500/... Maximum number of unique URLs (Snapshots) a single crawl is allowed to produce. `0` means unlimited. Counts both seed URLs you submitted and URLs discovered by recursive crawlers (`parse_dom_outlinks`, `parse_html_urls`, etc.).

ArchiveBox enforces this cap when admitting URLs and creating Snapshot rows in its database. Hooks still emit discoveries; reaching the cap prevents additional Snapshot rows from being created. Raising the cap and re-queuing the crawl allows further discovery. The standalone `abx-dl` downloader does not enforce crawl-wide limits.

> [!NOTE] Use this as a safety net for recursive crawls (`--depth=N`) that could otherwise blow up to thousands of pages on link-heavy sites.

Related options: CRAWL_MAX_SIZE, CRAWL_TIMEOUT, CRAWL_MAX_CONCURRENT_SNAPSHOTS, SNAPSHOT_MAX_SIZE

CRAWL_MAX_SIZE

Possible Values: [0]/50MB/5GB/104857600/... Maximum cumulative output size (in bytes) a single crawl is allowed to produce across all of its Snapshots. `0` means unlimited.

Accepts a raw byte count (`104857600`) or a unit-suffixed string (`100MB`, `5GB`, `1TiB`). ArchiveBox sums the output sizes recorded on its Snapshot rows. Once the cap is reached, in-flight Snapshots finish but no new ones start, and the crawl reports `stop_reason = "crawl_max_size"`. This is a scheduling budget, not a hard filesystem quota; concurrent captures and cleanup can exceed it.

> [!NOTE] Bounds the **disk footprint** of a crawl, not the wire transfer — a 2MB HTML page can produce 50MB of screenshots, PDFs, SingleFile bundles, and media downloads, and this cap applies to the on-disk total.

Related options: SNAPSHOT_MAX_SIZE, CRAWL_MAX_URLS, CRAWL_TIMEOUT

CRAWL_TIMEOUT

Possible Values: [0]/300/3600/... Maximum total wall-clock runtime for a single crawl in seconds. 0 means unlimited.

Distinct from *TIMEOUT*: TIMEOUT caps one extractor invocation on one Snapshot; CRAWL_TIMEOUT caps the *entire crawl* — all Snapshots, all extractors, all retries, all recursive discovery passes — together. The clock uses the earliest Snapshot created_at timestamp in the crawl and persists across resumes. Time spent queued after that Snapshot was created counts toward the budget; a crawl with no Snapshots has not started its clock. At Snapshot boundaries, an exhausted budget prevents queued Snapshots from starting and reports stop_reason = "crawl_timeout"; in-flight captures and cleanup can finish.

> [!NOTE] Useful as a scheduling budget for unattended/scheduled crawls (e.g. "spend at most 1 hour archiving Hacker News tonight"). Pair with CRAWL_MAX_URLS and CRAWL_MAX_SIZE for belt-and-suspenders bounds.

Related options: TIMEOUT, CRAWL_MAX_URLS, CRAWL_MAX_SIZE

CRAWL_MAX_CONCURRENT_SNAPSHOTS

Possible Values: [4]/1/8/16/... How many Snapshots within a single crawl ArchiveBox will archive in parallel. The runner schedules up to this many extractor pipelines at once, then waits for one to finish before starting the next.

Raising this speeds up large crawls on beefy hardware, but each concurrent Snapshot opens a new tab inside the crawl's shared Chrome instance (when Chrome-based extractors are enabled) — RAM per tab and CPU pressure scale roughly linearly with concurrency. On a typical laptop, 2-4 is sane; on a dedicated server with 32GB+ RAM, 8-16 can be reasonable.

> [!NOTE] This is **per-crawl** concurrency. If you run multiple crawls simultaneously, each one independently gets up to CRAWL_MAX_CONCURRENT_SNAPSHOTS parallel Snapshots.

Related options: CRAWL_MAX_URLS, TIMEOUT

SNAPSHOT_MAX_SIZE

Possible Values: [0]/10MB/500MB/... Maximum cumulative output size (in bytes) **per individual Snapshot**. 0 means unlimited. Same unit-suffix parsing as CRAWL_MAX_SIZE (10MB, 2GB, raw bytes, etc.).

Where CRAWL_MAX_SIZE is a *crawl-wide* budget, SNAPSHOT_MAX_SIZE puts a ceiling on any *one* page's output. The downloader tracks reported output metadata in memory for the current run, counting each plugin/path once. At hook boundaries, reaching the cap stops additional hooks from starting; background cleanup still runs to preserve recordings. Retrying starts fresh accounting and preserves partial files until the hooks overwrite them. Unrun hooks receive no ArchiveResult rows, and the rest of the crawl continues normally.

> [!NOTE] Particularly useful when crawling sites with occasional huge pages (e.g. a forum where most threads are small but a few are 500MB media galleries) — it caps the outliers without throttling the whole crawl.

Related options: CRAWL_MAX_SIZE, CRAWL_MAX_URLS

DELETE_AFTER

Possible Values: [0]/24h/7d/4w/6mo/1y/... Retention policy: automatically delete Crawls, Snapshots, ArchiveResults, and Process rows (and their on-disk outputs) after this duration has elapsed. `0`, `""`, or `None` disables auto-deletion (the default — ArchiveBox never deletes anything unless you ask).

Accepted units: `h`/`hr`/`hour`, `d`/`day`, `w`/`week`, `mo`/`month`, `y`/`yr`/`year`. The minimum non-zero duration is `1h`. Examples:

```
archivebox config --set DELETE_AFTER=24h     # daily rolling buffer
archivebox config --set DELETE_AFTER=30d     # 30-day retention
archivebox config --set DELETE_AFTER=6mo     # 6 months
```

`DELETE_AFTER` can be set globally, per-persona, per-crawl, or per-snapshot — the most-specific value wins. When a Snapshot is created, its `delete_at` timestamp is computed from the effective `DELETE_AFTER` and persisted; the retention sweeper then deletes rows whose `delete_at` is in the past.

> [!WARNING] Deletion is **destructive and irreversible**. Files in the snapshot's output directory are removed from disk. Use with care on important archives — and *never* set this on the global config if you have legacy snapshots you don't want garbage-collected.

Related options: PERMISSIONS

PERMISSIONS

Possible Values: [public]/unlisted/private Default visibility for newly created Snapshots. Inherited by every Snapshot in a Crawl unless explicitly overridden at the Crawl or Snapshot level.

- **`public`** — Snapshot appears in the public index *and* its content is directly accessible without login.
- **`unlisted`** — Snapshot content is accessible via direct link, but it is **not** listed in the public index. Equivalent to a "secret URL."
- **`private`** — Snapshot is hidden from the public index *and* its content requires admin login.

This option supersedes the removed `PUBLIC_SNAPSHOTS` boolean and is also driven by the still-current `PUBLIC_INDEX` flag — both are interpreted as a coarse mapping onto `PERMISSIONS` for backwards compatibility (`PUBLIC_SNAPSHOTS=False` `private`, `PUBLIC_INDEX=False` `unlisted`, either set to `True` `public`). Setting `PERMISSIONS` directly wins over either legacy flag.

> [!NOTE] `PERMISSIONS` controls **per-Snapshot** visibility. Server-wide auth (whether the whole UI requires login, whether the add-view is open) is still controlled by `PUBLIC_INDEX` and `PUBLIC_ADD_VIEW` under Server Settings.

Related options: `PUBLIC_INDEX`, `PUBLIC_ADD_VIEW`, *DELETE_AFTER*

PLUGINS

Possible Values: [""]/wget,favicon,screenshot/chrome,singlefile,dom/... Comma-separated **whitelist** of plugins to load and run for this archiving run. When empty (the default), ArchiveBox uses the installed/enabled plugin set — i.e. every plugin whose <PLUGIN>_ENABLED config evaluates true.

When set, only the listed plugins (plus any plugins they declare as `required_plugins` in their `config.json` — e.g. picking `singlefile` automatically pulls in `chrome`) participate in the run. Equivalent to the CLI flag:

```
archivebox add --plugins=wget,favicon,screenshot https://example.com
```

The admin "Add" form and REST API both write to PLUGINS when you select extractors — there is no separate "enabled set" config knob; PLUGINS is the single source of truth for which plugins run on any given Crawl, Snapshot, or Persona scope.

Useful for one-off runs ("just grab a screenshot and skip everything else") or for reproducible per-crawl pipelines stored on the Crawl row.

Server Settings

Options for the web UI, authentication, subdomain routing, and reverse proxy configuration.

ADMIN_USERNAME / ADMIN_PASSWORD

Possible Values: [None]/"admin"/...

Used on first run / initial setup in any installation method. ArchiveBox will create an admin superuser with the specified username and password when both options are present during `archivebox init`. After the user exists, changing these values has no effect — use `archivebox manage changepassword <username>` or the Django admin UI instead.

> [!WARNING] Setting ADMIN_PASSWORD via environment variable bakes the secret into your shell history, Docker inspect output, and process listings. For long-lived deployments, set it once during provisioning, create the user, then unset the variable.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Setting-up-Authentication

Related options: LDAP_ENABLED, REVERSE_PROXY_USER_HEADER

PUBLIC_INDEX / PUBLIC_ADD_VIEW

Possible Values: [True]/False (for PUBLIC_INDEX), [False]/True (for PUBLIC_ADD_VIEW)

Server-wide toggles for whether login is required to use each public area of ArchiveBox.

```
archivebox config --set PUBLIC_INDEX=True        # allow viewing the snapshot index␣
↪without login
archivebox config --set PUBLIC_ADD_VIEW=False    # require login to submit new URLs via␣
↪the web UI
```

- PUBLIC_INDEX (default True) — when on, anonymous visitors can browse the snapshot list page. Individual snapshot visibility is still gated by each Snapshot's own *PERMISSIONS* field.
- PUBLIC_ADD_VIEW (default False) — when on, anonymous visitors can submit new URLs to be archived via the /add form. Leave this off on any internet-exposed instance unless you actively want a public submission endpoint.

  [!NOTE] **PUBLIC_SNAPSHOTS has been removed as a global toggle.** Snapshot visibility is now decided per-Snapshot via the *PERMISSIONS* field (public / unlisted / private) under General Settings. The old anchors are preserved on PERMISSIONS so existing links keep working.

Related options: *PERMISSIONS*, *SERVER_SECURITY_MODE*, *ADMIN_USERNAME*

SECRET_KEY

Possible Values: *auto-generated 50-character random string*

Django's secret key, used for cryptographic signing of sessions, CSRF tokens, password reset links, and other signed payloads. Auto-generated on first server start and persisted to `ArchiveBox.conf` so it survives restarts. If the config file isn't writable (read-only mount, mid-init race), an in-memory random key is used and all users are logged out on the next boot.

> [!WARNING] Treat this value like a password. Anyone with the `SECRET_KEY` can forge sessions and CSRF tokens for your instance. Don't commit `ArchiveBox.conf` to public repos, and rotate it (forcing all users to log in again) if you suspect it's been exposed.

BIND_ADDR

Possible Values: [`127.0.0.1:5797`]/`0.0.0.0:5797`/`[::]:5797`/`0.0.0.0:80`/...

The `host:port` socket the ArchiveBox web server actually listens on. **This is the local bind socket, not the public URL** — for the public URL clients see, set *BASE_URL*.

- `127.0.0.1:5797` (default) — listen only on the loopback interface. Safest when you're running a reverse proxy on the same host and don't want the server reachable directly from the network.
- `0.0.0.0:5797` — listen on **all** IPv4 interfaces. Required when running in Docker without `--network=host`, or when you want the server reachable from other machines on your LAN without a reverse proxy.
- `[::]:5797` — listen on all IPv6 interfaces (most modern OSes will accept v4-mapped connections too). IPv6 literal addresses must be bracketed: `[::1]:5797`, not `::1:5797`.

  > [!NOTE] Inside Docker, binding to `127.0.0.1` means the server is unreachable from outside the container — use `0.0.0.0:5797` and let Docker handle the port-forwarding, or publish the port with `-p 127.0.0.1:5797:5797` on the host side instead.

Related options: *BASE_URL*, *SERVER_SECURITY_MODE*

BASE_URL

Possible Values: [`""`]/`https://archive.example.com`/`http://archivebox.localhost:5797`/...

The canonical public URL of your ArchiveBox instance. Used to build absolute links in templates, redirects (`/admin/login/?next=...`), admin notification emails, OG/meta tags, and — in subdomain security mode — to derive the `admin.`, `web.`, `api.`, and per-snapshot `snap-<id>.` subdomains.

When `BASE_URL` is set explicitly, ArchiveBox treats it as the source of truth and ignores the incoming `Host` header for URL building. In `safe-subdomains-fullreplay` mode setting it is **required for redirects to work correctly**.

When `BASE_URL` is empty, the value is resolved at request time from the incoming request's `Host` header (with any leading `admin.` / `web.` / `api.` / `snap-*.` label stripped to recover the canonical base). Loopback hostnames (`localhost`, `127.0.0.1`, `0.0.0.0`, `::`) are rewritten to `archivebox.localhost` so subdomain routing works without `/etc/hosts` edits. If there's no live request, *BIND_ADDR* is used as a last resort.

The scheme is taken from the explicit BASE_URL if set, otherwise from the request (so put a reverse proxy in front for HTTPS and trust X-Forwarded-Proto).

ArchiveBox automatically derives the underlying Django ALLOWED_HOSTS and CSRF_TRUSTED_ORIGINS settings from BASE_URL + *SERVER_SECURITY_MODE*, so you do **not** set those directly — the system widens them as needed to admit the admin/web/api subdomains.

> [!NOTE] **Pin BASE_URL explicitly on any deployment using safe-subdomains-fullreplay mode.** A misconfig banner will surface in the rendered UI until you do.
>
> [!NOTE] **Legacy upgrade path (0.7.3 → 0.9):** archivebox init preserves the complete legacy config and migrates the old ARCHIVE_BASE_URL, ADMIN_BASE_URL, or documented LISTEN_HOST hostname to BASE_URL. A single non-default CSRF_TRUSTED_ORIGINS or ALLOWED_HOSTS entry is used as a fallback when those settings are absent. New installs should set BASE_URL directly; the legacy hostname settings are no longer user-settable knobs.

Related options: *SERVER_SECURITY_MODE*, *BIND_ADDR*

SERVER_SECURITY_MODE

Possible Values: [auto]/safe-subdomains-fullreplay/safe-onedomain-nojsreplay/unsafe-onedomain-noadmin/danger-

The top-level security posture of the server. Controls how archived content is served, whether the admin/API control plane is reachable, and which host(s) the UI is split across. **This is the most important security knob** — pick the most restrictive mode that still works for your use case.

ArchiveBox splits its surfaces across three logical hosts: admin.* (Django admin + session cookies, the entire control plane), web.* (anonymous public browsing UI, optional public add view, and public snapshot index), and api.* (REST/JSON endpoints). In subdomain mode each gets its own host derived from *BASE_URL*; session/CSRF cookies are scoped to admin.* only, so web.*, api.*, and compromised replay pages on snap-<id>.* can't read admin auth.

Mode	Host layout	JS replay	Control plane	Use when
auto *(default, recommended*	Subdomains for any *.localhost request; otherwise one host	Full raw replay on *.localhost; otherwise raw archived HTML uses no-JS replay. Plugins can opt into a trusted viewer with an explicit full.html preview template.	Enabled	The zero-configuration default. Localhost gets the highest-fidelity isolated setup; ordinary public/LAN hostnames do not require wildcard DNS or TLS.
safe-s	admin/web/a * on separate subdomains	Full JS replay enabled	Enabled on admin.* only	You have wildcard DNS (*.archive.example.com) and a TLS cert that covers it. Archived JS runs sandboxed away from the admin origin.
safe-o	Everything on one host	Raw archived HTML cannot run JS. Explicit trusted plugin preview templates can run the JS needed by their viewer.	Enabled	You can't get wildcard DNS. Trades raw replay fidelity for same-origin safety while retaining trusted viewer formats.
unsafe	Everything on one host	Full JS replay enabled	**Disabled** — /admin, /accounts, /api, /add, /web return 403; only GET/HEAD/OPTIC allowed	Read-only public archive on a single host. Operate the instance via CLI only; the web admin is unreachable.
danger	Everything on one host	Full JS replay enabled	Enabled	Local dev / trusted-network only. Archived JS runs on the **same origin as the admin UI** — a malicious archived page can call admin endpoints with your session. **Do not expose this mode to the internet.**

SingleFile output is served as ordinary HTML in every mode; it remains usable in no-JS modes because SingleFile removes the page's scripts during capture. ArchiveWeb.page/ReplayWeb.page and MHTML use their existing trusted preview templates. ArchiveBox always attempts to load these viewers, including when the incoming request is plain HTTP, because HTTPS may be terminated by an upstream proxy. Browser service-worker rules still require ReplayWeb.page to be reached through HTTPS or localhost for replay to initialize.

Trusted preview wrappers allow embedding by their own origin and the configured web/admin origins, so the collection UI can display snapshot-host previews without opening them to arbitrary embedding sites. PDF previews use a document frame rather than an object/embed, retaining object-src 'none' on the wrapper. Allowing a trusted viewer's scripts does not exempt the archived documents inside it from their replay policy; the MHTML viewer keeps its archived child document sandboxed without scripts.

> [!WARNING] Switching to any mode whose name starts with unsafe- or danger- is logged at startup and surfaces a banner in the UI. **Don't use these modes on a public hostname** — archived JavaScript will run on the same origin as your admin session.
>
> [!NOTE] Explicit subdomain mode requires both wildcard DNS (*.archive.example.com) and (if using TLS) a wildcard certificate. The default auto mode only selects subdomain routing for *.localhost, which browsers resolve locally without custom DNS or TLS setup.
>
> [!NOTE] In one-domain auto mode, read-only requests are accepted through any valid ingress hostname, but POST/PUT/PATCH/DELETE requests are accepted only on the canonical BASE_URL host. Canonical

links always use BASE_URL. Explicit subdomain mode continues to allow state-changing requests on its derived admin/API hosts.

Related options: *BASE_URL*, *PERMISSIONS*

More info:

- *Security Overview*

SNAPSHOTS_PER_PAGE

Possible Values: [50]/25/100/200/...

Number of rows to render per page on the Snapshot and ArchiveResult list views (admin index, public index, and the Django admin changelists for both models). Lower values reduce per-request render time; higher values speed up bulk browsing at the cost of heavier rendering. Must be 1.

FOOTER_INFO

Possible Values: [Content is hosted for personal archiving purposes only. Contact server owner for any takedown requests.]/...

Free-form text rendered in the footer of every archive page. Useful for adding a takedown contact, an org disclaimer, or attribution. Plain text — no HTML.

REVERSE_PROXY_USER_HEADER

Possible Values: [Remote-User]/X-Remote-User/X-Forwarded-User/...

HTTP header your reverse proxy (Authelia, oauth2-proxy, Authentik, nginx auth_request, etc.) sets to the authenticated username. ArchiveBox's ReverseProxyAuthMiddleware reads this header **only when the request's source IP is inside *REVERSE_PROXY_WHITELIST*** — otherwise the header is ignored to prevent direct-connect spoofing.

The header name is matched case-insensitively and normalized to the HTTP_* form Django exposes (e.g. Remote-User → HTTP_REMOTE_USER).

Related options: *REVERSE_PROXY_WHITELIST*, *LOGOUT_REDIRECT_URL*

REVERSE_PROXY_WHITELIST

Possible Values: [""]/172.16.0.0/16/10.0.0.5/32,fd00::/8/...

Comma-separated list of IPv4 / IPv6 addresses or CIDR networks that are trusted to set *REVERSE_PROXY_USER_HEADER*. When empty (the default), reverse-proxy auth is **completely disabled** — the header is never consulted no matter who set it.

When non-empty, only requests whose REMOTE_ADDR falls inside one of the listed networks have the header honored. Anything else falls back to standard session auth. The CIDR list is validated on every request; an invalid entry raises ImproperlyConfigured and breaks the server, so test changes carefully.

[!WARNING] **Set this to the actual IP of your reverse proxy, never 0.0.0.0/0 or a public network.** With a wide-open whitelist, anyone who can reach the server directly can forge any username they like via the header.

Related options: *REVERSE_PROXY_USER_HEADER*, *LOGOUT_REDIRECT_URL*

LOGOUT_REDIRECT_URL

Possible Values: [/]/https://example.com/some/other/app//accounts/logout-landing//...

URL users are redirected to after logging out. The default / keeps users on ArchiveBox; set this to an external URL when using reverse-proxy SSO so logout terminates the upstream session too (e.g. `https://auth.example.com/logout`).

Related options: *REVERSE_PROXY_USER_HEADER*, *REVERSE_PROXY_WHITELIST*

LDAP Settings

Options for LDAP / Active Directory authentication via django-auth-ldap. Requires `uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox[ldap]>=0.9.0rc0,<0.10'` *(which also pulls in the system* `libldap` / `libsasl` *headers).*

LDAP_ENABLED

Possible Values: [False]/True

Master switch for LDAP authentication. When `True`, ArchiveBox loads the `django-auth-ldap` backend and validates that `LDAP_SERVER_URI`, `LDAP_BIND_DN`, `LDAP_BIND_PASSWORD`, and `LDAP_USER_BASE` are all set — startup fails fast otherwise.

```
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox[ldap]>=0.9.
↪0rc0,<0.10'
```

Then set these configuration values:

```
LDAP_ENABLED: True
LDAP_SERVER_URI: "ldap://ldap.example.com:3389"
LDAP_BIND_DN: "ou=archivebox,ou=services,dc=ldap.example.com"
LDAP_BIND_PASSWORD: "secret-bind-user-password"
LDAP_USER_BASE: "ou=users,ou=archivebox,ou=services,dc=ldap.example.com"
LDAP_USER_FILTER: "(uid=%(user)s)"
LDAP_USERNAME_ATTR: "username"
LDAP_FIRSTNAME_ATTR: "givenName"
LDAP_LASTNAME_ATTR: "sn"
LDAP_EMAIL_ATTR: "mail"
LDAP_CREATE_SUPERUSER: False
```

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Setting-up-Authentication
- https://github.com/django-auth-ldap/django-auth-ldap#example-configuration

Related options: *ADMIN_USERNAME*, *REVERSE_PROXY_USER_HEADER*

LDAP_SERVER_URI

Possible Values: [None]/ldap://ldap.example.com:389/ldaps://ldap.example.com:636/...

URI of the LDAP server to bind against. Use ldaps:// for TLS or ldap:// for plaintext (plus optional StartTLS at the protocol level). Required when *LDAP_ENABLED* is True.

LDAP_BIND_DN

Possible Values: [None]/cn=archivebox,ou=services,dc=example,dc=com/...

Distinguished name of the service account used to perform user searches. This account only needs read access to the user subtree under *LDAP_USER_BASE*. Required when *LDAP_ENABLED* is True.

LDAP_BIND_PASSWORD

Possible Values: [None]/<bind-service-password>/...

Password for the *LDAP_BIND_DN* service account. Required when *LDAP_ENABLED* is True.

> [!WARNING] Treat this like any other service credential — keep it out of shell history and version control. Prefer setting it via the config file (which has owner-only permissions) over environment variables.

LDAP_USER_BASE

Possible Values: [None]/ou=users,dc=example,dc=com/...

Base DN under which to search for user entries. Required when *LDAP_ENABLED* is True. The search is performed as LDAP_BIND_DN with the filter from *LDAP_USER_FILTER*.

LDAP_USER_FILTER

Possible Values: [(uid=%(user)s)]/(sAMAccountName=%(user)s)/(&(objectClass=person)(mail=%(user)s))/...

LDAP search filter used to find a user entry at login. The literal token %(user)s is replaced with the username the user typed into the login form. Common values:

- (uid=%(user)s) — OpenLDAP-style
- (sAMAccountName=%(user)s) — Active Directory
- (mail=%(user)s) — match by email

LDAP_USERNAME_ATTR

Possible Values: [username]/uid/sAMAccountName/...

LDAP attribute on the user entry that becomes the local Django username. Must be unique within the directory.

LDAP_FIRSTNAME_ATTR

Possible Values: [givenName]/...

LDAP attribute mapped to Django's User.first_name.

LDAP_LASTNAME_ATTR

Possible Values: [sn]/...

LDAP attribute mapped to Django's User.last_name.

LDAP_EMAIL_ATTR

Possible Values: [mail]/userPrincipalName/...

LDAP attribute mapped to Django's User.email.

LDAP_CREATE_SUPERUSER

Possible Values: [False]/True

When True, every LDAP user who successfully authenticates is auto-promoted to Django superuser. **Off by default** — leave it off unless your directory's user base is already restricted to operators, since superusers can modify config, delete snapshots, and run server commands.

> [!WARNING] Combining LDAP_CREATE_SUPERUSER=True with a broad *LDAP_USER_BASE* (e.g. an entire company OU) effectively grants admin to every employee. Scope the user base or use group-based access control via django-auth-ldap's AUTH_LDAP_USER_FLAGS_BY_GROUP (configured in custom settings.py) instead.

Storage Settings

Options for the on-disk layout, file permissions, and temp/lib directories that ArchiveBox reads and writes during archiving.

> [!NOTE] Docker-only *PUID and PGID* are entrypoint environment variables, not ArchiveBox configuration keys.

OUTPUT_PERMISSIONS

Possible Values: [644]/755/... Permissions to set on output files written into the archive directory. The directory mode is derived from this by OR-ing in the execute bits (so 644 files imply 755 dirs), which subsumes the legacy DIR_OUTPUT_PERMISSIONS option (formerly a separate 755-default field) — directory mode is no longer settable on its own.

> [!NOTE] Set this to 600 if you want archives to be readable only by the ArchiveBox user, or 664/775 if you need a shared group to read/write the data dir.

Related options: *ENFORCE_ATOMIC_WRITES*

- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#docker-permissions-issues

---

ENFORCE_ATOMIC_WRITES

Possible Values: [True]/False Whether to write output files atomically (write to a tempfile + rename() into place) so that a crash or kill -9 mid-write can never leave a partial file in the archive. Disable only if you are debugging a filesystem that doesn't support atomic renames (some FUSE mounts).

TMP_DIR

Possible Values: [data/tmp/<machine_id>]/tmp/archivebox/abc5d851/... Path for temporary files, the supervisord unix socket, and generated supervisor config. The default is a per-machine subdirectory under the data folder (tmp/<machine_id>) so multiple machines sharing the same data dir (e.g., over NFS) don't collide on socket files.

> [!WARNING] TMP_DIR *must* be a short, local path readable/writable by the ArchiveBox user. Unix socket paths have a hard ~96-character limit, so a deeply nested TMP_DIR will silently break the supervisor. It also must live on a real local filesystem (tmpfs/SSD) — FUSE, network mounts, and Docker bind mounts on macOS often cannot host unix sockets at all (see *ALLOW_NO_UNIX_SOCKETS*).

If ArchiveBox detects the configured TMP_DIR is unwritable or too long, it will auto-fall-back to /tmp/archivebox/<collection_id> at startup.

Related options: *ABXPKG_LIB_DIR*, *ALLOW_NO_UNIX_SOCKETS*

ABXPKG_LIB_DIR

Possible Values: [<user-config>/abx/lib]/opt/archivebox/lib/~/.config/abx/lib/... Path for installed binary dependencies (chromium, single-file, yt-dlp, ripgrep, etc.) managed by abxpkg. The default is the platform-standard user-config location (~/.config/abx/lib on Linux, ~/Library/Application Support/abx/lib on macOS, %APPDATA%\abx\lib on Windows) so a single binary installation can be shared across multiple collections on the same machine without re-downloading.

> [!NOTE] ABXPKG_LIB_DIR can grow to several GB. Put it on a fast local disk — running extractors off a network-mounted ABXPKG_LIB_DIR will be painfully slow.

Related options: *TMP_DIR*

ALLOW_NO_UNIX_SOCKETS

Possible Values: [False]/True **Alias:** ARCHIVEBOX_ALLOW_NO_UNIX_SOCKETS

Skip the startup check that verifies *TMP_DIR* can host unix-domain sockets (a real bind() on a .sock file). Set to True only when running ArchiveBox on a filesystem that cannot back unix sockets — most commonly Docker Desktop on macOS with a host bind-mounted TMP_DIR, where the osxfs/virtiofs layer rejects bind() calls.

> [!WARNING] This disables a real safety check, not a cosmetic one. When unix sockets are unavailable some plugins that talk to long-lived helpers over .sock files (supervisord control socket, browser launcher RPC) may behave unpredictably. Prefer fixing *TMP_DIR* to point at a tmpfs/SSD inside the container; reach for ALLOW_NO_UNIX_SOCKETS only when that's genuinely not possible.

Related options: *TMP_DIR*

Database Settings

Options for choosing and tuning the index database that backs ArchiveBox's snapshot, tag, and crawl metadata.

ArchiveBox stores all of its index metadata in a single SQLite database file (`index.sqlite3` inside your data directory) by default, or optionally in a PostgreSQL database (see `DATABASE_ENGINE`). The defaults are tuned for nearly all users — the knobs below mostly govern **lock-contention behavior**, which matters when multiple workers touch the database concurrently (e.g. supervised orchestrators, parallel `archivebox add` runs, container restarts that race against an in-flight write, or long-running web/admin processes alongside CLI commands).

> [!NOTE] These are advanced operator tuning options. If you are not actively diagnosing `database is locked` errors or planning a non-default storage layout, you can safely leave everything in this section at its default.

Learn more:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#database-and-filesystem-contention-issues-when-running-mu
- https://www.sqlite.org/wal.html
- https://www.sqlite.org/pragma.html

DATABASE_ENGINE

Possible Values: `[sqlite]/postgres` Which database backend to use for the main index. Settable as `ARCHIVEBOX_DATABASE_ENGINE` or under `[DATABASE_CONFIG]` in `ArchiveBox.conf`.

The default `sqlite` keeps everything in a single `index.sqlite3` file inside the data directory and requires no external services. Set this to `postgres` to store the index in a PostgreSQL database instead — useful for large collections with many concurrent writers, or when the data directory lives on a filesystem where SQLite performs poorly (e.g. network mounts).

With `postgres`, connection details come from the options below. `archivebox init` will create the configured database automatically if the server is reachable and the database does not exist yet. Only the *index* moves to PostgreSQL — snapshot output files stay in `./archive/` in the data directory, and plugin-owned sidecar databases (e.g. the `search.sqlite3` full-text index) are unaffected.

```
# example ArchiveBox.conf
[DATABASE_CONFIG]
DATABASE_ENGINE = postgres
DATABASE_NAME = archivebox
DATABASE_HOST = 127.0.0.1
DATABASE_PORT = 5432
DATABASE_USER = archivebox
DATABASE_PASSWORD = s3cret
```

> [!WARNING] Pick a backend when you first run `archivebox init` and stick with it. There is no built-in tool (yet) to move an existing collection's index between SQLite and PostgreSQL.

Related options: `DATABASE_NAME`, `DATABASE_HOST`

DATABASE_HOST / DATABASE_PORT / DATABASE_USER / DATABASE_PASSWORD

Possible Values: [127.0.0.1] / [5432] / [archivebox] / [empty] PostgreSQL connection settings, used only when DATABASE_ENGINE=postgres. Settable as ARCHIVEBOX_DATABASE_HOST, ARCHIVEBOX_DATABASE_PORT, ARCHIVEBOX_DATABASE_USER, and ARCHIVEBOX_DATABASE_PASSWORD.

DATABASE_HOST may also be a path to a directory containing a PostgreSQL unix socket (e.g. /var/run/postgresql).

DATABASE_NAME

Possible Values: [index.sqlite3] / archivebox / ... The main index database. Settable as ARCHIVEBOX_DATABASE_NAME.

With the default DATABASE_ENGINE=sqlite, this is the path to the SQLite index file inside the data directory (index.sqlite3). With DATABASE_ENGINE=postgres, it is the name of the PostgreSQL database instead (default: archivebox), created automatically by archivebox init if it does not exist.

SQLITE_JOURNAL_MODE

Possible Values: [WAL on native installs]/[DELETE in Docker]/TRUNCATE/PERSIST/MEMORY/OFF SQLite journal mode, applied via PRAGMA journal_mode = ... on every new connection. Settable as ARCHIVEBOX_SQLITE_JOURNAL_MODE.

The default is WAL (Write-Ahead Logging) on native installs and DELETE in Docker. WAL lets readers and a single writer operate concurrently without blocking each other, but its shared-memory locking is unsafe when a Docker bind mount exposes the same live database to host-side SQLite processes. Docker therefore rejects an explicit WAL override for the SQLite backend; use PostgreSQL for safe cross-runtime concurrency.

> [!WARNING] Do not change this unless you have a specific reason. DELETE and TRUNCATE serialize readers against writers. MEMORY and OFF disable durable journaling and can corrupt the database on crash or power loss. WAL requires one local filesystem and one locking domain; it is unsafe across network filesystems and Docker host bind mounts.

SQLITE_MMAP_SIZE

Possible Values: [134217728] (128 MiB) on bare-metal, [0] (disabled) inside Docker / 0 / 268435456 / ... Maximum number of bytes of the database file SQLite is allowed to map into memory via mmap(), applied via PRAGMA mmap_size = ... Settable as ARCHIVEBOX_SQLITE_MMAP_SIZE.

When mmap is enabled, SQLite reads pages directly from the OS page cache instead of issuing read() syscalls and copying into a userspace buffer — meaningfully faster page reads on large databases when there is RAM available to cache them. Setting this to 0 disables memory-mapped I/O entirely and falls back to regular read() calls.

Note: The default is 0 (disabled) inside Docker, because the container's reported memory limits often do not reflect the host page cache and large mmap regions can interact poorly with cgroup accounting. On bare-metal installs the default is 134217728 (128 MiB).

SQLITE_BUSY_TIMEOUT

Possible Values: [30000]/5000/60000/... (milliseconds, integer) SQLite busy-wait timeout in **milliseconds**, applied via PRAGMA busy_timeout = ... on every new connection and converted to seconds for the Python sqlite3 connection-level timeout= argument. Settable as ARCHIVEBOX_SQLITE_BUSY_TIMEOUT.

When a statement encounters a write lock, SQLite will sleep and retry internally for up to this many milliseconds before returning SQLITE_BUSY to the Python driver. Raise it if you see spurious lock errors under sustained write contention and you would rather block than fail; lower it if you want callers to fail fast.

SQLITE_LOCK_RETRY_TIMEOUT

Possible Values: [60.0]/0/120.0/... (seconds, float) Total wall-clock budget in **seconds** that ArchiveBox's own retry loop will spend re-attempting a single locked statement before aborting it. Settable as ARCHIVEBOX_SQLITE_LOCK_RETRY_TIMEOUT.

When the SQLite driver eventually surfaces a database is locked error (after *SQLITE_BUSY_TIMEOUT* has already elapsed), ArchiveBox wraps the cursor in a higher-level retry loop that logs the locking holders and re-issues the statement. This is the maximum total time spent in that outer loop, across all retries, before giving up and raising. Set to 0 to disable the cap and retry indefinitely.

> [!NOTE] The outer retry only applies to statements that are *not* inside an explicit transaction.atomic() block. Statements inside an explicit transaction propagate the error to the caller immediately, since silently retrying would re-execute statements the caller already considered committed.

Related options: SQLITE_LOCK_RETRY_INTERVAL

SQLITE_LOCK_RETRY_INTERVAL

Possible Values: [5.0]/1.0/10.0/... (seconds, float, must be > 0) Sleep duration in **seconds** between successive attempts inside the ArchiveBox lock-retry loop. Settable as ARCHIVEBOX_SQLITE_LOCK_RETRY_INTERVAL.

Lower values retry more aggressively (useful if you expect locks to clear quickly and want to minimize end-to-end latency); higher values reduce log noise and wasted CPU when locks are typically held for a long time. Must be strictly greater than 0.

Related options: SQLITE_LOCK_RETRY_TIMEOUT

Search Settings

Options for full-text search backend configuration.

ArchiveBox can index Snapshot text/HTML output into searchable indexes that power the search bar in the Web UI and the archivebox search <query> CLI command. Multiple backend engines can be enabled at once; SEARCH_BACKEND_ENGINE selects the default used by the UI and CLI.

> [!NOTE] Each backend has its own tuning knobs (e.g. Sonic host/port, ripgrep flags, SQLite FTS database path). Those backend-specific options now live with the plugin that implements them — see the abx-plugins docs for the full per-backend schema.

SEARCH_BACKEND_ENGINE

Possible Values: [ripgrep]/sqlite/sonic

Which search backend engine to use when running `archivebox search` and rendering the Web UI search bar.

- **`ripgrep`** — Pure filesystem grep across each Snapshot's archived output (HTML, text, metadata) via the `search_backend_ripgrep` plugin. No extra daemon, no extra database to maintain — just install `rg` and it works. Slow on very large collections (each query re-scans the disk) but always 100% correct: results reflect what's actually on disk *right now*, no stale index. ArchiveBox keeps it enabled as the fallback when Sonic is unavailable.
- **`sonic`** *(default)* — Fast, suggest-style fuzzy search via a running Sonic daemon (configured via the `search_backend_sonic` plugin). ArchiveBox pushes text into Sonic at index time and queries it at search time. Sub-millisecond queries even at very large scale; ArchiveBox starts the managed service automatically when this backend is selected.
- **`sqlite`** — FTS5 full-text index stored alongside ArchiveBox's main `index.sqlite3`, configured via the `search_backend_sqlite` plugin. No extra processes, no extra binary — uses the SQLite already shipped with Python. Faster than `ripgrep` on large collections, slightly slower than `sonic`, but no daemon to babysit. Good middle ground for users who want a real index without operational overhead.

Note: Backend-specific tuning (Sonic host/port/password, ripgrep flag overrides, SQLite FTS database path, indexer batch size, etc.) lives in each search-backend plugin's own config schema — see the abx-plugins docs for the full per-backend option list.

Shell Options

Options around the format & behavior of CLI output.

Most of the values in this section are auto-detected from your terminal at startup, but each can be overridden explicitly via env var, `ArchiveBox.conf`, or `archivebox config --set` — useful for CI logs, cron jobs, log files, and Docker stdout where the auto-detection isn't what you want.

DEBUG

Possible Values: [False]/True

Enable verbose debug mode for the entire ArchiveBox process. Automatically set to `True` when `--debug` is passed on the command line; otherwise honors the env var / config value.

When enabled this turns on:

- Full Python tracebacks (instead of the trimmed friendly version) on any error
- Django SQL query logging to stderr
- Template auto-reload (no caching) for the web UI
- Verbose plugin / hook lifecycle logging
- Extra detail in `archivebox version`, `archivebox status`, and crash reports

  [!WARNING] **Do not leave `DEBUG=True` enabled on a production / publicly-reachable server.** It exposes tracebacks with file paths, SQL queries, and environment details that can leak sensitive info to anyone who triggers an error page.

Related options: USE_COLOR, SHOW_PROGRESS

USE_COLOR

Possible Values: [`True` *(auto-detected)*]/`False`

Whether to colorize console output with ANSI escape codes. Defaults to `True` when stdout is a TTY (interactive terminal) and `False` otherwise.

Override to **force-off** when piping `archivebox` output into a log file or cron-mail wrapper that doesn't strip ANSI codes (otherwise you'll see `^[[31m...^[[0m` litter throughout your logs). Override to **force-on** for tools like `script(1)` or some CI runners that don't report as a TTY but *do* render ANSI correctly.

```
USE_COLOR=False archivebox add https://example.com >> archive.log
```

Related options: *SHOW_PROGRESS*, *DEBUG*

SHOW_PROGRESS

Possible Values: [`True` *(auto-detected)*]/`False`

Whether to render live progress bars during long-running operations (archiving, indexing, migrations). Defaults to `True` when stdout is a TTY, `False` otherwise.

Override to **force-off** in environments where the auto-detection is fooled into thinking it has a TTY (some Docker setups, Kubernetes log collectors, `tmux`/`screen` pipes) but the redrawing carriage-return output ends up as garbage in your logs.

```
SHOW_PROGRESS=False archivebox add < urls.txt
```

Related options: *USE_COLOR*

Plugin Configuration

> [!IMPORTANT] **Per-plugin configuration has moved to its own documentation site.** This `Configuration.md` doc covers only ArchiveBox's *core* settings. For everything that lives inside a plugin — extractor toggles, binary paths, timeouts, args, user agents, cookies, persona scoping, etc. — see:
>
> **https://plugins.archivebox.io/**

That site is regenerated from each plugin's `config.json` schema on every release, so it stays in sync with the code. Looking for `WGET_ARGS`, `CHROME_USER_DATA_DIR`, `SCREENSHOT_RESOLUTION`, `YTDLP_EXTRA_ARGS`, `SINGLEFILE_*`, `SONIC_HOST`, etc.? They all live there now.

Shared core options that plugins fall back to

A handful of *core* options (documented above on this page) act as the **fallback default** for every plugin that has a matching per-extractor override. If you set the core option, every plugin honors it; if you also set the plugin-specific override, that wins for just that one plugin.

Core option (this doc)	Plugin-level overrides (see abx-plugins)
TIMEOUT	WGET_TIMEOUT, CHROME_TIMEOUT, YTDLP_TIMEOUT, SINGLEFILE_TIMEOUT, TITLE_TIMEOUT, FAVICON_TIMEOUT, …
CHECK_SSL_VALI	WGET_CHECK_SSL_VALIDITY, YTDLP_CHECK_SSL_VALIDITY, GALLERYDL_CHECK_SSL_VALIDITY, CHROME_CHECK_SSL_VALIDITY, …
USER_AGENT	WGET_USER_AGENT, CHROME_USER_AGENT, …
COOKIES_FILE	WGET_COOKIES_FILE, YTDLP_COOKIES_FILE, GALLERYDL_COOKIES_FILE, …
RESOLUTION	SCREENSHOT_RESOLUTION, PDF_RESOLUTION, CHROME_RESOLUTION

[!TIP] The resolution order for any plugin-tunable option is always: **1.** <PLUGIN>_<OPTION> (explicit per-plugin override) → **2.** the matching shared core option above → **3.** the plugin's own hardcoded default.

So setting `TIMEOUT=120` once at the top of your `ArchiveBox.conf` raises the timeout for *every* extractor at once; setting `CHROME_TIMEOUT=300` on top of that lifts it further for just Chrome.

Listing & setting plugin options

All plugin options can be set via the same three mechanisms as core options — env var, `ArchiveBox.conf`, or `archivebox config --set` — and inspected with `archivebox config`:

```
archivebox config                                  # show every option (core + every␣
↪installed plugin)
archivebox config --get SCREENSHOT_RESOLUTION      # read one value
archivebox config --set SCREENSHOT_RESOLUTION=1920,1080
archivebox config --search wget                    # search options by name/description
```

Why is plugin config documented separately?

Plugin schemas evolve on their own release cadence — new extractors ship between ArchiveBox releases, options are added/renamed as hooks revise, and pinning their docs to the core release schedule produced unavoidable drift. The per-plugin doc is auto-generated from each plugin's `config.json` schema at build time, so it never lags behind the code.

1.3.5 Security Overview

Web UI Permissions

```
archivebox config --set PUBLIC_INDEX=False      # require login to access the list of␣
↪Snapshots
archivebox config --set PUBLIC_ADD_VIEW=False   # require log-in to submit new URLs for␣
↪archiving
archivebox config --set PERMISSIONS=private     # default new snapshots to login-
↪required (was: PUBLIC_SNAPSHOTS=False)

archivebox manage [createsuperuser|changepassword] # create/modify admin UI users
```

See *Setting Up Authentication* for more…

ArchiveBox Use-Cases

Archiving Public Content Only `[Default, recommended for most people]`

This is the default (lax) mode, intended for archiving public (non-secret) URLs without authenticating the headless browser. This is the mode used if you're archiving news articles, audio, video, etc. browser bookmarks to a folder published on your webserver. This allows you to access and link to content on `http://your.archive.com/archive...` after the originals go down.

The default mode should not be used for archiving entire browser history or authenticated private content like Google Docs, paywalled content, invite-only subreddits, private photo share urls, etc.

```
# (these are the defaults)
archivebox config --set ARCHIVEDOTORG_ENABLED=True   # see https://plugins.archivebox.io/
↪#archivedotorg
archivebox persona create public
archivebox add --persona=public 'https://example.com'
```

Archiving Content Behind Log-Ins `[Advanced users only]`

ArchiveBox is able to archive content that requires authentication or cookies, but it comes with some caveats. Create dedicated logins for archiving to access paywalled content, private forums, LAN-only content, etc. then share them with ArchiveBox via Chrome profile + cookies.txt file.

```
archivebox config --set ARCHIVEDOTORG_ENABLED=False
archivebox persona create --import=chrome personal
archivebox add --persona=personal 'https://members.example.com/'
```

To get started, import a dedicated browser profile into a *persona*. A persona keeps its Chrome profile and `cookies.txt` together and applies the same identity consistently across extractors.

For full instructions on setting up a Chromium user profile see here: https://github.com/ArchiveBox/ArchiveBox/wiki/Chromium-Install#setting-up-a-chromium-user-profile

If you're importing private links or authenticated content, you probably don't want to share your archive folder publicly on a webserver, so don't follow the *Publishing Your Archive* instructions unless you are only serving it on a trusted LAN or have some sort of authentication in front of it. Make sure to point ArchiveBox to an output folder with conservative permissions, as it may contain archived content with secret session tokens or pieces of your user data. You may also wish to encrypt the archive using an encrypted disk image or filesystem like ZFS as it will contain all requests and response data, including session keys, user data, usernames, etc.

Things to watch out for:

- any cookies / secret state present in a Chrome user profile or `cookies.txt` file may be reflected in server responses and saved in the Snapshot output (e.g. in `headers` extractor output) — visible in cleartext to anyone viewing the Snapshot. **Don't use your personal Chrome profile for archiving** or people viewing your archive can then authenticate as you.
- any secret tokens embedded in URLs (e.g. secret invite links, Google Doc URLs, etc.) will be visible on `archive.org` as the URLs are not filtered when saving to it. Disable submitting to Archive.org entirely with `ARCHIVEDOTORG_ENABLED=False`.
- the domain portion in archived URLs is sent to a favicon service in order to retrieve an icon more reliably than a janky internal implementation would be able to (if leaking domains is a concern, you can change the `FAVICON_PROVIDER` or disable favicon fetching entirely with `FAVICON_ENABLED=False`).
- viewing malicious archived JS could allow an attacker to access your other archive items + the admin interface — use the default *`SERVER_SECURITY_MODE=auto`* (`*.localhost` uses isolated replay subdomains and other

hosts use one-domain/no-JS raw replay), or disable risky extractors entirely with `WGET_ENABLED=False` and `DOM_ENABLED=False`. Plugins that need their own trusted viewer can opt in with an explicit `full.html` preview template; ArchiveBox applies the no-JS policy to all other raw archived HTML without maintaining a plugin whitelist.

An example of a session cookie reflected in `headers.json` *visible in the archive.*

Publishing

[!CAUTION] Re-hosting untrusted archived content on the same origin as an authenticated application can compromise that application.

Make sure you understand the dangers of hosting untrusted HTML/JS/CSS. The default `SERVER_SECURITY_MODE=auto` uses isolated subdomains with full replay on `*.localhost`, and a one-domain no-JS replay policy on ordinary public or LAN hostnames. Choose `safe-subdomains-fullreplay` only when wildcard DNS and TLS are configured; it separates admin, web, and API control-plane origins from replay content and gives each Snapshot its own replay subdomain.

Do not serve ArchiveBox from a shared subdirectory such as `myapps.example.com/archivebox/`; it cannot provide the required origin isolation.

Published archives automatically include a `robots.txt` `Disallow: /` to block search engines from indexing them. You may still wish to publish your contact info in the index footer though using *`FOOTER_INFO`* so that you can respond to any DMCA and copyright takedown notices if you accidentally rehost copyrighted content.

Make sure to read all the warnings *above* about the dangers of exposing Chrome profile data, cookies, secret tokens in URLs, and the risks of viewing archived JS on a shared origin before publishing your archive.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Publishing-Your-Archive
- https://github.com/ArchiveBox/ArchiveBox/wiki/Publishing-Your-Archive#security-concerns
- https://github.com/ArchiveBox/ArchiveBox/wiki/Publishing-Your-Archive#copyright-concerns
- https://en.wikipedia.org/wiki/Cross-site_request_forgery
- https://github.com/ArchiveBox/ArchiveBox/issues/239

Run ArchiveBox as an unprivileged user

[!NOTE] Use `docker compose run --rm archivebox ...` for normal one-shot CLI commands. `docker compose exec archivebox ...` is also supported against a running container; if it starts as root, ArchiveBox drops to the dedicated `archivebox` account before operating on the collection.

ArchiveBox drops privileges to the collection owner when it starts as root and can do so safely, including in the official Docker image. Do not bypass that boundary or force runtime dependencies to stay privileged:

- Browser sandboxing cannot provide its normal protection when the browser itself runs as root
- All dependencies will be run as root, if any of them have a vulnerability that's exploited by sites you're archiving you're opening yourself up to full system compromise
- ArchiveBox does lots of HTML parsing, filesystem access, and shell command execution. A bug in any one of those subsystems could potentially lead to deleted/damaged data on your hard drive, or full system compromise unless restricted to a user that only has permissions to access the directories needed

- Do you really trust a project created by a Github user called `@pirate` ? Why give a random program off the internet root access to your entire system? (I don't have malicious intent, I'm just saying in principle you should not be running random Github projects as root)

ArchiveBox creates and drops privileges to a dedicated `archivebox` account when run as root on Linux. Existing non-root users can run it directly:

```
mkdir -p ~/archivebox/data
cd ~/archivebox/data
archivebox init
archivebox install
```

Output Folder

Database

The ArchiveBox database is an unencrypted, uncompressed SQLite3 `index.sqlite3` file on disk, and such does not require an authenticated admin SQL login to access (like PostgreSQL/MySQL would). Make sure to protect your database file adequately as anyone who can read it can read your entire collection contents. Passwords for the admin users are stored as salted and PBKDF2 hashed strings in the `auth_user` table.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#disk-layout
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#database
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Setting-up-Authentication#username–password-the-default

Filesystem

How much are you planning to archive? Only a few bookmarked articles, or thousands of pages of browsing history a day? If it's only 1-50 pages a day, you can probably use a normal folder on your hard drive, but at higher volume you may want a compressed/deduplicated/encrypted filesystem like ZFS. Other distributed/networked/checksummed filesystems reported to work include SMB, NFS, Ceph, Unraid, and BTRFS. The database and config must remain on a local filesystem with reliable FSYNC. Current Snapshot directories are sharded under `archive/users/<user>/snapshots/<date>/<domain>/<uuid>/`, avoiding the old single-directory scaling limit.

Purging entries

`archivebox remove --yes URL` deletes matching Snapshot rows and schedules their Snapshot directories for cleanup through the normal state-machine path. The legacy `--delete` flag is accepted only for CLI compatibility and does not change that behavior. Original imports and operational history may still appear in `sources/`, `logs/`, or an external search backend; remove those separately if your goal is to erase every trace of a URL.

Permissions

Consider what permissioning to apply to your archive folder carefully. Limit access to the fewest possible users by checking folder ownership and setting *OUTPUT_PERMISSIONS* accordingly. Generally the `index.sqlite3` file, `archive/` folder, and `ArchiveBox.conf` file must all be owned and writable by the `archivebox` user or a dedicated non-root user.

When running with Docker, the entrypoint uses explicit *PUID/PGID* values when supplied, otherwise the existing non-root owner of the mounted data directory or the image's `archivebox` user.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#disk-layout
- https://github.com/ArchiveBox/ArchiveBox#output-formats
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#database
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#filesystem-doesnt-support-fsync-eg-network-mounts
- https://github.com/ArchiveBox/ArchiveBox#storage-requirements

1.3.6 Usage

Make sure the dependencies are fully installed before running any ArchiveBox commands.

ArchiveBox API Reference:

- *CLI Usage*: Docs and examples for the ArchiveBox command line interface.
- *Admin UI Usage*: Docs and screenshots for the outputted HTML archive interface.
- *Browser Extension Usage*: Docs and screenshots for the outputted HTML archive interface.
- *Disk Layout*: Description of the archive folder structure and contents.

Related:

- *Docker*: Learn about ArchiveBox usage with Docker and Docker Compose
- *Configuration*: Learn about the various archive method options
- *Scheduled Archiving*: Learn how to set up automatic daily archiving
- *Publishing Your Archive*: Learn how to host your archive for others to access
- *Troubleshooting*: Resources if you encounter any problems

CLI Usage

All three of these ways of running ArchiveBox are equivalent and interchangeable:

- `archivebox [subcommand] [...args]`
  Using the Python package via the current `uv` *instructions in* Install
- `docker run ... archivebox/archivebox [subcommand] [...args]`
  Using the official Docker image
- `docker compose run --rm archivebox [subcommand] [...args]` *Using the official Docker image w/ Docker Compose*

You can share a single archivebox data directory between Docker and non-Docker instances as well, allowing you to run the server in a container but still execute CLI commands on the host for example.

For more examples see README: Usage and *Docker* pages.

- *Run ArchiveBox with configuration options*
- *Import a single URL*
- *Import a list of URLs from a text file*
- *Import list of links from browser history*

Run ArchiveBox with configuration options

You can set environment variables in your shell profile, a config file, or by using the env command.

```
# set config via the CLI
archivebox config --set TIMEOUT=120

# OR edit ArchiveBox.conf and add this under its existing [ARCHIVING_CONFIG] section:
TIMEOUT=120

# OR use environment variables
env TIMEOUT=120 archivebox add 'https://example.com'
```

See *Configuration* page for core ArchiveBox config options and the abx-plugins config reference for per-plugin options (e.g. YTDLP_MAX_SIZE, CHROME_USER_DATA_DIR, WGET_ARGS, etc.). If you're using Docker, also make sure to read the Configuration section on the *Docker* page.

> [!TIP]
> You can run ArchiveBox commands from anywhere (without having to cd into a data directory first):
> /usr/bin/env --chdir=/path/to/archivebox/data archivebox update

---

Import a single URL

```
archivebox add 'https://example.com'
# OR
echo 'https://example.com' | archivebox add
```

You can also add --depth=1 to any of these commands if you want to recursively archive the URLs and all URLs one hop away. (e.g. all the outlinks on a page + the page).

Import a list of URLs from a text file

```
cat urls_to_archive.txt | archivebox add
# OR
archivebox add < urls_to_archive.txt
# OR
curl 'https://example.com/some/rss/feed.xml' | archivebox add
# OR
archivebox add --depth=1 'https://example.com/some/rss/feed.xml'
```

You can also pipe in RSS, XML, Netscape, or any of the other *supported import formats* via stdin.

```
archivebox add < ~/Downloads/browser_bookmarks_export.html
# OR
archivebox add < ~/Downloads/pinboard_bookmarks.json
# OR
archivebox add < ~/Downloads/any_text_containing_urls.txt
```

Import list of links from browser history

Look in the `bin/` folder of this repo to find a script to parse your browser's SQLite history database for URLs. Specify the type of the browser as the first argument, and optionally the path to the SQLite history file as the second argument.

```
bash ./bin/export_browser_history.sh --chrome
archivebox add < chrome_history.json
# or
bash ./bin/export_browser_history.sh --firefox
archivebox add < firefox_history.json
# or
bash ./bin/export_browser_history.sh --safari
archivebox add < safari_history.json
```

Import browser cookies into a persona

See *Chrome profile setup* for cookies, bookmarks, and host/Docker imports. Select the imported identity with `archivebox add --persona=personal URL`.

UI Usage

```
# configure which areas you want to require login to use vs make publicly available
archivebox config --set PUBLIC_INDEX=False
archivebox config --set PUBLIC_ADD_VIEW=False
archivebox config --set PERMISSIONS=private        # default visibility of newly created␣
↪snapshots (was: PUBLIC_SNAPSHOTS=False)

archivebox server 0.0.0.0:5797     # start the archivebox web server
```

Open http://admin.archivebox.localhost:5797/admin/ in a browser to create the first admin and finish web setup. To create additional accounts from the CLI, use `archivebox manage createsuperuser`.

See the Configuration Wiki *and* Security Wiki *for more info...*

Or if you prefer to generate a static HTML index instead of using the built-in web server, you can run `archivebox list --html --with-headers > ./index.html` and then open `./index.html` in a browser. You should see something like this.

You can sort by column, search using the box in the upper right, and see the total number of links at the bottom.

Click the Favicon under the "Files" column to go to the details page for each link.

Explanation of buttons in the web UI - admin snapshots list

A logged-in admin user may select one or more snapshots from the list and perform Snapshot actions:

- Search Search text in the Snapshot title, URL, tags, or archived content. Sonic is the default indexed search backend, with ripgrep enabled as the filesystem fallback and regex-search option. https://github.com/ArchiveBox/ArchiveBox/issues/956
- Tags Start typing in the field to select some tags, then click + to add them or - remove them from the checked snapshots (`Tags` can be created/edited from the `/admin/core/tag/` page)

- Title Pull the latest title and favicon without doing a full snapshot. (helpful to quickly ping any URLs that are stuck showing up as `Pending...` or are missing a title)
- Pull Finish downloading the Snapshot, pulls any missing/failed outputs/extractors methods (pdf, wget... etc). Resumes running the same archiving steps as when you add new URL. Useful to finish pulling when previous import was paused or interrupted by a reboot or something. https://github.com/ArchiveBox/ArchiveBox#output-formats
- Re-Snapshot Re-archive the original URL from scratch as a new separate snapshot. Differs from pulling in that it doesn't resume/update existing snapshot, it creates a new separate entry and re-snapshots the URL at the current point in time. (useful for saving multiple Snapshots of a single URL over time) https://github.com/ArchiveBox/ArchiveBox#saving-multiple-snapshots-of-a-single-url
- Reset Keep the Snapshot entry, but delete all its archive results and redownload them from scratch immediately. Useful for re-trying a bad Snapshot and overwriting its previous results, e.g. if it initially archived a temporary error page or hit a transient rate-limit/CAPTCHA/login page.
- Delete Delete a snapshot and all its archive results entirely. This action cannot be undone. (Note: to thoroughly remove every trace of a URL ever being added, you should also manually scrub log output found in `sources/` and `logs/`)

Browser Extension Usage

Set up the official ArchiveBox Browser Extension to submit URLs directly from your browser to ArchiveBox.

1. Install the extension in your browser:
   - Google Chrome / Edge / All Chromium-based browsers...
   - Firefox
2. Log into your ArchiveBox server's admin UI in the same browser where you installed the extension, e.g. `http://localhost:5797/admin/` or `https://demo.archivebox.io/admin/` The extension will re-use your admin UI login session to submit URLs to your server, so *make sure to log in!*
   ...
   Alternatively: You can configure Archivebox to allow submitting URLs without requiring log-in
   `archivebox config --set PUBLIC_ADD_VIEW=True`
3. Click the ArchiveBox extension in your browser and set `Config > ArchiveBox Base URL` to your server's URL, e.g.
   `http://localhost:5797` or `https://demo.archivebox.io`
4. Done! Test it out: `Right-click on any page > ArchiveBox Exporter > Archive Current Page`
   Then check your ArchiveBox instance to confirm the URL was added.

More Info

- https://github.com/ArchiveBox/archivebox-browser-extension
- https://github.com/ArchiveBox/archivebox-browser-extension#setup
- https://github.com/ArchiveBox/archivebox-browser-extension#features
- https://github.com/ArchiveBox/archivebox-browser-extension#alternative-extensions-for-archiving
- https://github.com/ArchiveBox/ArchiveBox/issues/577

Disk Layout

The OUTPUT_DIR folder (usually whatever folder you run the archivebox command in), contains the UI HTML and archived data with the structure outlined below.

Simply back up the entire data/ folder to back up your archive, e.g. zip -r data.backup.zip data.

```
- data/
  - index.sqlite3        # Main index of all archived URLs
  - ArchiveBox.conf      # Main config file in ini format

  - archive/
     - 155243135 -> users/admin/snapshots/20210406/example.com/SNAPSHOT_UUID/
     - users/
        - admin/
           - snapshots/
              - 20210406/
                 - example.com/
                    - SNAPSHOT_UUID/
                       - index.jsonl
                       - index.html

                       # Archive method outputs:
                       - wget/warc/
                       - ytdlp/media/
                       - git/
                       ...

  - sources/             # Each imported URL list is saved as a copy here
     - getpocket.com-1552432264.txt
     - stdin-1552291774.txt
     ...
```

For more info about ArchiveBox's database/filesystem layout and troubleshooting steps:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#database

Large Archives

I've found it takes about an hour to download 1000 articles, and they'll take up roughly 1GB.
Those numbers are from running it single-threaded on my i5 machine with 50mbps down. YMMV.

Storage requirements go up immensely if you're using MEDIA_ENABLED=True (or its FETCH_MEDIA / YTDLP_ENABLED aliases) and are archiving many pages with audio & video.

ArchiveBox's unified crawl runner handles bounded concurrency without starting competing writers:

```
env CRAWL_MAX_CONCURRENT_SNAPSHOTS=4 archivebox add < urls_to_archive.txt
```

Higher concurrency is not always faster on slow disks or network filesystems, so increase it gradually.

Users have reported running it with 50k+ bookmarks with success (though it will take more RAM while running).

If you already imported a huge list of bookmarks and want to import only new bookmarks, you can use the ONLY_NEW environment variable. This is useful if you want to import a bookmark dump periodically and want to skip broken links which are already in the index.

For more info about troubleshooting filesystem permissions, performance, or issues when running on a NAS:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#database

---

SQL Shell Usage

Explore the SQLite3 DB a bit to see what's available using the SQLite3 shell:

```
cd ~/archivebox/data
sqlite3 index.sqlite3

# example usage:
SELECT * FROM core_snapshot;
UPDATE auth_user SET email = 'someNewEmail@example.com' WHERE username =
↪'someUsernameHere';
...
```

More info:

- https://github.com/ArchiveBox/ArchiveBox#usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#database
- https://stackoverflow.com/questions/1074212/how-can-i-see-the-raw-sql-queries-django-is-running
- https://adamobeng.com/wddbfs-mount-a-sqlite-database-as-a-filesystem/

Python Shell Usage

Explore the Python API a bit to see what's available using the archivebox shell:

Python API Documentation: https://docs.archivebox.io/dev/apidocs/index.html

```
$ archivebox shell
[i] ArchiveBox shell
>>> from archivebox.core.models import Snapshot
>>> from archivebox.cli.archivebox_add import add

# count completed snapshots
>>> print(Snapshot.objects.filter(status=Snapshot.StatusChoices.SEALED).count())
24

# inspect or add URLs through current APIs
>>> Snapshot.objects.filter(url="https://example.com").first()
<Snapshot: https://example.com>
>>> crawl, snapshots = add(urls=["https://example.com/new"], index_only=True)
```

(continues on next page)

(continued from previous page)

```
# show raw SQL queries run
>>> from django.db import connection
>>> print(connection.queries)
```

For more info and example usage:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Setting-up-Authentication#username–password-the-default
- https://github.com/ArchiveBox/ArchiveBox/blob/dev/archivebox/cli/
- https://github.com/ArchiveBox/ArchiveBox/blob/dev/archivebox/config/common.py
- https://github.com/ArchiveBox/ArchiveBox/blob/dev/archivebox/core/models.py
- https://stackoverflow.com/questions/1074212/how-can-i-see-the-raw-sql-queries-django-is-running

---

Python API Usage

You can interact with ArchiveBox as a Python library from external scripts or programs.

This API is a *local* API, designed to be used on the same machine as the ArchiveBox collection.

For example, you could create a script `add_archivebox_url.py` like so:

```
import os
from pathlib import Path

DATA_DIR = Path("~/archivebox/data").expanduser()
os.chdir(DATA_DIR)

# you must import and setup django first to establish a DB connection
from archivebox.config.django import setup_django
setup_django(check_db=True)

# then import the specific API you need
from archivebox.cli.archivebox_add import add

crawl, snapshots = add(urls=["https://example.com"], index_only=True)
print(crawl.id, list(snapshots.values_list("id", flat=True)))
```

For more information see:

- ArchiveBox Python API Reference (ReadTheDocs)
- ArchiveBox Developer Documentation
- ArchiveBox Python source code

1.4 Guides

1.4.1 Setting Up Storage

ArchiveBox supports a wide range of local and remote filesystems using `rclone` and/or Docker storage plugins. The examples below use Docker Compose bind mounts to demonstrate the concepts; adapt the host paths, ownership, and provider settings to your environment.

Example `docker-compose.yml` storage setup:

```
services:
    archivebox:
        # ...other service settings...
        volumes:
            # your index db, config, logs, etc. should be stored on a local SSD (usually
↪<10Gb)
            - ./data:/data

            # but bulk archive/ content can be located on an HDD or remote filesystem
            - /mnt/archivebox-archive:/data/archive
```

Supported Local Filesystems

EXT4 (default on Linux), APFS (default on macOS)

[!TIP] These default filesystems are fully supported by ArchiveBox on Linux and macOS (w/wo Docker).

ZFS (recommended for experienced Linux/BSD operators)

[!TIP] *ZFS is a good choice when you already operate OpenZFS and want checksumming, compression, snapshots, replication, and optional encryption or disk redundancy.* On Ubuntu, follow the official OpenZFS installation guide. macOS and BSD installation and property support differ, so use the guide for your OS.

- https://openzfs.github.io/openzfs-docs/
- https://openzfs.github.io/openzfs-docs/man/v2.2/8/zpool-create.8.html
- https://openzfs.github.io/openzfs-docs/man/v2.2/8/zfs-create.8.html
- https://docs.docker.com/storage/storagedriver/zfs-driver/
- https://www.ixsystems.com/blog/fast-dedup-is-a-valentines-gift-to-the-openzfs-and-truenas-communities/

[!CAUTION] Creating a pool erases the selected disks. The two-disk example below creates a mirror; replace both `/dev/disk/by-id/...` placeholders with the persistent IDs of empty disks you intend to erase.

```
# Create a mirrored pool without forcing ZFS's safety checks.
sudo zpool create \
    -O mountpoint=none \
    -O compression=lz4 \
    -O dnodesize=auto \
    -O atime=off \
    -O xattr=sa \
    -O acltype=posixacl \
    -O aclinherit=passthrough \
    archivebox mirror \
    /dev/disk/by-id/DISK_ONE \
    /dev/disk/by-id/DISK_TWO
```

(continues on next page)

(continued from previous page)

```
# Create the unencrypted ArchiveBox data dataset.
sudo zfs create \
    -o mountpoint=/mnt/archivebox/data \
    archivebox/data
```

To encrypt a new dataset, use this command **instead of** the unencrypted `zfs create` command above. ZFS encryption must be selected when the dataset is created.

```
sudo zfs create \
    -o mountpoint=/mnt/archivebox/data \
    -o encryption=on \
    -o keyformat=passphrase \
    -o keylocation=prompt \
    archivebox/data
```

NTFS, HFS+, BTRFS

[!WARNING] These filesystems are likely supported, but are not officially tested.

EXT2, EXT3, FAT32, exFAT

[!CAUTION] Not recommended. Cannot store files >4GB or more than 31k ~ 65k Snapshot entries due to directory entry limits.

Supported Remote Filesystems

ArchiveBox supports many common types of remote filesystems using Rclone, FUSE, Docker storage providers, and Docker volume plugins.

The `data/archive/` subfolder contains the bulk archived content, and it supports being stored on a slower remote server (SMB/NFS/SFTP/etc.) or object store (S3/B2/R2/etc.). For data integrity and performance reasons, the rest of the `data/` directory (`data/ArchiveBox.conf`, `data/logs`, etc.) must be stored locally while ArchiveBox is running.

[!IMPORTANT] `data/index.sqlite3` is your main archive DB, *it must be on a fast, reliable, local filesystem* which supports FSYNC (SSD/NVMe recommended for best experience).

[!TIP] If you use a remote filesystem, you should switch ArchiveBox's search backend from `ripgrep` to `sonic` (or `FTS5`).
(`ripgrep` scans over every byte in the archive to do each search, which is **slow and potentially costly** on remote cloud storage)

NFS (Docker Driver)

Configure the export's ownership or ACL so a non-root numeric identity can create and remove files, then set Docker `PUID/PGID` to that identity if it differs from the local collection owner. NFS `root_squash` is compatible; ArchiveBox does not rely on container root to write the export.

`docker-compose.yml`:

```
services:
    archivebox:
        volumes:
            - ./data:/data
            - archivebox-archive:/data/archive

volumes:
    archivebox-archive:
        driver: local
        driver_opts:
            type: "nfs"
            o: "addr=some-remote-server.example.com,rw,nfsvers=4"
            device: ":/archivebox-archive"
```

SMB / Ceph (Docker CIFS Driver)

The example below presents the mount as the image default `911:911`. If you choose another forced `uid/gid`, set the ArchiveBox service's `PUID/PGID` to the same values.

`docker-compose.yml`:

```
services:
    archivebox:
        volumes:
            - ./data:/data
            - archivebox-archive:/data/archive

volumes:
    archivebox-archive:
        driver: local
        driver_opts:
            type: cifs
            device: "//some-remote-server.example.com/archivebox-archive"
            o: "username=XXX,password=YYY,uid=911,gid=911"
```

Amazon S3 / Backblaze B2 / Google Drive / etc. (RClone)

ArchiveBox stores snapshot content under `data/archive/users/<user>/snapshots/<date>/<domain>/<uuid>/` and keeps backwards-compatible `data/archive/<timestamp>` symlinks. Object-storage mounts must enable Rclone's VFS symlink translation so both parts of this layout work.

Install the `rclone` binary through `abxpkg`:

```
uv tool install abxpkg
abxpkg install rclone
abxpkg run rclone version
```

Then install the FUSE 3 system integration supplied by your OS. For example, on Ubuntu:

```
sudo apt-get install fuse3
grep -qxF user_allow_other /etc/fuse.conf ||
    printf '%s\n' user_allow_other | sudo tee -a /etc/fuse.conf
```

Then define your remote storage config `~/.config/rclone/rclone.conf`:

> [!TIP] You can also create `rclone.conf` using the Rclone Web GUI: `abxpkg run rclone rcd --rc-web-gui`

```
# Example rclone.conf using Amazon S3 for storage:
[archivebox-s3]
type = s3
provider = AWS
access_key_id = XXX
secret_access_key = YYY
region = us-east-1
```

Rclone Config Examples

- SMB / Ceph / SFTP / FTP / WebDAV (e.g. Nextcloud)
- Google Drive / Dropbox / OneDrive
- Amazon S3 / Backblaze B2 / Cloudflare R2 / DigitalOcean Spaces
- Google Cloud Storage / Azure Blob / Azure Files
- Storj / Sia / Archive.org Storage
- And many more…
  - https://rclone.org/s3/
  - https://rclone.org/overview/

Bonus:

- Set up gzip compression: https://rclone.org/compress/
- Set up file encryption: https://rclone.org/crypt/
- Set up hashing engine: https://rclone.org/hasher/

Option A: Running Rclone on a bare-metal host

1. *If Needed:* Transfer any existing local archive data to the remote volume first

   > [!CAUTION] Stop ArchiveBox before migrating its archive directory. `rclone sync` makes the remote destination match the local source and can delete files already present at the destination. Run it with `--dry-run` first, make a separate backup, and do not move the local copy until `rclone check` succeeds.

```
abxpkg run rclone sync \
    --dry-run \
    --links \
    --fast-list \
    --transfers 20 \
    --progress \
    /opt/archivebox/data/archive/ \
    archivebox-s3:data/archive/
```

(continues on next page)

(continued from previous page)

```
# Remove --dry-run only after reviewing the proposed changes, then verify them.
abxpkg run rclone sync \
    --links \
    --fast-list \
    --transfers 20 \
    --progress \
    /opt/archivebox/data/archive/ \
    archivebox-s3:data/archive/
abxpkg run rclone check --links /opt/archivebox/data/archive/ archivebox-s3:data/archive/

mv /opt/archivebox/data/archive /opt/archivebox/data/archive.localbackup
mkdir -p /opt/archivebox/data/archive
```

2. **Mount the remote storage volume as FUSE filesystem**

Run the mount as the numeric user that owns the local ArchiveBox collection. The command stays in the foreground so a service manager can supervise it.

```
abxpkg run rclone mount \
    archivebox-s3:data/archive/ \
    /opt/archivebox/data/archive/ \
    --allow-other \
    --vfs-cache-mode=full \
    --vfs-links \
    --transfers=16 \
    --checkers=4
```

See Rclone's `rclone mount` documentation for service-manager and cache-size configuration.

> [!TIP] You can use an existing Rclone FUSE mount as a normal Docker bind mount. A separate storage plugin is usually unnecessary when `user_allow_other` and `--allow-other` are configured correctly.

```
docker run --rm \
    -v "$PWD:/data" \
    -v /opt/archivebox/data/archive:/data/archive \
    archivebox/archivebox:dev status
```

`docker-compose.yml`:

```
services:
    archivebox:
        # ...other service settings...
        volumes:
            - ./data:/data
            - /opt/archivebox/data/archive:/data/archive
```

Option B: Running Rclone with the Docker storage plugin

This Linux Docker Engine option is only needed if you cannot use Option A for compatibility or performance reasons, or if you prefer defining your remote storage in `docker-compose.yml`.

See here for full instructions: Rclone Documentation: Docker Plugin

1. First, install the Rclone Docker Volume Plugin for your CPU architecture (e.g. `amd64` or `arm64`):

```
sudo mkdir -p \
    /var/lib/docker-plugins/rclone/config \
    /var/lib/docker-plugins/rclone/cache
sudo install -m 600 \
    ~/.config/rclone/rclone.conf \
    /var/lib/docker-plugins/rclone/config/rclone.conf

# Replace amd64 with arm64 on ARM hosts.
docker plugin install rclone/docker-volume-rclone:amd64 --grant-all-permissions --alias␣
↪rclone
```

2. Then, create a volume using the Docker CLI or define one using Docker Compose / Swarm:

`docker-compose.yml`:

```
services:
    archivebox:
        volumes:
            - ./data:/data
            - archivebox-s3:/data/archive

volumes:
    archivebox-s3:
        driver: rclone
        driver_opts:
            remote: 'archivebox-s3:data/archive'
            allow_other: 'true'
            vfs_cache_mode: full
            vfs_links: 'true'
            # Match these and the ArchiveBox PUID/PGID to the effective mount owner;
            # 911:911 is the image default.
            uid: 911
            gid: 911
            transfers: 16
            checkers: 4
```

To start the container and verify the filesystem is accessible within it:

```
docker compose run --rm archivebox \
    /bin/bash -c 'touch /data/archive/.write_test && rm /data/archive/.write_test'
```

More Docker Storage Plugins

- IPFS / Peergos / GlusterFS
- DigitalOcean Block Storage Volumes / Linode Block Storage Volumes
- More volume plugins…

1.4.2 Setting Up Authentication

ArchiveBox supports several types of authentication for users logging in via the Admin Web UI or REST API.

Set Up Admin Web UI Permissions

Use these options to set up your desired permissions for non-admin guest users:

- `PUBLIC_INDEX=True`: Default *allows* non-logged-in users to see Snapshot list
- `PUBLIC_ADD_VIEW=False`: Default *doesn't allow* non-logged-in users to submit new URLs
- *`PERMISSIONS=public`*: Default *allows* non-logged-in users to see Snapshot content (set to `unlisted` or `private` to gate it; replaces the removed legacy `PUBLIC_SNAPSHOTS` toggle, which was a global on/off — `PERMISSIONS` is now per-Snapshot)

  [!NOTE] ArchiveBox does not currently support setting up *non-admin* users and groups with custom permissions.
- *Wiki: Configuration* (`PUBLIC_INDEX`, `PUBLIC_ADD_VIEW`, `PERMISSIONS`)
- *Wiki: Security Overview*

Admin Web UI Authentication Methods

Username & Password (the default)

On a new server, open http://admin.archivebox.localhost:5797/admin/ to create the first admin through the setup UI. Existing users can be created or edited from the CLI:

```
archivebox manage createsuperuser
archivebox manage changepassword <username>

# equivalent: docker compose run --rm archivebox manage [...]
# equivalent: docker run -v $PWD:/data archivebox/archivebox:dev manage [...]
```

Existing users can be managed from the Admin UI here: `/admin/auth/user/`, and you can change your password here: `/admin/password_change/`.

Reverse Proxy Authentication

> Can be used with a reverse proxy auth provider like oauth2-proxy, Cloudflare Zero Trust, Authentik, and others.

Set these ArchiveBox configuration values based on your reverse proxy setup and needs:

```
# REQUIRED: the header where your upstream reverse proxy will place the authenticated␣
↪user's username/email
# EXAMPLE: Cf-Access-Authenticated-User-Email (if using Cloudflare Access / Zero Trust)
REVERSE_PROXY_USER_HEADER=X-Remote-User

# REQUIRED: the IP/CIDR of your upstream reverse proxy server
# WARNING: make sure this range contains ONLY your reverse proxy server!
# ArchiveBox will completely trust any IP in this range for authentication
REVERSE_PROXY_WHITELIST=192.0.2.3/32

# OPTIONAL: redirect users to an external URL after they log out
LOGOUT_REDIRECT_URL=https://auth.yourcompany.example.com/after/logout
```

- https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration#reverse_proxy_user_header
- https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration#reverse_proxy_whitelist
- https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration#logout_redirect_url

- https://github.com/ArchiveBox/ArchiveBox/pull/866

LDAP Authentication

Can be used with an SSO provider like Authentik, Authelia, Okta / Auth0, Keycloak, and others.

First, install the `ldap` add-on to use this feature (not needed for Docker Archivebox).

```
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox[ldap]>=0.9.
↪0rc0,<0.10'
```

Then set these configuration values to finish configuring LDAP:

```
LDAP_ENABLED=True
LDAP_SERVER_URI="ldap://ldap.example.com:3389"
LDAP_BIND_DN="ou=archivebox,ou=services,dc=ldap.example.com"
LDAP_BIND_PASSWORD="secret-bind-user-password"
LDAP_USER_BASE="ou=users,ou=archivebox,ou=services,dc=ldap.example.com"
LDAP_USER_FILTER="(objectClass=user)"

LDAP_USERNAME_ATTR="uid"
LDAP_FIRSTNAME_ATTR="givenName"
LDAP_LASTNAME_ATTR="sn"
LDAP_EMAIL_ATTR="mail"
```

- https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration#ldap-settings
- https://github.com/ArchiveBox/ArchiveBox/pull/1214
- https://github.com/django-auth-ldap/django-auth-ldap#example-configuration
- https://jumpcloud.com/blog/what-is-ldap-authentication

Not Yet Supported: SAML / OAuth2 / OpenID Authentication

We'd welcome PRs to add support for these using `django-allauth`!

These methods are not natively supported by ArchiveBox at the moment. However it is still possible to use them with ArchiveBox by running your own IdP (Identity Provider) server to act as a bridge (e.g. Authentik, Authelia, oauth2-proxy).

The IdP server can act as a middleman gateway to authenticate users using an external SAML/OAuth/OpenID/etc. provider (e.g. Google, Microsoft, Github, Facebook, etc.), and then pass on the authenticated user's session info to ArchiveBox using LDAP or reverse proxy headers (as described above).

- https://www.cloudflare.com/learning/access-management/what-is-saml/
- https://docs.goauthentik.io/docs/providers/saml/
- https://docs.goauthentik.io/docs/providers/oauth2/
- https://www.authelia.com/configuration/identity-providers/introduction/#openid-connect-10
- https://github.com/oauth2-proxy/oauth2-proxy
- https://oauth2-proxy.github.io/oauth2-proxy/configuration/overview

REST API

The REST API (available starting in v0.8.0) supports several methods of authentication for convenience.

To see API docs, try endpoints interactively, and see how auth works, visit this URL on your ArchiveBox server: http://api.archivebox.localhost:5797/api/v1/docs

To get started using the REST API, you can generate an API key for your user in the Admin Web UI: http://admin.archivebox.localhost:5797/admin/api/apitoken/add/

or by calling the `http://api.archivebox.localhost:5797/api/v1/auth/get_api_token` endpoint with a username & password:

```
curl -X 'POST' \
  'http://api.archivebox.localhost:5797/api/v1/auth/get_api_token' \
  -H 'Content-Type: application/json' \
  -d '{"username": "YOURUSERNAMEHERE", "password": "YOURPASSWORDHERE"}'
```

[!TIP] Bearer Tokens are the recommended method for the best balance of security and convenience.

API Bearer Token Authentication

Pass `Authorization=Bearer YOURAPITOKENHERE` as a request header.

```
curl -X 'GET' \
  'http://api.archivebox.localhost:5797/api/v1/core/snapshots?limit=10' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer YOURAPITOKENHERE'
```

API Request Header Authentication

This method is provided in case you have a reverse proxy in front of ArchiveBox that consumes the bearer header.

Pass `X-ArchiveBox-API-Key=YOURAPITOKENHERE` as a request header.

```
curl -X 'GET' \
  'http://api.archivebox.localhost:5797/api/v1/core/snapshots?limit=10' \
  -H 'accept: application/json' \
  -H 'X-ArchiveBox-API-Key: YOURAPITOKENHERE'
```

API Query Parameter Authentication

[!WARNING] This method is sometimes known as "Capability URLs" because anyone in possession of the URL can perform API actions. It comes with important security caveats and is not recommended unless you fully understand the risks.

Pass `api_key=YOURAPITOKENHERE` as a GET/POST query parameter.

```
curl -X 'GET' \
  'http://api.archivebox.localhost:5797/api/v1/core/snapshots?limit=10&api_
↪key=YOURAPITOKENHERE' \
  -H 'accept: application/json'
```

Further Reading

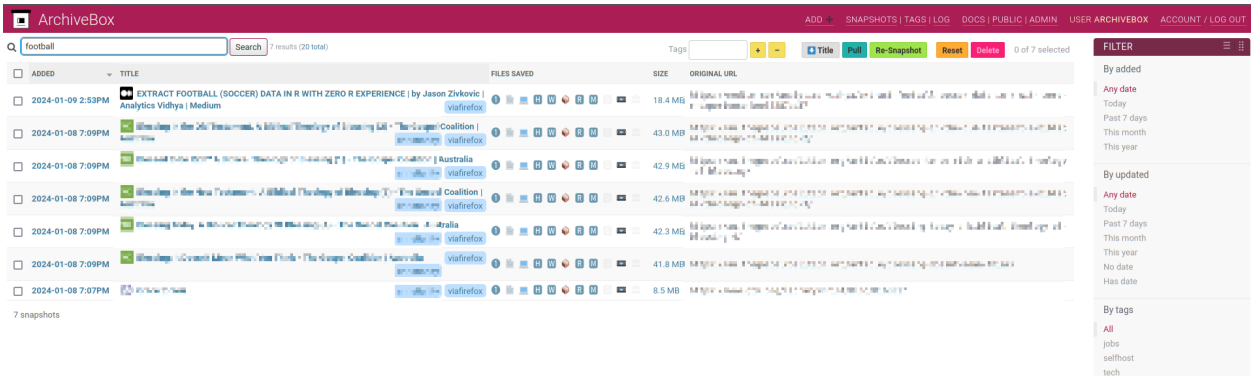
- The ArchiveBox API auth implementation: `archivebox/api/auth.py` + `archivebox/api/v1_auth.py`
- The `django-ninja` auth documentation (which powers our API)
- The Swagger auth documentation for the interactive API Docs UI

1.4.3 Setting Up Search

How to Search in ArchiveBox

You can search your ArchiveBox data in a number of ways:

- using the CLI: `archivebox search 'text to search'` (`archivebox search --help` for more)
- using the Web UI: both the `/public` index and `/admin/core/snapshot` pages provide a search box
- using the REST API: `/api/v1/core/snapshots?search=text+to+search&search_mode=contents`
- by searching the archive data folder directly with external tools (e.g. macOS Spotlight, Cerebro, `ag`, Yacy, etc.)



How Search Works

ArchiveBox search works by doing substring matches in `Snapshot` metadata fields (`url`, `title`, `timestamp`, `tags`), and by searching the full archived content within each Snapshot (using the selected search backend below). You can find the search implementation source code here: `archivebox/core/views.py: PublicIndex.get_queryset()`.

> *Note: ArchiveBox currently only returns the bare list of snapshots that match when performing a search.*
>
> This will be improved in the future to highlight the *specific paragraph/line/area that matched* within a Snapshot.
> For now we recommend using Ctl+F in the browser or one of the external tools listed above to further filter for a term within a Snapshot's contents.

ArchiveBox Search Backends

ArchiveBox provides a number of "Search Backend Engines" to tune its performance & behavior for different use-cases.

```
# this setting controls which search backend ArchiveBox uses
archivebox config --set SEARCH_BACKEND_ENGINE=sonic

# to see information about the backend you are currently using, run:
archivebox version
archivebox config --get SEARCH_BACKEND_ENGINE
```

ArchiveBox installs and enables both Sonic and `ripgrep`. Sonic is selected by default for fast indexed search, while ripgrep remains available as the filesystem fallback when Sonic is unavailable or explicitly selected.

[!TIP] **You should consider selecting `ripgrep` instead of Sonic if:**

- Sonic is unavailable on your platform
- you prefer filesystem scanning without a background daemon
- you need ripgrep-compatible regular expressions

`ripgrep` *(fallback)*

ArchiveBox resolves `ripgrep` through `abxpkg`: a compatible host installation is used first, otherwise a managed copy is installed.

```
archivebox install ripgrep
archivebox config --set SEARCH_BACKEND_ENGINE=ripgrep

# check the resolved provider, version, and projected binary:
archivebox version

# then try it out by searching via the Web UI or CLI:
archivebox search 'text to search for'
```

Pros

- supports advanced searching with regex patterns
- simple, few moving parts, and broadly available for all OSs and CPU architectures
- 0 idle resource use as there is no background indexer process running
- 0 additional disk storage needed as it searches the original data instead of maintaining a separate index
- reasonably fast on NVMe and SSD drives for small collections

Cons

- very slow as archive collection size increases (doesn't scale well beyond 500~1,000 Snapshots)
- very slow if underlying filesystem is slow (e.g. HDDs or network mounts)
- doesn't support stemming, boolean operators, or other advanced full-text search features

`ripgrep-all` (aka `rga`)

`ripgrep-all` supports more binary file types such as PDFs, eBooks, Office documents, zip, and tar files. It is useful as an external companion tool, but it is **not currently a supported drop-in binary for ArchiveBox's `ripgrep` backend**. The backend relies on `rg`'s command and output contract.

ugrep

ugrep is another capable external search tool, but it is **not a supported drop-in binary** for ArchiveBox's ripgrep backend. Contributions adding a dedicated integration are welcome.

ugrep is similar to ripgrep and ripgrep-all in that it's an indexless disk-search tool, but it provides some more of the full-text search features without the performance overhead of maintaining a separate search backend worker with an independent index.

https://github.com/Genivia/ugrep

Pros

- supports boolean operators in search queries
- supports binary formats like compressed archives, PDFs, eBooks, etc.
- better support for Unicode, special characters, and searching across multiple lines of text
- supports fuzzy search

Cons

- not as fast as sonic and but also not as simple as ripgrep
- not all of its features are fully integrated with ArchiveBox yet

sonic (the recommended upgrade path for most people)

Sonic is a fast, lightweight, rust-based alternative to super-heavy traditional search backends like Elasticsearch. It is capable of normalizing natural language search queries, fuzzy matching, and searching Unicode, without needing to maintain a duplicate document store index of all the searchable text.

Internally it functions as an index store, storing only the original IDs of the Snapshots with a super-compressed representation of the text. This allows it to scale to searching terabytes of archive data while maintaining an index only a fraction of that size.

ArchiveBox has supported Sonic for years, and it is the most thoroughly tested and recommended backend for ArchiveBox users that need to scale beyond ripgrep.

ArchiveBox resolves and starts Sonic through the same abxpkg lifecycle on both Docker and bare-metal installations.

```
archivebox config --set SEARCH_BACKEND_ENGINE=sonic
archivebox install sonic
archivebox update --index-only
archivebox search 'some text to search'
```

Run the same commands as docker compose run --rm archivebox ... when using Docker Compose.

Fore more detailed instructions see here...

Pros

- extremely fast, most queries complete in microseconds even with 100k+ snapshots
- maintains lightweight, compressed search index that is minuscule compared to original data

- all-in-one binary written in rust, available cross-platform and easy to deploy
- supports advanced full-text search features like normalization, stemming, etc.
- supports indexing and querying on a remote server (many ArchiveBox instances can share a single `sonic` instance)

Cons

- one extra dependency to install and background worker to keep running (Docker Compose makes this easy though)
- does not support searching binary files like PDFs, eBooks, compressed archives, etc.

SQLite FTS5

This is a recently added experimental option that uses a separate SQLite3 Database (similar to the one ArchiveBox already uses for Snapshot metadata) to provide full-text search.

```
archivebox config --set SEARCH_BACKEND_ENGINE=sqlite

# add existing data to index by running update:
archivebox update --index-only

# test it out using the archivebox Web UI or CLI:
archivebox search 'some text to search'
```

You can also inspect the separate FTS database directly:

```
sqlite3 ./search.sqlite3

> SELECT snapshot_id, url FROM search_index
      WHERE search_index MATCH 'some text to search';
```

```
# optional advanced tuning:
archivebox config --set FTS_SEPARATE_DATABASE=True
archivebox config --set FTS_TOKENIZERS="porter unicode61 remove_diacritics 2"
```

- https://www.sqlite.org/fts5.html
- https://github.com/ArchiveBox/ArchiveBox/pull/1241

Pros

- No additional dependencies needed to install, SQLite3 is already available and used by ArchiveBox
- No long-running background search worker process needed, 0 idle resource use
- Supports advanced full-text search features like boolean operators, stemming, phrases, etc.
- Comparable speed and efficiency to `sonic` for most use-cases (much faster than `ripgrep`/`ugrep`)
- Durability and portability, SQLite is widely used and supported by every major platform on earth

Cons

- Not as thoroughly-tested by ArchiveBox team as our `sonic` or `ripgrep` backends
- Maintains a (compressed, but still potentially large) duplicate copy of all searchable text in `search.sqlite3` db
- Does not support searching binary files PDFs, eBooks, compressed archives, etc.
- Search indexing and querying must be performed on same server as ArchiveBox data (we don't yet support sending FTS5 queries to a remote server)

Further Reading

- https://github.com/ArchiveBox/ArchiveBox/blob/dev/docker-compose.yml#:~:text=SEARCH_BACKEND_ENGINE
- https://plugins.archivebox.io/#search_backend_ripgrep
- https://plugins.archivebox.io/#search_backend_sonic
- https://plugins.archivebox.io/#search_backend_sqlite
- #22 Original Issue where full-text search functionality was proposed
- #543 + #570 Original PR where full-text search functionality was implemented
- #956 Documentation: Document how search works
- #654 Support: Search Backend only searching admin Snapshot fields instead of archive content
- #1087 Support: Help setting up full text search
- #1091 Support: Help switching to ripgrep-all
- #1318 Troubleshooting: Search times out on v0.7.2 installed on Synology using Portainer
- #1333 + #1316 Text Search and Filters don't work at the same time in the web UI
- #1320 Troubleshooting: Sonic backend Error: ENDED authentication_failed doesn't contain protocol(NUMBER)
- #1139 Feature Request: Add AI-assisted summarization, tagging, search, and more using LLMs / RAG
- #1358 Django Admin general improvements: tree view, better filters, better sorting, custom pages, etc.

1.4.4 Publishing Your Archive

There are two ways to publish your archive: using the `archivebox server` or by exporting and hosting it as static HTML.

1. Use the built-in web server

```
# start the webserver
archivebox server 0.0.0.0:5797
```

Open http://admin.archivebox.localhost:5797/admin/ in a browser to create the first admin and finish web setup. To create additional accounts from the CLI, use `archivebox manage createsuperuser`.

This server is enabled out-of-the-box if you're using `docker-compose` to run ArchiveBox. If `BASE_URL` is not configured yet, complete the first-run wizard before exposing ArchiveBox. It guides you through choosing a security mode

and shows the settings to enter in Cloudflare, Nginx Proxy Manager, Caddy, Traefik, Tailscale, or your hosting platform's ingress UI. Configure HTTPS there, then open the canonical URL you chose in the wizard. Existing `BASE_URL` and security settings are trusted, so configured servers skip this wizard.

After ingress is working, set the permissions depending on how public you want the archive to be:

```
archivebox config --set PUBLIC_INDEX=True
archivebox config --set PUBLIC_ADD_VIEW=True
archivebox config --set PERMISSIONS=public
```

> [!TIP] Advanced: You can use nginx to serve a static export directly from the filesystem. Do not proxy live replay paths back onto the admin origin; use ArchiveBox's security-mode routing.

2. Export and host it as static HTML

```
archivebox list --html --with-headers > index.html
archivebox list --json --with-headers > index.json

# then upload the entire output folder containing index.html and archive/ somewhere
# e.g. github pages or another static hosting provider

# you can also serve it with the simple python HTTP server
python3 -m http.server --bind 0.0.0.0 --directory . 5797
open http://127.0.0.1:5797
```

Here's a sample nginx configuration that works to serve your static archive folder:

```
location / {
    alias       /path/to/your/ArchiveBox/data/;
    index       index.html;
    autoindex   on;
    try_files   $uri $uri/ =404;
}
```

Make sure you're not running any content as CGI or PHP, you only want to serve static files!

The generated links are relative, so the export works at a domain root or a project subpath such as GitHub Pages. Snapshot pages and outputs remain under their real filesystem paths, for example:

`archive/users/alice/snapshots/20260829/example.com/SNAPSHOT_UUID/index.html`

ArchiveBox does not create top-level timestamp symlinks for current snapshots. Run `archivebox update --migrate-only` to move real legacy `archive/<timestamp>/` directories into the user-scoped layout and remove obsolete timestamp projections.

Security Concerns

> [!CAUTION] Re-hosting untrusted archived content on the same origin as an authenticated application can compromise that application.

Make sure you understand the dangers of hosting untrusted HTML/JS/CSS. The default `SERVER_SECURITY_MODE=auto` uses isolated subdomains with full replay on `*.localhost`, and a one-domain no-JS replay policy on ordinary public or LAN hostnames. Choose `safe-subdomains-fullreplay` only when wildcard DNS and TLS for `*.archive.example.com` are configured; it separates the admin, web, and API control planes from replay content and gives each Snapshot its own replay subdomain.

Do not serve the authenticated ArchiveBox application from a shared subdirectory such as `myapps.example.com/archivebox/`; it cannot provide the required origin isolation. A standalone static export can be hosted at a project subpath because it contains no authenticated control plane. If you do not need JavaScript-capable replay, you can also disable the relevant extractors with `WGET_ENABLED=False` and `DOM_ENABLED=False`.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#publishing
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#-things-to-watch-out-for-

---

Copyright Concerns

> [!WARNING] Be aware that some sites you archive may not allow you to rehost their content publicly for copyright reasons, it's up to you to host responsibly and respond to takedown requests appropriately based on the laws in your jurisdiction.

Archiving for personal backups, research, and some other use-cases are covered by fair use copyright exemptions in the USA, but if your archive can deprive the original author of revenue (e.g. if you rehost it for profit), then your use case might no longer be covered and you have to respond to DMCA takedown notices.

As a general rule of thumb:

- Copies cannot be made for commercial purposes
- The copying cannot be systematic (e.g., to replace subscriptions)
- All copies made must include a notice stating that the materials may be protected under copyright.

Please modify the *`FOOTER_INFO`* config variable to add your contact info to the footer of your index.

Note: ArchiveBox prevents search engines from indexing your archives using `/robots.txt` by default. It is not recommended to override this file in the collection's fixed `custom_templates/` directory, as public indexing often leads to automated takedown requests and abuse reports.

Keep in mind individuals, companies, schools, and libraries all have different copyright exemptions in different countries. Double check the specific laws for your situation in your own jurisdiction!

Further Reading: USA Copyright Law & Fair Use Exemptions

- https://www.copyright.gov/title17/
- https://help.archive.org/help/rights/
- https://blog.archive.org/2024/03/01/fair-use-in-action-at-the-internet-archive/
- https://www.lib.ncsu.edu/workshops/understanding-copyright-and-fair-use-archival-research
- https://libguides.colorado.edu/c.php?g=1154758&p=8428124
- https://fairuse.stanford.edu/2003/11/10/digital_preservation_and_copyr/
- https://guides.library.oregonstate.edu/copyright/libraries
- https://www.clir.org/pubs/reports/pub112/body/
- https://github.com/pirate/internet-archiving-talk

1.4.5 Scheduled Archiving

ArchiveBox now stores schedules in the database and lets the orchestrator materialize them into queued `Crawl` records at the right time. You no longer need host cron, user crontabs, or a separate `archivebox_scheduler` container when `archivebox server` is running.

How It Works

1. `archivebox schedule ...` creates a `CrawlSchedule` record plus a sealed template `Crawl`.
2. The long-running global orchestrator inside `archivebox server` watches enabled schedules.
3. When a schedule becomes due, the orchestrator creates a new queued `Crawl`.
4. That queued crawl is processed the same way as UI/API-submitted work.

One-shot foreground flows such as `archivebox add ...` continue to process only the crawl they were asked to run. They do not also sweep and execute unrelated scheduled crawls.

CLI Usage

```
cd ~/archivebox/data

archivebox schedule --every=daily --depth=1 https://example.com/feed.xml
archivebox schedule --every='0 */6 * * *' https://example.com/feed.xml
archivebox schedule --show
archivebox schedule --clear
archivebox schedule --run-all
archivebox schedule --foreground
```

Accepted schedule formats:

- Aliases: `minute`, `hour`, `day`, `week`, `month`, `year`, `daily`, `weekly`, `monthly`, `yearly`
- Cron expressions: e.g. `0 */6 * * *`

`archivebox schedule --run-all` dispatches every enabled schedule immediately: crawl schedules are enqueued, while maintenance schedules run directly.

`archivebox schedule --foreground` runs the global orchestrator in the foreground, which is useful outside `archivebox server` if you want a dedicated long-running scheduler/worker process without the web UI.

Running `archivebox schedule --every=day` with no `import_path` creates a recurring maintenance schedule. The scheduler dispatches its bounded database/filesystem maintenance directly instead of creating a synthetic Crawl or Snapshot.

Docker Compose

With the new orchestrator flow, you only need the main `archivebox` service:

```
services:
  archivebox:
    image: archivebox/archivebox:dev
    command: server --init 0.0.0.0:5797
    volumes:
      - ./data:/data
```

Create schedules with:

```
docker compose run --rm archivebox schedule --every=weekly --depth=1 https://example.com/
↪feed.xml
docker compose run --rm archivebox schedule --show
```

If the main `archivebox server` container is already running, its orchestrator will pick up future scheduled runs automatically. There is no scheduler sidecar to restart.

Examples

Archive a Twitter mirror once a week:

```
archivebox schedule --every=weekly --depth=1 'https://nitter.net/ArchiveBoxApp'
```

Archive a subreddit and linked discussions once a week:

```
archivebox config --set URL_ALLOWLIST='^http(s)?:\/\/(.+)?teddit\.net\/?.*$'
archivebox schedule --every=weekly --depth=1 'https://teddit.net/r/DataHoarder/'
```

Archive Hacker News every day:

```
archivebox config --set URL_DENYLIST='^http(s)?:\/\/(.+\.)?(youtube\.com)|(amazon\.com)\/
↪.*$'
archivebox schedule --every=daily --depth=1 'https://news.ycombinator.com'
```

Queue a daily maintenance update:

```
archivebox schedule --every=day
```

1.4.6 Chrome / Chromium Setup

ArchiveBox uses a Chromium-based browser to load pages and save screenshots, PDFs, and HTML.

Install ArchiveBox's browser

Docker images include the browser. For an installation without Docker, run these commands from your ArchiveBox data folder:

```
archivebox install chrome
archivebox version
```

ArchiveBox uses a suitable installed browser or downloads one if needed. `archivebox version` shows the browser it selected.

Set `CHROME_BINARY` to the browser's command or executable path, then run the installer:

```
archivebox config --set CHROME_BINARY=google-chrome
archivebox install chrome
```

1.4.7 Setting Up a Chromium User Profile

ArchiveBox uses a browser to load pages and save their content. If a page requires you to log in, ArchiveBox needs to be logged in too. Otherwise, it may save the login screen instead of the page.

A **browser profile** stores your logins and browser settings. Importing it into ArchiveBox lets ArchiveBox view pages using your account. ArchiveBox calls its copy a **persona**. Select that persona when saving a page.

Profile importing is for **Chromium-based browsers**. The examples below use Chrome; the same workflow applies to other browsers in that family. You can also *log in using ArchiveBox's own browser* instead of importing a profile.

Import an existing browser profile

These steps cover the common setup where your browser is on your computer and ArchiveBox runs in Docker on a server. You will export the profile on your computer, transfer the exported persona directly into the server's ArchiveBox data folder with `rsync`.

You need the *ArchiveBox CLI* on your computer and SSH access to your server. The temporary folder below is only for exporting the profile; you do not need to download your server's archive.

If ArchiveBox runs on the same computer as your browser, use your existing ArchiveBox data folder for step 1, skip step 2, and run `archivebox add --persona=personal URL` in step 3.

1. Export the profile on your computer

Open a page you want to save and check that you can view it while logged in. Close the browser, then create a temporary export folder:

```
mkdir -p ~/archivebox-profile-export
cd ~/archivebox-profile-export
archivebox init
```

Using the CLI:

```
archivebox persona create --import=chrome personal
```

`personal` is a name you choose for the persona. Replace `chrome` with your browser's import name: `chromium`, `brave`, or `edge` for those browsers. The command chooses a profile automatically; use `--profile=Default` or `--profile='Profile 1'` to select a specific one.

Using the UI instead: run `archivebox server` from the export folder and follow the *web UI setup*. Choose **Admin** → **Personas & Configs** → **Add persona** → **Use a detected profile**, name it `personal`, select the browser profile, and save. You can stop this local ArchiveBox server afterward.

Both methods create `~/archivebox-profile-export/personas/personal`, ready to transfer.

Check the **Profile Path** on your browser's version page (`chrome://version` in Chrome). The final folder name identifies the profile, such as `Default` or `Profile 1`.

For a browser installed in a different location, the CLI accepts `--source` for its profile path and `--browser-binary` for its executable. Run `archivebox persona create --help` for the available options. Automatic detection currently lists Chrome, Chromium, Brave, and Edge; other Chromium-based browsers need their paths supplied explicitly.

2. Transfer the persona to your server

Run these commands **on your computer**. Replace `user@your-server` with your SSH login and `~/archivebox/data` with your server's ArchiveBox data folder (the folder mounted at `/data` in Docker):

```
ssh user@your-server 'mkdir -p ~/archivebox/data/personas/personal'
rsync -av ~/archivebox-profile-export/personas/personal/ user@your-server:~/archivebox/
↪data/personas/personal/
```

The persona is now in place. Docker uses the existing data mount; no Compose changes are needed.

For a server without Docker, transfer to `personas/personal/` inside its ArchiveBox data folder in the same way.

3. Save and check the page

Connect to your server and enter the folder containing its Docker Compose file:

```
ssh user@your-server
cd ~/archivebox
```

Replace the example URL with the page you want to save:

```
docker compose exec archivebox archivebox add --persona=personal 'https://example.com/
↪page'
```

For a server without Docker, run `archivebox add --persona=personal URL` from its data folder.

Using the UI instead: if `personal` is not listed yet, choose **Admin** → **Personas & Configs** → **Add persona** → **Blank Persona**, name it `personal`, and save. This registers the transferred folder without changing its contents. Then open **Add URL**, paste the URL, and select `personal` before starting the capture.

Open the saved screenshot or HTML. Check that it contains the content you wanted, rather than a login screen. If the site requires you to log in again later, sign in in your browser and repeat the export and transfer.

Treat imported personas and logged-in snapshots as private: they can contain account information. Review the *guidance on sharing archives* before publishing them.

Setting up a new profile

Use this option if you prefer to log in through ArchiveBox's browser instead of importing a browser profile. Log in once, then select the same persona for future captures.

Non-Docker Setup (Local Host)

From your ArchiveBox data folder:

```
archivebox persona create personal
archivebox persona open personal
```

Log in to the sites you want to archive, then close the browser. Save a page with `archivebox add --persona=personal URL` and check the resulting Screenshot, DOM, PDF, or SingleFile output.

Docker VNC Setup

This opens the selected persona in a browser you can control remotely. It also works with an *imported profile*.

1. In your Docker Compose file, uncomment the `novnc` service and the `depends_on` block under `archivebox`. Uncomment `ARCHIVEBOX_VNC_PERSONA=personal` to select your persona.
2. From the Compose folder, run:

```
docker compose up -d
```

ArchiveBox waits for VNC to be ready, then opens the persona's browser with its imported logins. If an ArchiveBox browser is already open on that display, it waits for that browser to close first.

3. Visit http://localhost:8080/vnc.html and click **Connect**. For a remote server, first run this on your computer and keep the connection open:

```
ssh -N -L 8080:127.0.0.1:8080 user@your-server
```

4. Open a page and confirm you can view it while logged in. For a new persona, log in to the sites you want to archive. Then follow *Save and check the page*, selecting the same persona.

One-off Compose setup

To set up a persona without starting the ArchiveBox server, enable the same `novnc` service and healthy `depends_on` block, then run:

```
docker compose up -d --wait novnc
docker compose run --rm archivebox persona create personal
docker compose run --rm -e DISPLAY=novnc:0.0 archivebox persona open personal
```

Connect at http://localhost:8080/vnc.html, log in, and close the browser when finished. The persona remains in `data/personas/personal` across one-off containers.

`persona open` resolves the installed Chromium binary automatically. Do not hard-code `/data/lib/env/bin/chromium` or a versioned Playwright cache path: the browser location varies by image version and architecture.

An explicitly set `DISPLAY` is preserved. When it is unset, ArchiveBox probes `novnc:0.0` and falls back to headless operation if unavailable. Set `DISPLAY=` to disable this automatic display selection.

Troubleshooting Chromium Install

- **Browser missing or unable to start:** see *Chromium troubleshooting*.
- **Saved a login screen:** confirm you can view the page in the source browser, re-import the correct profile, and select that persona when saving.
- **Need browser settings:** see the Chrome plugin configuration.

1.4.8 Upgrading Versions

```
# Stop any running ArchiveBox server/workers first (Ctrl+C, or stop the service/process␣
↪manager that started them)

# Back up the entire collection, including index.sqlite3, ArchiveBox.conf, and all␣
↪archived outputs
cd ~/archivebox
tar -czf "archivebox-data-$(date +%s).tar.gz" data/

# Update ArchiveBox using the package manager you originally installed it with
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox>=0.9.0rc0,<0.10
↪'
# or: sudo apt update && sudo apt install --only-upgrade archivebox
# or: brew update && brew upgrade archivebox

cd data
archivebox init
archivebox install
archivebox update --migrate-only
archivebox status

# Docker Compose upgrade
cd ~/archivebox
```

(continues on next page)

(continued from previous page)

```
docker compose down
docker compose pull
docker compose run --rm archivebox init
docker compose run --rm archivebox update --migrate-only
docker compose up -d
```

Upgrading checklist:

1. Find the version you want to upgrade to on https://github.com/ArchiveBox/ArchiveBox/releases
2. **Read the release notes carefully** for any instructions or extra steps around upgrading for each release you're skipping or installing
3. **Stop any running ArchiveBox server, scheduler, and worker processes**, then back up the entire collection data directory before upgrading. `archivebox config --get ...` and a database-only backup do not include archived outputs. `cd ~/archivebox && tar -czf "archivebox-data-$(date +%s).tar.gz" data/`
4. Follow the steps below for your installation method. Bare-metal installs run `archivebox init`, `archivebox install`, and `archivebox update --migrate-only` inside the collection; Docker images already include runtime dependencies.
5. Confirm the upgrade succeeded and check for any orphan/corrupted snapshots with `archivebox status`

Open an issue in our bug tracker if you experience any problems with upgrading/merging/modifying collections.

Note: It's recommended to only upgrade one major version at a time. Follow each intermediate release's upgrade instructions and Python requirements rather than installing old releases with the current Python version. Upgrading directly across multiple major versions may work in some cases, but is not recommended for maximum data safety.

How it works internally:

The same command is used for initializing a new archive and upgrading an existing database. `archivebox init` is idempotent and can safely be run multiple times; it applies database migrations and prepares collection-level state. For bare-metal installs, `archivebox install` resolves runtime dependencies for the new version; Docker images include those dependencies at build time. `archivebox update --migrate-only` performs filesystem migrations and reconciles Snapshot metadata with the current layout without scheduling normal archive maintenance jobs. `archivebox status` checks collection health afterward. Ordinary updates stay lazy and do not discover current-layout directories copied from another collection. After copying directories, use `archivebox update --rescan --migrate-only --index-only`; see *Merging Collections*.

There are three main areas on disk that ArchiveBox modifies during upgrades:

- `index.sqlite3` contains the SQLite3 DB index that gets upgraded automatically by Django based on the changes in `archivebox/core/models.py`.
- `archive/users/<user>/snapshots/<date>/<domain>/<uuid>/index.jsonl` stores per-Snapshot metadata alongside plugin-namespaced output. `archivebox update --migrate-only` may rewrite metadata, migrate older layouts, and remove obsolete timestamp projections after verified migration.
- Snapshot output directories and plugin paths can move as filesystem schemas evolve, so the entire `archive/` tree must be backed up with the database.

`ArchiveBox.conf` is migrated through the normal config loader/writer when options are renamed or normalized. Back it up with the rest of the collection and review release notes for config changes.

As of v0.4 and above, ArchiveBox uses the Django migrations system for deterministic, atomic, safe upgrades, so your DB should always be left in a consistent state in the event of a failure or power outage. If you need help fixing a corrupted collection, open an issue using the link above.

More info:

- https://docs.djangoproject.com/en/4.0/topics/migrations/
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#database-migrations-errors-or-upgrade-issues
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting

---

Upgrading with Docker Compose

Using Docker Compose keeps the image and migration commands consistent across upgrades.

```
cd ~/archivebox        # or wherever your folder containing docker-compose.yml is
docker compose down    # stop the currently running ArchiveBox containers
docker compose pull    # pull the latest image version from Docker Hub
docker compose run --rm archivebox init
docker compose run --rm archivebox update --migrate-only
docker compose up -d
```

More info:

- https://github.com/ArchiveBox/ArchiveBox#-easy-setup
- https://github.com/ArchiveBox/ArchiveBox/wiki/Docker#docker-compose
- https://github.com/ArchiveBox/ArchiveBox/wiki/Docker#setup

Upgrading with plain Docker

Upgrading with plain Docker is similar to the process with Docker Compose, but each command must mount the existing collection directory.

```
docker ps -a -q  --filter ancestor=archivebox/archivebox  # find any currently running␣
↪archivebox containers
docker stop CONTAINER_ID

cd ~/archivebox/data  # or wherever your existing collection is stored
docker pull archivebox/archivebox:dev
docker run --rm -v $PWD:/data -it archivebox/archivebox:dev init
docker run --rm -v $PWD:/data -it archivebox/archivebox:dev update --migrate-only

# restart the archivebox server container if needed
docker run -v $PWD:/data -it -p 5797:5797 archivebox/archivebox:dev server 0.0.0.0:5797
```

More info:

- https://github.com/ArchiveBox/ArchiveBox#-easy-setup
- https://github.com/ArchiveBox/ArchiveBox/wiki/Docker#docker
- https://github.com/ArchiveBox/ArchiveBox/wiki/Docker#setup-1

Upgrading with a package manager

Package manager releases take a lot of effort to maintain (*contributions welcome!*) and sometimes lag behind the Docker releases. We make a best effort to have the latest release available through all channels within a reasonable timeframe.

```
cd ~/archivebox/data   # or wherever your data folder is

# upgrade ArchiveBox using the package manager you originally used to install it
uv tool install --python 3.13 --prerelease explicit --upgrade 'archivebox>=0.9.0rc0,<0.10
↪'
```

Or, for apt:

```
cd ~/archivebox/data   # or wherever your data folder is
sudo apt update
sudo apt install --only-upgrade archivebox
```

Or, for Homebrew:

```
cd ~/archivebox/data   # or wherever your data folder is
brew update
brew upgrade archivebox
```

Then finish the collection upgrade:

```
archivebox init        # run init to upgrade the collection to the latest version
archivebox install     # refresh runtime dependencies if needed

archivebox update --migrate-only  # migrate/reconcile Snapshot files and metadata

archivebox status      # check that everything succeeded
```

More info:

- https://github.com/ArchiveBox/ArchiveBox#-package-manager-setup
- https://github.com/ArchiveBox/ArchiveBox/wiki/Install#option-c-bare-metal-setup
- https://github.com/ArchiveBox/homebrew-archivebox
- https://github.com/ArchiveBox/debian-archivebox
- https://aur.archlinux.org/packages/archivebox
- https://github.com/NixOS/nixpkgs/blob/master/pkgs/applications/misc/archivebox/default.nix

Merge two or more existing archives

See *Merging Collections*...

Related Documents

- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting#database
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#disk-layout
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#large-archives
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#python-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage

1.4.9 Upgrading or Merging Archives

Moved to:

- *Upgrading*
- *Merging Collections*
- *Database Troubleshooting*

1.4.10 Merging Collections

Copy snapshot directories into the destination collection, then explicitly rescan them. Ordinary `archivebox init` and `archivebox update` intentionally avoid discovering current-layout orphan directories: that would make every startup/update scan the entire collection.

1. Back up both collections and finish active captures before copying. Upgrade legacy collections using the normal *Upgrading* instructions.
2. Initialize the destination collection with `archivebox init`.
3. Copy or drag the contents of each source `archive/` into the destination `archive/`, preserving the `users/<username>/snapshots/<date>/<domain>/<uuid>/` hierarchy and every snapshot's `index.jsonl`. Inspect collisions; do not overwrite different files at the same path. Do not replace the destination SQLite database.
4. From the destination data directory, run:

```
archivebox init
archivebox update --rescan --migrate-only --index-only
archivebox status
```

With Docker Compose:

```
docker compose run --rm archivebox init
docker compose run --rm archivebox update --rescan --migrate-only --index-only
docker compose run --rm archivebox status
```

`--rescan` deliberately performs an O(N) scan of snapshot directories. It imports sealed orphan snapshots, restores missing archive-result records and tags, and repairs missing metadata files and crawl links for known snapshots. It does not recover missing payload bytes or guess the identity of malformed metadata. Unresolved directories are reported, retained in place, and cause a nonzero exit status. Real legacy `archive/<timestamp>/` directories still use the existing filesystem migration flow.

The scan preserves Snapshot, Crawl, and ArchiveResult IDs and the timestamps present in their metadata. ID/URL/timestamp conflicts are reported rather than assigned replacement IDs. Existing destination records take precedence over imported metadata. Missing owners are created as inactive accounts, without importing passwords or granting administrator privileges. Older metadata without permissions imports privately; exports without Crawl records or creation timestamps cannot reproduce those missing fields exactly. Machine-local processes, credentials, schedules, and full user accounts are not synchronized.

Stop with Ctrl+C and rerun the same command to resume. There is no checkpoint file or new database state: each successful record is durable, and the scan compares existing IDs/results/tags/links before doing repair work. It discovers orphans before repairing known snapshots, with newest date/UUID names first in each group. Completed imports move to the known group on restart. Add `--reverse` to process oldest first instead. Known snapshot asset checks reuse the existing output manifest scanner. A restart still enumerates directories and reads metadata to find incomplete work; timestamps alone cannot prove completeness because copied files may retain old dates. Filters, `--resume`, and `--continuous` cannot be combined with `--rescan`.

`--rescan` discovers filesystem-only snapshots and implies filesystem maintenance without new captures. `--migrate-only` eagerly finishes pending migrations for known database snapshots; ordinary updates leave them

lazy. `--index-only` checks known snapshots, rebuilds output metadata, and backfills missing search indexes from archived content. All three explicit modes default to newest first; `--reverse` flips the order. `archivebox update --rescan` is enough when search backfill is unnecessary.

For shared remote mounts, first make sure newly written objects have uploaded and the destination's directory cache can see them. Sharing `archive/` does not synchronize independent databases automatically. Confirm imported entries and representative archived files before removing any source or backup copies.

Modify the ArchiveBox SQLite3 DB directly

If you need to automate changes to the ArchiveBox DB (for example adding a User from an Ansible script), you can modify the SQLite3 DB directly.

Note, this is often unnecessary for modifying ArchiveBox on a host that doesn't have the CLI installed, as you can also copy the `index.sqlite3` to a local machine that has it, do the modifications locally, then copy the modified db back into place on the host. (Docker/CLI/GUI/Web ArchiveBox all share the same DB schema/format)

```
cd ~/archivebox/data    # cd into your archivebox collection dir
sqlite3 index.sqlite3   # open the db with sqlite3 shell
```

Example: Modifying an existing user's email

```
UPDATE auth_user
SET email = 'someNewEmail@example.com', is_superuser = 1
WHERE username = 'someUsernameHere';
```

Example: Adding a new user with a hashed password

Note: this is just an example to demonstrate direct database usage. For initial setup, open the Admin UI and create the first admin there.

1. First, generate the hashed password in a Python shell using Django's `make_password` function.

Use the Django version bundled with the ArchiveBox installation that owns the collection:

```
archivebox shell -c "from django.contrib.auth.hashers import make_password; print(make_
↪password('somePasswordHere', 'someSaltHere', 'pbkdf2_sha256'))"
```

```
>>> from django.contrib.auth.hashers import make_password
>>> make_password('somePasswordHere', 'someSaltHere', 'pbkdf2_sha256')         # choose␣
↪a password and a salt (can be anything 12 chars long)
'pbkdf2_sha256$...$someSaltHere$...'
```

2. Use the generated hashed password to insert a new User row in the SQLite3 database directly:

```
cd ~/archivebox/data    # cd into your archivebox collection dir
sqlite3 index.sqlite3   # open the db with sqlite3 shell
```

```
INSERT INTO "auth_user" ("password", "last_login", "is_superuser", "username", "first_
↪name", "last_name", "email", "is_staff", "is_active", "date_joined")
VALUES ('GENERATED_PASSWORD_HASH', NULL, 0, 'someUsername', '', '', 'someEmail@example.
↪com', 0, 1, '2022-03-22 23:34:02.333042')
```

Replace the values above with the desired username, email, and password hash from python output^.

3. Log in using the new generated user to confirm it works http://admin.archivebox.localhost:5797/admin/login/ user: `someUsername` pass:`somePasswordHere`

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#python-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage

---

Database Troubleshooting

See here *Troubleshooting: Database…*

Related Documents

- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#disk-layout
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#large-archives
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#python-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage

1.4.11 Troubleshooting

If you need help or have a question, you can open an issue or reach out on Twitter.

What are you having an issue with?:

- *Installing ArchiveBox*
- *Upgrading ArchiveBox*
- *Configuring ArchiveBox*
- *Archiving content with ArchiveBox*
- *Hosting your collection publicly*
- *Database and filesystem issues*

Installing

If using `archivebox` without Docker, make sure you've followed the full guide in the *Install* instructions first. Then check here for help depending on what component you need help with.

Then make sure `archivebox` is installed available in your `$PATH`.

```
apt show archivebox      # show info about the apt-installed version of archivebox
brew info archivebox     # show info about the brew-installed version of archivebox
uv tool list             # show info about uv-installed tools

echo $PATH               # show the directories your system is searching for binaries
type -a archivebox       # show all installed archivebox binaries available
```

Show the full archivebox version info + info about all installed dependencies:

```
archivebox version       # shows lots of useful info about installed dependencies and␣
↪more
```

(ensure the version shown is the most recent available from Releases)

macOS

ArchiveBox can be installed with Homebrew or `uv` on macOS:

```
brew tap archivebox/archivebox
brew trust archivebox/archivebox
brew install archivebox

mkdir -p ~/archivebox/data
cd ~/archivebox/data     # (for example, can be anywhere)

archivebox init
archivebox install       # finish installing runtime dependencies
```

More info: https://github.com/ArchiveBox/homebrew-archivebox

Python and uv

ArchiveBox's supported bare-metal install uses `uv`, which manages the Python 3.13 environment for the tool:

```
uv --version
uv tool list
archivebox version
```

If `archivebox` is missing, repeat the `uv tool install` command from the *Install* guide.

Chromium/Google Chrome

For more info, see the *Chromium Install* page.

ArchiveBox resolves Chrome through `abxpkg`, preferring a compatible browser already installed on the host and otherwise installing a managed build:

```
archivebox install chrome
archivebox version
```

The version output shows the selected provider, version, and projected path. If it reports an incompatible host browser, update that browser or let ArchiveBox install the managed fallback; do not bypass the resolver with an unrelated path.

Wget & Curl

Resolve or update both tools through the same installer:

```
archivebox install wget curl
archivebox version
```

NPM Dependencies

Node.js and JavaScript extractor packages such as `readability` and `singlefile` are resolved through `abxpkg`; they do not require a separate global npm setup.

```
cd ~/archivebox/data   # go into your data directory
archivebox install node singlefile readability
archivebox version
```

Archiving

No links parsed from export file

Please open an issue with a description of where you got the export, and preferably your export file attached (you can redact the links). We'll fix the parser to support your format.

Lots of skipped sites

If you ran the archiver once, it wont re-download sites subsequent times, it will only download new links. If you haven't already run it, make sure you have a working internet connection and that the parsed URLs look correct. You can check the ArchiveBox stdout logs or the Web UI to see what links it's downloading.

To intentionally capture an already indexed URL again, use `archivebox add --no-only-new URL`. Do not delete or move the `archive/` tree to work around `ONLY_NEW`; that separates database state from its Snapshot files.

Lots of errors

Make sure you have all the dependencies installed and that you're able to visit the links from your browser normally. Open an issue with a description of the errors if you're still having problems.

Lots of broken links from the index

Not all sites can be effectively archived with each method, that's why it's best to use a combination of `wget`, PDFs, and screenshots. If it seems like more than 10-20% of sites in the archive are broken, open an issue with some of the URLs that failed to be archived and I'll investigate.

Removing unwanted links from the index

```
archivebox remove --help
```

Hosting the Archive

If you're having issues trying to host the archive via nginx, make sure you already have nginx running with SSL. If you don't, google around, there are plenty of tutorials to help get that set up. Open an issue if you have problem with a particular nginx config.

Other database or filesystem issues

Docker Permissions issues

The Docker entrypoint starts as root for mount and runtime-cache setup, then runs ArchiveBox and Chrome as a non-root user. It checks required paths as that user first, so normal starts do not traverse cache trees; an inaccessible runtime/cache tree may need a one-time recursive repair. Browser-lock cleanup checks only known files at each persona profile root, not the contents of large profiles. By default the entrypoint uses the first non-root numeric owner

found in the collection, falling back to `911:911` for a new or root-owned collection. Set `PUID` and `PGID` when an NFS/CIFS/FUSE server or host requires a specific numeric identity.

Required collection directories get a non-root access check or creation attempt first, followed by real create/delete probes on the critical output paths. Only a failed check triggers a shallow root `chown`/`chmod` of the exact collection path. This avoids repeated metadata changes on writable `root_squash` or synthetic-permission mounts. Startup never recursively scans or changes `data/archive`: byte size is not a safe proxy for traversal cost, and a large or metadata-heavy archive can take hours or days to walk.

With SQLite, `index.sqlite3`, configuration, logs, temp/runtime files, and sidecar databases must remain on reliable local storage; only `data/archive/` may be remote. PostgreSQL is the supported exception for the main index.

The default `restart: unless-stopped` policy repeats a startup failure but does not create it and should not be removed. If the entrypoint still cannot create and delete its probe file, stop the noisy loop, inspect the resolved volume sources and logs, then retry once after fixing the server-side UID/GID mapping or ACL:

```
docker compose stop archivebox
docker compose config
docker compose logs --tail=200 archivebox
docker compose run --rm archivebox init
docker compose up -d --wait
```

A read-only mount or NFS export that denies both the selected user and root cannot be repaired from inside the container.

Database

Database and filesystem issues are uncommon but do come up from time to time (especially when using networked storage, large archives, or multiple ArchiveBox processes for a single collection).

Generally, these commands can help you resolve most issues:

```
archivebox init                 # upgrade the archivebox collection
archivebox install              # upgrade the archivebox runtime dependencies
archivebox update --migrate-only  # migrate/reconcile Snapshot files and metadata
archivebox server --debug       # run the server with more verbose debug log output
archivebox shell                # access the Python API / Django management shell
sqlite3 index.sqlite3           # access the SQLite3 SQL database shell
```

Don't be scared by the volume of content here. Almost all of these issues linked below are duplicates or old resolved bugs, but they contain valuable context and troubleshooting steps if you're trying to figure out the cause of a problem with your setup.

Filesystem doesn't support FSYNC (e.g. network mounts)

The `index.sqlite3` file must be stored on a filesystem that supports FSYNC (most local filesystems) in order to ensure SQLite3 database integrity when multiple ArchiveBox processes may be accessing it simultaneously. However, the `./archive` folder can be on a NAS or other filesystem that does not support FSYNC.

- Archivebox hangs when initializing collection on network drive that doesn't support FSYNC #742
- Question: How to run AB on localhost but store data on NAS? #894
- Question: Docker on Windows archiving to an SMB path that doesn't support FSYNC #722
- Support for network drives or filesystems that don't implement FSYNC #456

More info:

- https://www.geeksforgeeks.org/python-os-fsync-method/
- https://man7.org/linux/man-pages/man2/fdatasync.2.html
- https://www.samba.org/samba/docs/current/man-html/smb.conf.5.html
- https://eclecticlight.co/2022/02/18/how-can-you-trust-a-disk-to-write-data/

Database and filesystem contention issues when running multiple ArchiveBox processes

ArchiveBox can sometimes struggle when archiving many links in parallel with multiple ArchiveBox processes trying to write to the database at the same time, leading to errors like this:

```
Unable to create the django_migrations table (database is locked)
```

These errors can also be encountered when there are permissions, network, or filesystem issues preventing writes to `index.sqlite3`.

- Question: Unable to create the django_migrations table (database is locked) - When OUTPUT_DIR to SAMBA share #946
- Question: ... Unable to create the django_migrations table (database is locked) #880
- Database is locked and other weird behavior when doing simultaneous adds #781
- Bugfix: Retry on "database locked" error (or add support for PostgreSQL/MySQL DB backend) #601
- Architecture: Use multiple cores to run link archiving in parallel #91
- ArchiveBox index corruption when running multiple import processes on v0.5.0 #454
- Architecture: Concurrent runs accidentally delete each other's temp files, leaving the index broken #234
- Database is locked and other weird behavior when doing simultaneous adds #781
- Bugfix: Retry on "database locked" error (or add support for PostgreSQL/MySQL DB backend) #601

More info:

- https://www.sqlite.org/lockingv3.html
- https://charlesleifer.com/blog/going-fast-with-sqlite-and-python/
- https://victoria.dev/blog/sqlite-in-production-with-wal/
- https://code.djangoproject.com/ticket/29280
- https://stackoverflow.com/questions/47761570/how-can-i-avoid-database-is-locked-sqlite3-errors-in-django

Database migrations errors or upgrade issues

Migration or upgrade issues happen occasionally with some niche setups or when skipping major versions during archiving.
Always backup your archive before upgrading, but know that migrations are deterministic and atomic using Django's migration system, so a failed migration does not mean your archive is unrecoverable, you just have to downgrade to the previous stable major version then continue upgrading.

```
archivebox init  # this usually applies any necessary migrations (atomically and␣
↪indempotently, safe to run multiple times)
```

- Bug: NOT NULL constraint failed: core_archiveresult.output when upgrading v0.4.24 archive to v0.6 #705
- Bugfix: sqlite3.IntegrityError: NOT NULL constraint failed: core_archiveresult.cmd_version and .output #597

- Error: django.db.utils.IntegrityError: UNIQUE constraint failed: core_tag.slug #596
- Bugfix: django.db.utils.IntegrityError: UNIQUE constraint failed: core_snapshot.timestamp #412
- Best Practices for Backup/Restore #341
- Bug: Running archivebox update –index-only doesn't upgrade Snapshot index.{html,json} files #962
- Feature Request: Deduplicate files on archives #704

More info:

- https://docs.djangoproject.com/en/4.0/topics/migrations/
- https://realpython.com/django-migrations-a-primer/
- https://realpython.com/digging-deeper-into-migrations/
- https://www.kite.com/blog/python/django-database-migrations-overview/
- https://markusholtermann.eu/2021/06/writing-safe-database-migrations-in-django/

Repairing a corrupted SQLite3 database file

A corrupted database file can theoretically only happen if an external process or filesystem error corrupts the SQLite3 database (there have only been two reports of a user encountering this in real life). If you ever need to repair a corrupted ArchiveBox index you can run the following steps.

Note this is specific to this error, these steps do not apply to other migrations/db errors (see above/below for other issues):

```
sqlite3.DatabaseError: database disk image is malformed
```

Generally all index issues should be fixable by running `archivebox init`.
Use `archivebox status` to inspect collection health. To deliberately discover copied orphan snapshots and repair metadata/links, run `archivebox update --rescan --migrate-only`. This scans all snapshot directories and may take a long time on remote storage; rerun the same command after interruption. See *Merging Collections* for identity preservation and conflict handling.

Error output:

```
[i] [2022-03-24 20:37:27] ArchiveBox v0.6.2: archivebox init
    > /data

[^] Verifying and updating existing ArchiveBox collection to v0.6.2...
----------------------------------------------------------------------

[*] Verifying archive folder structure...
    + ./archive, ./sources, ./logs...
    + ./ArchiveBox.conf...

[*] Verifying main SQL index and running any migrations needed...
Traceback (most recent call last):
  File "/usr/local/lib/python3.9/site-packages/django/db/backends/utils.py", line 82, in␣
↪_execute
    return self.cursor.execute(sql)
  File "/usr/local/lib/python3.9/site-packages/django/db/backends/sqlite3/base.py", line␣
↪411, in execute
    return Database.Cursor.execute(self, query)
sqlite3.DatabaseError: database disk image is malformed
```

Steps to fix:

```
cd ~/archivebox/data
echo '.dump' | sqlite3 index.sqlite3 | sqlite3 repaired_index.sqlite3
mv index.sqlite3 corrupt_index.sqlite3
mv repaired_index.sqlite3 index.sqlite3
```

More info:

- https://github.com/ArchiveBox/ArchiveBox/issues/955 and https://github.com/ArchiveBox/ArchiveBox/issues/1699
- https://stackoverflow.com/questions/5274202/sqlite3-database-or-disk-is-full-the-database-disk-image-is-malformed

---

See here for more info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading
- https://github.com/ArchiveBox/ArchiveBox/wiki/Merging-Collections
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#python-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#run-archivebox-as-an-unprivileged-user
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder

1.5 Architecture

1.5.1 ArchiveBox Architecture Diagrams

This page is a map of the current execution and persistence paths. The implementation lives primarily in:

- `archivebox/cli/` for CLI entry points
- `archivebox/services/runner.py` for crawl and snapshot execution
- `archivebox/crawls/models.py` for the `Crawl` model and its atomic queue transitions
- `archivebox/core/models.py` for `Snapshot`, `ArchiveResult`, and Snapshot queue transitions
- `archivebox/services/` for bus event projectors
- `abxpkg` and `abx-plugins` for binary resolution and plugin hooks

High-Level Execution Flow

ArchiveBox has one normal crawl execution path. CLI commands and web/API actions create or select database rows, then call the same runner. The runner emits lifecycle events, abx-plugin hooks do the extraction work, and service projectors persist processes and results.

Binary discovery and installation always goes through abxpkg. Compatible host binaries are preferred; managed providers are the fallback. Resolved binaries are projected into `LIB_DIR/env/bin` before programmatic use. `LIB_DIR/bin` is only a convenience directory for humans.

Persistent Data

The database is the source of truth for model state. Snapshot directories contain captured artifacts and rendered metadata. Older collections may also contain legacy timestamp-named snapshot directories.

`Crawl` Queue Lifecycle

Implemented directly by `Crawl` in `archivebox/crawls/models.py`. The database row is the durable state; the runner claims `retry_at` with a conditional update before it performs side effects, then calls the model's explicit lifecycle methods. There is deliberately no second in-memory state machine that can drift from the row owned by another process.

A crawl owns a set of snapshots. The runner creates or discovers those snapshots and projects crawl events while the row is `STARTED`; sealing waits for their normal lifecycle to finish. Pausing also schedules child snapshots to pause, and resuming returns the crawl to the runnable queue. Scheduled maintenance is dispatched directly by `CrawlSchedule`; it does not create a synthetic crawl or snapshot.

`Snapshot` Queue Lifecycle

Implemented directly by `Snapshot` in `archivebox/core/models.py`, using the same conditional `retry_at` claim protocol as `Crawl`.

The runner creates one queued `ArchiveResult` per selected hook, executes those hooks through the shared event bus, and seals the snapshot after every result reaches a final status. The narrow search-index maintenance operation on an already sealed snapshot is the intentional exception; it does not reopen or invent a second general lifecycle path.

`ArchiveResult` Projection

`ArchiveResult` is not driven by a separate in-memory state machine. The runner creates queued rows, and `ArchiveResultService` projects `ArchiveResultEvent` and `ProcessCompletedEvent` data into them.

`succeeded`, `failed`, `skipped`, and `noresults` are final result statuses. Each row identifies the plugin and hook that produced it and stores structured output, file metadata, timing, and error details.

1.6 API Reference

1.6.1 API Reference

This page contains auto-generated API reference documentation[1].

archivebox

Subpackages

archivebox.machine

Submodules

archivebox.machine.apps

Module Contents

Classes

[1] Created with sphinx-autodoc2

MachineConfig

Functions

register_admin

API

class archivebox.machine.apps.**MachineConfig**

Bases: django.apps.AppConfig

default_auto_field

'django.db.models.BigAutoField'

name

'archivebox.machine'

label

'machine'

verbose_name

'Machine Info'

archivebox.machine.apps.**register_admin**(*admin_site*)

archivebox.machine.detect

Module Contents

Functions

_run_abxpkg_host_binary
get_vm_info
get_public_ip
get_local_ip
unknown_if_blank
get_isp_info
get_host_network
get_os_info
get_host_stats
get_host_guid

Data

PACKAGE_DIR
DATA_DIR
ip_addrs
mac_addrs

API

archivebox.machine.detect.**PACKAGE_DIR**

None

archivebox.machine.detect.**DATA_DIR**

'resolve(...)'

archivebox.machine.detect.**_run_abxpkg_host_binary**(*name: str*, **args: str*) → subprocess.CompletedProcess[str]

archivebox.machine.detect.**get_vm_info**()

archivebox.machine.detect.**get_public_ip**() → str

archivebox.machine.detect.**get_local_ip**(*remote_ip: str = '1.1.1.1'*, *remote_port: int = 80*) → str

archivebox.machine.detect.**ip_addrs**

None

archivebox.machine.detect.**mac_addrs**

None

archivebox.machine.detect.**unknown_if_blank**(*value*)

archivebox.machine.detect.**get_isp_info**(*ip=None*)

archivebox.machine.detect.**get_host_network**() → dict[str, Any]

archivebox.machine.detect.**get_os_info**() → dict[str, Any]

archivebox.machine.detect.**get_host_stats**() → dict[str, Any]

archivebox.machine.detect.**get_host_guid**() → str

archivebox.machine.models

Module Contents

Classes

MachineManager	
Machine	
NetworkInterfaceManager	
NetworkInterface	
BinaryManager	
Binary	Tracks a binary on a specific machine.
ProcessManager	Manager for Process model.
Process	Tracks a single OS process execution.

Functions

get_current_pid_namespace	Return the OS PID namespace that makes numeric PIDs meaningful.

continues on next page

Table 6 – continued from previous page

_default_exit_code_for_unowned_process	
_find_existing_binary_for_reference	
_canonical_binary_name	
_get_process_binary_env_keys	
_sanitize_machine_config	Validate `Machine.config` in place.

Data

_psutil
_CURRENT_MACHINE
_CURRENT_INTERFACE
_CURRENT_BINARIES
_CURRENT_PROCESS
MACHINE_RECHECK_INTERVAL
NETWORK_INTERFACE_RECHECK_INTERVAL
BINARY_RECHECK_INTERVAL
PID_REUSE_WINDOW
PROCESS_TIMEOUT_GRACE
START_TIME_TOLERANCE
PROCESS_PID_NAMESPACE_KEY

API

archivebox.machine.models.**_psutil: Any | None**

None

archivebox.machine.models.**_CURRENT_MACHINE: *archivebox.machine.models.Machine* | None**

None

archivebox.machine.models.**_CURRENT_INTERFACE: *archivebox.machine.models.NetworkInterface* | None**

None

archivebox.machine.models.**_CURRENT_BINARIES: dict[str, *archivebox.machine.models.Binary*]**

None

archivebox.machine.models.**_CURRENT_PROCESS: *archivebox.machine.models.Process* | None**

None

archivebox.machine.models.**MACHINE_RECHECK_INTERVAL**

None

archivebox.machine.models.**NETWORK_INTERFACE_RECHECK_INTERVAL**

None

archivebox.machine.models.**BINARY_RECHECK_INTERVAL**

None

archivebox.machine.models.**PID_REUSE_WINDOW**

'timedelta(...)'

archivebox.machine.models.**PROCESS_TIMEOUT_GRACE**

'timedelta(...)'

archivebox.machine.models.**START_TIME_TOLERANCE**

5.0

archivebox.machine.models.**PROCESS_PID_NAMESPACE_KEY**

'_ARCHIVEBOX_PID_NAMESPACE'

archivebox.machine.models.**get_current_pid_namespace**() → str

Return the OS PID namespace that makes numeric PIDs meaningful.

archivebox.machine.models.**_default_exit_code_for_unowned_process**(*process_type: str*) → int

archivebox.machine.models.**_find_existing_binary_for_reference**(*machine: Machine*, *reference: str*) → *Binary* | None

archivebox.machine.models.**_canonical_binary_name**(*name: Any*) → str

archivebox.machine.models.**_get_process_binary_env_keys**(*plugin_name: str*, *hook_path: str*, *env: dict[str, Any] | None*) → list[str]

archivebox.machine.models.**_sanitize_machine_config**(*config: dict[str, Any] | None*, ** (Keyword-only parameters separator (PEP 3102))*, *lib_dir: str | pathlib.Path | None = None*) → dict[str, Any]

Validate `Machine.config` in place.

Drops stale `*_BINARY` overrides whose path no longer exists or whose path falls outside of `ABXPKG_LIB_DIR` (so a binary uninstall or a lib_dir move clears the override automatically). Non-`_BINARY` keys (`BASE_URL`, `SERVER_SECURITY_MODE`, plugin tunables, etc.) are pass-through — they're arbitrary config overrides and not ours to filter.

class archivebox.machine.models.**MachineManager**

Bases: `django.db.models.Manager`

current() → *archivebox.machine.models.Machine*

class archivebox.machine.models.**Machine**

Bases: *archivebox.base_models.models.ModelWithHealthStats*

id

'CompactUUIDField(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

guid

'CharField(...)'

hostname

'CharField(...)'

hw_in_docker

'BooleanField(...)'

hw_in_vm

'BooleanField(...)'

hw_manufacturer
'CharField(...)'

hw_product
'CharField(...)'

hw_uuid
'CharField(...)'

os_arch
'CharField(...)'

os_family
'CharField(...)'

os_platform
'CharField(...)'

os_release
'CharField(...)'

os_kernel
'CharField(...)'

stats
'JSONField(...)'

config
'JSONField(...)'

num_uses_failed
'PositiveIntegerField(...)'

num_uses_succeeded
'PositiveIntegerField(...)'

objects
'MachineManager(...)'

networkinterface_set: django.db.models.Manager[*NetworkInterface*]
None

class Meta
Bases: *archivebox.base_models.models.ModelWithHealthStats.Meta*

app_label
'machine'

classmethod current_readonly() → *archivebox.machine.models.Machine* | None
Read the current machine without registration, refresh, or config writes.

classmethod current() → *archivebox.machine.models.Machine*

classmethod _sanitize_config(*machine:* archivebox.machine.models.Machine**)** → *archivebox.machine.models.Machine*

to_json() → dict
Convert Machine model instance to a JSON-serializable dict.

static from_json(*record: dict[str, Any]*, *overrides: dict[str, Any] | None = None*)

Update Machine config from JSON dict.

Args: record: JSON dict with 'config': {key: value} patch overrides: Not used

Returns: Machine instance or None

save(**args*, ***kwargs*)

class archivebox.machine.models.**NetworkInterfaceManager**

Bases: django.db.models.Manager

current() → *archivebox.machine.models.NetworkInterface*

class archivebox.machine.models.**NetworkInterface**

Bases: *archivebox.base_models.models.ModelWithHealthStats*

id

'CompactUUIDField(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

machine

'ForeignKey(...)'

mac_address

'CharField(...)'

ip_public

'GenericIPAddressField(...)'

ip_local

'GenericIPAddressField(...)'

dns_server

'GenericIPAddressField(...)'

hostname

'CharField(...)'

iface

'CharField(...)'

isp

'CharField(...)'

city

'CharField(...)'

region

'CharField(...)'

country

'CharField(...)'

objects

'NetworkInterfaceManager(...)'

machine_id: uuid.UUID

None

class Meta

Bases: *archivebox.base_models.models.ModelWithHealthStats.Meta*

app_label

'machine'

unique_together

(('machine', 'ip_public', 'ip_local', 'dns_server'),)

constraints

None

classmethod current(*refresh: bool = False*) → *archivebox.machine.models.NetworkInterface*

class archivebox.machine.models.**BinaryManager**

Bases: models.Manager.from_queryset(PagedQuerySet)

get_from_db_or_cache(*name: str*, *abspath: str = ''*, *version: str = ''*, *sha256: str = ''*, *binprovider: str = 'env'*) → *archivebox.machine.models.Binary*

Get or create an Binary record from the database or cache.

get_valid_binary(*name: str*, *machine:* archivebox.machine.models.Machine | *None = None*) → *archivebox.machine.models.Binary* | None

Get a valid Binary for the given name on the current machine, or None if not found.

class archivebox.machine.models.**Binary**

Bases: *archivebox.base_models.models.ModelWithHealthStats*, *archivebox.workers.models.ModelWithQueue*

Tracks a binary on a specific machine.

Simple queue lifecycle with 2 states:

- queued: Binary needs to be installed
- installed: Binary installed successfully (abspath, version, sha256 populated)

Installation is synchronous during queued→installed transition. If installation fails, Binary stays in queued with retry_at set for later retry.

BinaryService claims queued rows with a conditional update, then run() emits an abxpkg BinaryRequestEvent and persists the resolved installation. The database row is the only lifecycle state; there is intentionally no second in-memory state machine to reconcile after a worker interruption.

class StatusChoices

Bases: django.db.models.TextChoices

QUEUED

('queued', 'Queued')

INSTALLED

('installed', 'Installed')

id

'CompactUUIDField(…)'

created_at

'DateTimeField(…)'

modified_at

'DateTimeField(…)'

machine

'ForeignKey(…)'

name

'CharField(…)'

binproviders

'CharField(…)'

overrides

'JSONField(…)'

binprovider

'CharField(…)'

abspath

'CharField(…)'

version

'CharField(…)'

sha256

'CharField(…)'

status

'StatusField(…)'

retry_at

'RetryAtField(…)'

num_uses_failed

'PositiveIntegerField(…)'

num_uses_succeeded

'PositiveIntegerField(…)'

machine_id: uuid.UUID

None

INITIAL_STATE

None

ACTIVE_STATE

None

FINAL_STATES

()

FINAL_OR_ACTIVE_STATES

()

active_state: str

None

warn_on_save_outside_runner

False

objects

'BinaryManager(...)'

class Meta

Bases: *archivebox.base_models.models.ModelWithHealthStats.Meta*, *archivebox.workers.models.ModelWithQueue.Meta*

app_label

'machine'

verbose_name

'Binary'

verbose_name_plural

'Binaries'

unique_together

(('machine', 'name', 'abspath', 'version', 'sha256'),)

__str__() → str

property is_valid: bool

A binary is valid if it has a resolved path and is marked installed.

property can_install: bool

binary_info() → dict

Return info about the binary.

property output_dir: pathlib.Path

Get output directory for this binary's hook logs. Path: data/machines/{machine_uuid}/binaries/{binary_name}/{binary_uuid

to_json() → dict

Convert Binary model instance to a JSON-serializable dict.

static from_json(*record: dict[str, Any]*, *overrides: dict[str, Any] | None = None*)

Create/update Binary from JSON dict.

Handles two cases:

1. From binaries.json: creates queued binary with name, binproviders, overrides
2. From hook output: updates binary with abspath, version, sha256, binprovider

Args: record: JSON dict with 'name' and either: - 'binproviders', 'overrides' (from binaries.json) - 'abspath', 'version', 'sha256', 'binprovider' (from hook output) overrides: Not used

Returns: Binary instance or None

_allowed_binproviders() → set[str] | None

Return the allowed binproviders for this binary, or None for wildcard.

run()

Execute binary installation through the ArchiveBox binary runner.

install() → bool

Run one synchronous installation attempt for a claimed Binary.

advance_lifecycle() → bool

Advance the explicit binary lifecycle after its queue row is claimed.

install_claimed(*, *lock_seconds: int = 600*) → bool

cleanup()

Clean up background binary installation hooks.

Called after an installation attempt if needed (not typically used for binaries since installations are foreground, but included for consistency).

symlink_to_lib_bin(*lib_bin_dir: str | pathlib.Path*) → pathlib.Path | None

Symlink this binary into a derived lib/bin directory for human-facing convenience.

After a binary is installed by any binprovider (pip, npm, brew, apt, etc), we can optionally expose a flat convenience directory for shell users. ArchiveBox/abx-dl runtime lookup must use the provider-specific ABXPKG_LIB_DIR paths, not this indirection.

Args: lib_bin_dir: Path to the derived convenience bin dir (e.g., /data/lib/bin)

Returns: Path to the created symlink, or None if symlinking failed

Example: >>> binary = Binary.objects.get(name='yt-dlp') >>> binary.symlink_to_lib_bin('/data/lib/arm64-darwin/bin') Path('/data/lib/arm64-darwin/bin/yt-dlp')

symlink_to_lib_bin_after_commit(*lib_bin_dir: str | pathlib.Path*) → None

Symlink after the current DB transaction commits.

Binary rows are projections of provider/hook state and are allowed to be updated directly, but filesystem writes must not run while an outer transaction is still open. Refetch after commit so the symlink points at the committed row, not a possibly-rolled-back in-memory value.

class archivebox.machine.models.**ProcessManager**

Bases: models.Manager.from_queryset(PagedQuerySet)

Manager for Process model.

current() → *archivebox.machine.models.Process*

Get the Process record for the current OS process.

get_by_pid(*pid: int*, *machine:* archivebox.machine.models.Machine | *None = None*) → *archivebox.machine.models.Process* | None

Find a Process by PID with proper validation against PID reuse.

IMPORTANT: PIDs are reused by the OS! This method:

1. Filters by machine (required - PIDs are only unique per machine)
2. Filters by time window (processes older than 24h are stale)
3. Validates via psutil that start times match

Args: pid: OS process ID machine: Machine instance (defaults to current machine)

Returns: Process if found and validated, None otherwise

create_for_archiveresult(*archiveresult*, ***kwargs*)

Create a Process record for an ArchiveResult.

Called during migration and when creating new ArchiveResults.

class archivebox.machine.models.**Process**

Bases: *archivebox.base_models.models.ModelWithDeleteAfter*, django.db.models.Model

Tracks a single OS process execution.

Process represents the actual subprocess spawned to execute a hook. One Process can optionally be associated with an ArchiveResult (via OneToOne), but Process can also exist standalone for internal operations.

Follows the unified process lifecycle:

- queued: Process ready to launch
- running: Process actively executing
- exited: Process completed (check exit_code for success/failure)

Direct Process methods and the abx-dl event projector own these transitions. Keeping the DB row as the only lifecycle state makes interrupted subprocess recovery observable without reconciling a second in-memory state machine.

class StatusChoices

Bases: django.db.models.TextChoices

QUEUED

('queued', 'Queued')

RUNNING

('running', 'Running')

EXITED

('exited', 'Exited')

class TypeChoices

Bases: django.db.models.TextChoices

SUPERVISORD

('supervisord', 'Supervisord')

ORCHESTRATOR

('orchestrator', 'Orchestrator')

SERVER

('server', 'Server')

UPDATE

('update', 'Update')

ADD

('add', 'Add')

SEARCH

('search', 'Search')

WORKER

('worker', 'Worker')

CLI

('cli', 'CLI')

HOOK

('hook', 'Hook')

BINARY

('binary', 'Binary')

id

'CompactUUIDField(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

machine

'ForeignKey(...)'

parent

'ForeignKey(...)'

process_type

'CharField(...)'

worker_type

'CharField(...)'

pwd

'CharField(...)'

cmd

'JSONField(...)'

env

'JSONField(...)'

timeout

'IntegerField(...)'

pid

'IntegerField(...)'

exit_code

'IntegerField(...)'

stdout

'TextField(...)'

stderr

'TextField(...)'

started_at

'DateTimeField(...)'

ended_at

'DateTimeField(...)'

binary

'ForeignKey(...)'

iface

'ForeignKey(...)'

url

'URLField(...)'

status

'CharField(...)'

retry_at

'DateTimeField(...)'

machine_id: uuid.UUID

None

parent_id: uuid.UUID | None

None

binary_id: uuid.UUID | None

None

children: django.db.models.Manager[*archivebox.machine.models.Process*]

None

archiveresult: *archivebox.core.models.ArchiveResult*

None

delete_after_final_statuses

()

objects

'ProcessManager(...)'

class Meta

Bases: *archivebox.base_models.models.ModelWithDeleteAfter.Meta*

app_label

'machine'

verbose_name

'Process'

verbose_name_plural

'Processes'

indexes

None

constraints

None

__str__() → str

get_delete_after_config_value()

classmethod missing_delete_at_candidates()

property cmd_version: str

Get version from associated binary.

property bin_abspath: str

Get absolute path from associated binary.

property plugin: str

Get plugin name from associated ArchiveResult (if any).

property hook_name: str

Get hook name from associated ArchiveResult (if any).

to_json() → dict

Convert Process model instance to a JSON-serializable dict.

hydrate_binary_from_context(*, *plugin_name: str = ''*, *hook_path: str = ''*) → *archivebox.machine.models.Binary* | None

classmethod parse_records_from_text(*text: str*) → list[dict]

Parse JSONL records from raw text using the shared JSONL parser.

get_records() → list[dict]

Parse JSONL records from this process's stdout.

static from_json(*record: dict[str, Any]*, *overrides: dict[str, Any] | None = None*)

Create/update Process from JSON dict.

Args: record: JSON dict with 'id' or process details overrides: Optional dict of field overrides

Returns: Process instance or None

safe_update(*update_fields: dict[str, Any]*, *, *refresh: bool = True*, *extra_filter: dict[str, Any] | None = None*) → bool

Compare-and-swap update for short Process scheduler writes.

Process is not a ModelWithQueue subclass, but its scheduler methods still need the same modified_at CAS behavior as Crawl/Snapshot/Binary without falling back to save().

update_and_requeue(**kwargs*) → bool

Scheduler-facing wrapper around safe_update().

mark_running(*, *process_type: str | None = None*, *pwd: str | pathlib.Path | None = None*, *url: str | None = None*, *worker_type: str = ''*, *timeout: int | None = None*) → None

Record the current process role without changing ownership state elsewhere.

heartbeat() → None

Keep a long-lived watcher visible in recent-process monitoring.

mark_exited(*, *exit_code: int = 0*) → None

Mark a foreground/internal process row exited after command cleanup.

classmethod current_readonly() → *archivebox.machine.models.Process* | None

Return this process's registered identity without registration or probes.

classmethod current() → *archivebox.machine.models.Process*

Get or create the Process record for the current OS process.

Similar to Machine.current(), this:

1. Checks cache for existing Process with matching PID
2. Validates the cached Process is still valid (PID not reused)
3. Creates new Process if needed

IMPORTANT: Uses psutil to validate PID hasn't been reused. PIDs are recycled by OS, so we compare start times.

classmethod _find_parent_process(*machine:* archivebox.machine.models.Machine | *None* = *None*) → *archivebox.machine.models.Process* | None

Find the parent Process record by looking up PPID.

IMPORTANT: Validates against PID reuse by checking:

1. Same machine (PIDs are only unique per machine)
2. Start time matches OS process start time
3. Process is still RUNNING and recent

Returns None if parent is not an ArchiveBox process.

classmethod _detect_process_type() → str

Detect the type of the current process from sys.argv.

`archivebox add --bg` is a fire-and-forget queue write — it does not run the runner or own the runtime stack — so it's classified as CLI instead of ADD. The ADD process_type is reserved for the foreground `archivebox add` flow that actually takes over the runtime stack via `current_command(TypeChoices.ADD, ...)`. Misclassifying `--bg` as ADD makes `runtime_stack_owner` treat it as a newer stack owner for the few seconds it's alive, knocks the running `archivebox server` out of leadership, and triggers a supervisord tear-down + respawn cycle (~5s of dead time per add). Detecting bg here at insert time avoids any race window where the row briefly exists as ADD before a higher-level demotion.

classmethod cleanup_stale_running(*machine:* archivebox.machine.models.Machine | *None* = *None*) → int

Mark stale RUNNING processes as EXITED in the DB.

Processes are stale if:

- Status is RUNNING but OS process no longer exists
- A bounded HOOK or BINARY process exceeded its timeout plus a small grace margin
- Status is RUNNING but started_at is older than PID_REUSE_WINDOW

Returns count of processes cleaned up.

property root: ***archivebox.machine.models.Process***

Get the root process (CLI command) of this hierarchy.

property ancestors: **list[*archivebox.machine.models.Process*]**

Get all ancestor processes from parent to root.

property depth: **int**

Get depth in the process tree (0 = root).

get_descendants(*include_self: bool = False*)

Get all descendant processes recursively.

property shares_pid_namespace: bool

property proc: psutil.Process | None

Get validated psutil.Process for this record.

Returns psutil.Process ONLY if:

1. Process with this PID exists in OS
2. OS process start time matches our started_at (within tolerance)
3. Process is on current machine

Returns None if:

- PID doesn't exist (process exited)
- PID was reused by a different process (start times don't match)
- We're on a different machine than where process ran
- psutil is not available

This prevents accidentally matching a stale/recycled PID.

property is_running: bool

Check if process is currently running via psutil.

More reliable than checking status field since it validates the actual OS process exists and matches our record.

is_alive() → bool

Alias for is_running, for compatibility with subprocess.Popen API.

get_memory_info() → dict | None

Get memory usage if process is running.

get_cpu_percent() → float | None

Get CPU usage percentage if process is running.

get_children_pids() → list[int]

Get PIDs of child processes from OS (not DB).

property stdout_file: pathlib.Path | None

Path to stdout log.

property stderr_file: pathlib.Path | None

Path to stderr log.

property hook_script_name: str | None

Best-effort hook filename extracted from the process command.

property runtime_dir: pathlib.Path | None

Directory where this process stores runtime stdout/stderr logs.

tail_stdout(*lines: int = 50*, *follow: bool = False*)

Tail stdout log file (like `tail` or `tail -f`).

Args: lines: Number of lines to show (default 50) follow: If True, follow the file and yield new lines as they appear

Yields: Lines from stdout

tail_stderr(*lines: int = 50*, *follow: bool = False*)

Tail stderr log file (like `tail` or `tail -f`).

Args: lines: Number of lines to show (default 50) follow: If True, follow the file and yield new lines as they appear

Yields: Lines from stderr

pipe_stdout(*lines: int = 10*, *follow: bool = True*)

Pipe stdout to sys.stdout.

Args: lines: Number of initial lines to show follow: If True, follow the file and print new lines as they appear

pipe_stderr(*lines: int = 10*, *follow: bool = True*)

Pipe stderr to sys.stderr.

Args: lines: Number of initial lines to show follow: If True, follow the file and print new lines as they appear

ensure_log_files() → None

Ensure stdout/stderr log files exist for this process.

_build_env() → dict

Build environment dict for subprocess, merging stored env with system.

launch(*background: bool = False*, *cwd: str | None = None*) → *archivebox.machine.models.Process*

Spawn the subprocess and update this Process record.

Args: background: If True, don't wait for completion (for daemons/bg hooks) cwd: Working directory for the subprocess (defaults to self.pwd)

Returns: self (updated with pid, started_at, etc.)

kill(*signal_num: int = 15*) → bool

Kill this process and update status.

Uses self.proc for safe killing - only kills if PID matches our recorded process (prevents killing recycled PIDs).

Args: signal_num: Signal to send (default SIGTERM=15)

Returns: True if killed successfully, False otherwise

poll() → int | None

Check if process has exited and update status if so.

Cleanup when process exits:

- Copy stdout/stderr to DB (keep files for debugging)
- Delete PID file

Returns: exit_code if exited, None if still running

wait(*timeout: int | None = None*) → int

Wait for process to exit, polling periodically.

Args: timeout: Max seconds to wait (None = use self.timeout)

Returns: exit_code

Raises: TimeoutError if process doesn't exit in time

terminate(*graceful_timeout: float = 5.0*) → bool

Gracefully terminate process: SIGTERM → wait → SIGKILL.

This consolidates SIGTERM/SIGKILL logic used by:

- workers/management/commands/runner_watch.py
- workers/pid_utils.py stop_worker()
- supervisord_util.py stop_existing_supervisord_process()

Args: graceful_timeout: Seconds to wait after SIGTERM before SIGKILL

Returns: True if process was terminated, False if already dead

kill_tree(*graceful_timeout: float = 2.0*) → int

Kill this process and all its children (OS children, not DB children) in parallel.

Uses parallel polling approach - sends SIGTERM to all processes at once, then polls all simultaneously with individual deadline tracking.

This consolidates child-killing logic used by:

- core/takeover_util.py
- supervisord_util.py stop_existing_supervisord_process()

Args: graceful_timeout: Seconds to wait after SIGTERM before SIGKILL

Returns: Number of processes killed (including self)

kill_children_db() → int

Kill all DB-tracked child processes (via parent FK).

Different from kill_tree() which uses OS children. This kills processes created via Process.create(parent=self).

Returns: Number of child Process records killed

classmethod get_running(*process_type: str | None = None*, *machine:* archivebox.machine.models.Machine *| None = None*) → django.db.models.QuerySet[*archivebox.machine.models.Process*]

Get all running processes, optionally filtered by type.

Replaces:

- workers/pid_utils.py get_all_worker_pids()
- workers/orchestrator.py get_total_worker_count()

Args: process_type: Filter by TypeChoices (e.g., 'worker', 'hook') machine: Filter by machine (defaults to current)

Returns: QuerySet of running Process records

classmethod get_running_count(*process_type: str | None = None*, *machine:* archivebox.machine.models.Machine | *None = None*) → int

Get count of running processes.

Replaces:

- workers/pid_utils.py get_running_worker_count()

classmethod stop_all(*process_type: str | None = None*, *machine:* archivebox.machine.models.Machine | *None = None*, *graceful: bool = True*) → int

Stop all running processes of a given type.

Args: process_type: Filter by TypeChoices machine: Filter by machine graceful: If True, use terminate() (SIGTERM→SIGKILL), else kill()

Returns: Number of processes stopped

classmethod get_next_worker_id(*process_type: str = 'worker'*, *machine:* archivebox.machine.models.Machine | *None = None*) → int

Get the next available worker ID for spawning new workers.

Replaces workers/pid_utils.py get_next_worker_id(). Simply returns count of running workers of this type.

Args: process_type: Worker type to count machine: Machine to scope query

Returns: Next available worker ID (0-indexed)

classmethod cleanup_orphaned_chrome() → int

Kill orphaned Chrome processes using chrome_utils.js killZombieChrome.

Scans DATA_DIR for chrome/*.pid files from stale crawls (>5 min old) and kills any orphaned Chrome processes.

Called by:

- Orchestrator on startup (cleanup from previous crashes)
- Orchestrator periodically (every N minutes)

Returns: Number of zombie Chrome processes killed

classmethod cleanup_orphaned_workers() → int

Mark orphaned worker/hook processes as EXITED in the DB.

Orphaned if:

- Root (orchestrator/cli) is not running, or
- No orchestrator/cli ancestor exists.

Standalone worker runs (archivebox run –snapshot-id) are allowed.

archivebox.machine.env_util

Module Contents

Functions

stringify_env_value
is_redacted_env_key
redact_env

continues on next page

Table 8 – continued from previous page

env_to_dotenv_text
env_to_shell_exports

Data

SENSITIVE_ENV_KEY_PARTS
SHELL_ENV_KEY_RE

API

archivebox.machine.env_util.**SENSITIVE_ENV_KEY_PARTS**
('KEY', 'TOKEN', 'SECRET')

archivebox.machine.env_util.**SHELL_ENV_KEY_RE**
'compile(...)'

archivebox.machine.env_util.**stringify_env_value**(*value: Any*) → str

archivebox.machine.env_util.**is_redacted_env_key**(*key: str*) → bool

archivebox.machine.env_util.**redact_env**(*env: dict[str, Any] | None*) → dict[str, Any]

archivebox.machine.env_util.**env_to_dotenv_text**(*env: dict[str, Any] | None*) → str

archivebox.machine.env_util.**env_to_shell_exports**(*env: dict[str, Any] | None*) → str

archivebox.machine.admin

Module Contents

Classes

MachineAdmin
NetworkInterfaceAdmin
BinaryAdmin
ProcessAdmin

Functions

_render_copy_block
_format_process_duration_seconds
register_admin

API

archivebox.machine.admin.**_render_copy_block**(*text: str*, *, *multiline: bool = False*)

archivebox.machine.admin.**_format_process_duration_seconds**(*started_at*, *ended_at*) → str

class archivebox.machine.admin.**MachineAdmin**

Bases: *archivebox.base_models.admin.ConfigEditorMixin*, *archivebox.base_models.admin.BaseModelAdmin*

list_display

('id_display', 'created_at', 'hostname', 'ips', 'os_platform', 'hw_in_docker', 'hw_in_vm', 'hw_manuf…

sort_fields

('id', 'created_at', 'hostname', 'ips', 'os_platform', 'hw_in_docker', 'hw_in_vm', 'hw_manufacturer'…

search_fields

('id', 'guid', 'hostname', 'networkinterface__ip_public', 'networkinterface__ip_local', 'hw_manufact…

readonly_fields

('guid', 'created_at', 'modified_at', 'ips')

fieldsets

(('Identity',), ('Hardware',), ('Operating System',), ('Statistics',), ('Configuration',), ('Timesta…

list_filter

('hw_in_docker', 'hw_in_vm', 'os_arch', 'os_family', 'os_platform')

ordering

('-created_at',)

list_per_page

100

actions

('delete_selected',)

ips(*machine*)

health_display(*obj*)

id_display(*machine*)

class archivebox.machine.admin.**NetworkInterfaceAdmin**

Bases: *archivebox.base_models.admin.BaseModelAdmin*

list_display

('id', 'created_at', 'machine_info', 'ip_public', 'dns_server', 'isp', 'country', 'region', 'city', …

sort_fields

('id', 'created_at', 'machine_info', 'ip_public', 'dns_server', 'isp', 'country', 'region', 'city', …

search_fields

('id', 'machine__id', 'iface', 'ip_public', 'ip_local', 'mac_address', 'dns_server', 'hostname', 'is…

readonly_fields

('machine', 'created_at', 'modified_at', 'mac_address', 'ip_public', 'ip_local', 'dns_server')

fieldsets

(('Machine',), ('Network',), ('Location',), ('Usage',), ('Timestamps',))

list_filter

('isp', 'country', 'region')

ordering

('-created_at',)

list_per_page

100

actions

('delete_selected',)

machine_info(*iface*)

health_display(*obj*)

class archivebox.machine.admin.**BinaryAdmin**

Bases: *archivebox.base_models.admin.BaseModelAdmin*

list_display

('id', 'created_at', 'machine_info', 'name', 'binprovider', 'version', 'abspath', 'sha256', 'status'…

sort_fields

('id', 'created_at', 'machine_info', 'name', 'binprovider', 'version', 'abspath', 'sha256', 'status'…

search_fields

('id', 'machine__id', 'name', 'binprovider', 'version', 'abspath', 'sha256')

readonly_fields

('created_at', 'modified_at', 'output_dir')

fieldsets

(('Binary Info',), ('Location',), ('Version',), ('State',), ('Usage',), ('Timestamps',))

list_filter

('name', 'binprovider', 'status', 'machine_id')

ordering

('-created_at',)

list_per_page

100

actions

('delete_selected',)

machine_info(*binary*)

health_display(*obj*)

class archivebox.machine.admin.**ProcessAdmin**

Bases: *archivebox.base_models.admin.BaseModelAdmin*

list_display

('id', 'created_at', 'machine_info', 'archiveresult_link', 'snapshot_link', 'crawl_link', 'cmd_str',…

sort_fields

('id', 'created_at', 'machine_info', 'archiveresult_link', 'snapshot_link', 'crawl_link', 'cmd_str',…

search_fields

('id', 'machine__id', 'binary__name', 'cmd', 'pwd', 'stdout', 'stderr')

readonly_fields
('created_at', 'modified_at', 'machine', 'binary_link', 'iface_link', 'archiveresult_link', 'snapsho…

fieldsets
(('Process Info',), ('Command',), ('Execution',), ('Timing',), ('Output',), ('Timestamps',))

list_filter
('status', 'exit_code', 'machine_id')

ordering
('-created_at',)

list_per_page
100

actions
('kill_processes', 'delete_selected')

change_actions
('kill_process',)

get_queryset(*request*)

_terminate_processes(*request*, *processes*)

kill_processes(*request*, *queryset*)

kill_process(*request*, *obj*)

machine_info(*process*)

binary_info(*process*)

binary_link(*process*)

iface_link(*process*)

archiveresult_link(*process*)

snapshot_link(*process*)

crawl_link(*process*)

cmd_str(*process*)

status_badge(*process*)

duration_display(*process*)

output_summary(*process*)

cmd_display(*process*)

env_display(*process*)

stdout_display(*process*)

stderr_display(*process*)

archiveresult_output_display(*process*)

archivebox.machine.admin.**register_admin**(*admin_site*)

archivebox.progressmonitor

Submodules

archivebox.progressmonitor.apps

Module Contents

Classes

ProgressMonitorConfig

API

class archivebox.progressmonitor.apps.**ProgressMonitorConfig**

Bases: `django.apps.AppConfig`

name

'archivebox.progressmonitor'

label

'progressmonitor'

archivebox.progressmonitor.collection

Cached display totals for progress polling and the public snapshot list.

Missing/stale statistics are intentional: returning useful rows and progress quickly matters more than exact initial counts. Each viewer may poll once per second; repeating aggregates per request would consume the small server's CPU and hold SQLite read locks needed by the archiving runner. Callers must accept None on a cold cache and must never wait for the refresh or count synchronously.

Background reads still take normal SQLite read locks. Existing indexes, short batches where needed, and throttled refreshes limit their cost; moving work to a thread does not make a scan free or require switching the database to WAL.

Module Contents

Functions

crawl_summary	Sample sealed snapshot metadata without adding table scans to polls.
collection_summary	Return cached totals immediately and, when due, start one refresh.

Data

_refresh_lock

API

archivebox.progressmonitor.collection.**_refresh_lock**

'Lock(...)'

archivebox.progressmonitor.collection.**crawl_summary**(*crawl_id*, *, *snapshot_id=None*)

Sample sealed snapshot metadata without adding table scans to polls.

archivebox.progressmonitor.collection.**collection_summary**(*user*, *, *public=False*)

Return cached totals immediately and, when due, start one refresh.

Public pages need no extra AJAX/polling for these totals: an ordinary later page load reads the populated cache. Keep this best-effort display cache separate from pagination/authorization decisions, which need actual rows.

archivebox.progressmonitor.views

Module Contents

Functions

progress_endpoint	Return the canonical same-origin progress endpoint for monitor embeds.
_live_progress_plugin_names	
live_progress_view	Simple JSON endpoint for live progress status - used by admin progress monitor.

API

archivebox.progressmonitor.views.**progress_endpoint**(*scope: Literal[crawl, snapshot] | None = None*, *object_id: object | None = None*) → str

Return the canonical same-origin progress endpoint for monitor embeds.

archivebox.progressmonitor.views.**_live_progress_plugin_names**() → tuple[frozenset[str], frozenset[str]]

archivebox.progressmonitor.views.**live_progress_view**(*request*, *, *authorized_snapshot=None*)

Simple JSON endpoint for live progress status - used by admin progress monitor.

archivebox.ldap

LDAP authentication module for ArchiveBox.

This module provides native LDAP authentication support using django-auth-ldap. It only activates if:

1. LDAP_ENABLED=True in config
2. Required LDAP libraries (python-ldap, django-auth-ldap) are installed

To install LDAP dependencies: pip install archivebox[ldap]

Or manually: apt install build-essential python3-dev libsasl2-dev libldap2-dev libssl-dev pip install python-ldap django-auth-ldap

Submodules

archivebox.ldap.apps

Django app configuration for LDAP authentication.

Module Contents

Classes

LDAPConfig	Django app config for LDAP authentication.

API

class archivebox.ldap.apps.**LDAPConfig**

Bases: `django.apps.AppConfig`

Django app config for LDAP authentication.

default_auto_field

'django.db.models.BigAutoField'

name

'archivebox.ldap'

verbose_name

'LDAP Authentication'

archivebox.ldap.auth

LDAP authentication backend for ArchiveBox.

This module extends django-auth-ldap to support the LDAP_CREATE_SUPERUSER flag.

Module Contents

Classes

ArchiveBoxLDAPBackend	Custom LDAP authentication backend for ArchiveBox.

API

class archivebox.ldap.auth.**ArchiveBoxLDAPBackend**

Bases: `archivebox.ldap.auth.BaseLDAPBackend`

Custom LDAP authentication backend for ArchiveBox.

Extends django-auth-ldap's LDAPBackend to support:

- LDAP_CREATE_SUPERUSER: Automatically grant superuser privileges to LDAP users

authenticate_ldap_user(*ldap_user*, *password*)

Authenticate using LDAP and optionally grant superuser privileges.

This method is called by django-auth-ldap after successful LDAP authentication.

archivebox.misc

Submodules

archivebox.misc.paginators

Module Contents

Classes

AcceleratedPaginator	Paginator that accepts exact count hints for already-optimized query paths.

API

class archivebox.misc.paginators.**AcceleratedPaginator**

Bases: `django.core.paginator.Paginator`

Paginator that accepts exact count hints for already-optimized query paths.

count()

archivebox.misc.toml_util

Module Contents

Classes

CustomTOMLEncoder	Custom TomlEncoder to work around https://github.com/uiri/toml's many encoding bugs.

Functions

better_toml_dump_str

API

archivebox.misc.toml_util.**better_toml_dump_str**(*val: Any*) → str

class archivebox.misc.toml_util.**CustomTOMLEncoder**(***kwargs*)

Bases: `toml.encoder.TomlEncoder`

Custom TomlEncoder to work around https://github.com/uiri/toml's many encoding bugs.

```
toml.dumps(value, encoder=CustomTOMLEncoder())
```

Initialization

archivebox.misc.system

Module Contents

Functions

atomic_write	Safe atomic write to filesystem by writing to temp file + atomic rename
get_dir_size	get the total disk size of a given directory, optionally summing up recursively and limiting to a given filter list

API

archivebox.misc.system.**atomic_write**(*path: pathlib.Path | str*, *contents: dict | str | bytes*, *overwrite: bool = True*, *config=None*, ***config_kwargs*) → None

Safe atomic write to filesystem by writing to temp file + atomic rename

archivebox.misc.system.**get_dir_size**(*path: str | pathlib.Path*, *recursive: bool = True*, *pattern: str | None = None*) → tuple[int, int, int]

get the total disk size of a given directory, optionally summing up recursively and limiting to a given filter list

archivebox.misc.shell_welcome_message

archivebox.misc.logging

Module Contents

Classes

AttrDict	
RainbowHighlighter	

Functions

stdout	
stderr	
hint	

Data

CONSOLE	
STDERR	
IS_TTY	
rainbow	
DEFAULT_CLI_COLORS	
ANSI	
COLOR_DICT	

API

archivebox.misc.logging.**CONSOLE**

'Console(…)'

archivebox.misc.logging.**STDERR**

'Console(…)'

archivebox.misc.logging.**IS_TTY**

'isatty(…)'

class archivebox.misc.logging.**AttrDict**(**args*, ***kwargs*)

Bases: dict

classmethod _wrap(*value*)

__setitem__(*key*, *value*)

update(**args*, ***kwargs*)

__getattr__(*key*)

class archivebox.misc.logging.**RainbowHighlighter**

Bases: rich.highlighter.Highlighter

highlight(*text*)

archivebox.misc.logging.**rainbow**

'RainbowHighlighter(…)'

archivebox.misc.logging.**DEFAULT_CLI_COLORS**

'AttrDict(…)'

archivebox.misc.logging.**ANSI**

'AttrDict(…)'

archivebox.misc.logging.**COLOR_DICT**

'defaultdict(…)'

archivebox.misc.logging.**stdout**(**args*, *color: str | None = None*, *prefix: str = ''*, *config: dict | None = None*) → None

archivebox.misc.logging.**stderr**(**args*, *color: str | None = None*, *prefix: str = ''*, *config: dict | None = None*) → None

archivebox.misc.logging.**hint**(*text: tuple[str, ...] | list[str] | str*, *prefix=' '*, *config: dict | None = None*) → None

archivebox.misc.logging_util

Module Contents

Classes

TimedProgress	Show a progress bar and measure elapsed time until .end() is called

Functions

progress_bar	show timer in the form of progress bar, with percentage and seconds remaining
log_list_started	
log_list_finished	
log_removal_started	
log_removal_finished	
pretty_path	convert paths like .../ArchiveBox/archivebox/../output/abc into output/abc
printable_filesize	
format_duration	Format duration in human-readable form.
truncate_url	Truncate URL to max_length, keeping domain and adding ellipsis.
log_worker_event	Log a worker event with structured metadata and indentation.
printable_folders	
printable_config	
printable_folder_status	
printable_dependency_version	

API

class archivebox.misc.logging_util.**TimedProgress**(*seconds*, *prefix=''*, *config=None*, ***config_kwargs*)

Show a progress bar and measure elapsed time until .end() is called

Initialization

end()

immediately end progress, clear the progressbar line, and save end_ts

archivebox.misc.logging_util.**progress_bar**(*seconds: int*, *prefix: str = ''*, *ANSI: dict[str, str] = ANSI*, *config=None*, ***config_kwargs*) → None

show timer in the form of progress bar, with percentage and seconds remaining

archivebox.misc.logging_util.**log_list_started**(*filter_patterns: list[str] | None*, *filter_type: str*)

archivebox.misc.logging_util.**log_list_finished**(*snapshots*)

archivebox.misc.logging_util.**log_removal_started**(*snapshots*, *yes: bool*)

archivebox.misc.logging_util.**log_removal_finished**(*remaining_links: int*, *removed_links: int*)

archivebox.misc.logging_util.**pretty_path**(*path: pathlib.Path | str*, *pwd: pathlib.Path | str = CONSTANTS.DATA_DIR*, *color: bool = True*) → str

convert paths like .../ArchiveBox/archivebox/../output/abc into output/abc

archivebox.misc.logging_util.**printable_filesize**(*num_bytes: int | float*) → str

archivebox.misc.logging_util.**format_duration**(*seconds: float*) → str

Format duration in human-readable form.

archivebox.misc.logging_util.**truncate_url**(*url: str*, *max_length: int = 60*) → str

Truncate URL to max_length, keeping domain and adding ellipsis.

archivebox.misc.logging_util.**log_worker_event**(*worker_type: str*, *event: str*, *indent_level: int = 0*, *pid: int | None = None*, *worker_id: str | None = None*, *url: str | None = None*, *plugin: str | None = None*, *metadata: dict[str, Any] | None = None*, *error: Exception | None = None*) → None

Log a worker event with structured metadata and indentation.

Args: worker_type: Type of worker (Orchestrator, CrawlWorker, SnapshotWorker) event: Event name (Starting, Completed, Failed, etc.) indent_level: Indentation level (0=Orchestrator, 1=CrawlWorker, 2=SnapshotWorker) pid: Process ID worker_id: Worker ID (UUID for workers) url: URL being processed (for SnapshotWorker) plugin: Plugin name (for hook processes) metadata: Dict of metadata to show in curly braces error: Exception if event is an error

archivebox.misc.logging_util.**printable_folders**(*folders: dict[str, Optional[*archivebox.core.models.Snapshot*]]*, *with_headers: bool = False*) → str

archivebox.misc.logging_util.**printable_config**(*config: dict*, *prefix: str = ''*) → str

archivebox.misc.logging_util.**printable_folder_status**(*name: str*, *folder: dict*) → str

archivebox.misc.logging_util.**printable_dependency_version**(*name: str*, *dependency: dict*) → str

archivebox.misc.checks

Module Contents

Functions

_migration_interrupt_message	
_exit_on_migration_interrupt	
check_data_folder	
check_migrations	
check_io_encoding	
check_not_root	
is_archivebox_source_root	Return whether path is the root of an ArchiveBox source checkout.
check_not_inside_source_dir	Refuse data-producing execution from the ArchiveBox source checkout.
check_data_dir_permissions	
check_tmp_dir	
check_lib_dir	

API

archivebox.misc.checks.**_migration_interrupt_message**(*, *before_apply: bool = False*) → str

archivebox.misc.checks.**_exit_on_migration_interrupt**()

archivebox.misc.checks.**check_data_folder**(*config=None*, ***config_kwargs*) → None

archivebox.misc.checks.**check_migrations**(***, *blocking: bool = True*, *auto_apply: bool = False*, *cancel_delay: int = 3*) → list[str]

archivebox.misc.checks.**check_io_encoding**()

archivebox.misc.checks.**check_not_root**()

archivebox.misc.checks.**is_archivebox_source_root**(*path: pathlib.Path | str | None = None*) → bool

Return whether path is the root of an ArchiveBox source checkout.

archivebox.misc.checks.**check_not_inside_source_dir**(*path: pathlib.Path | str | None = None*) → None

Refuse data-producing execution from the ArchiveBox source checkout.

archivebox.misc.checks.**check_data_dir_permissions**(*config=None*, ***config_kwargs*)

archivebox.misc.checks.**check_tmp_dir**(*tmp_dir=None*, *throw=False*, *quiet=False*, *must_exist=True*, *config=None*, ***config_kwargs*)

archivebox.misc.checks.**check_lib_dir**(*lib_dir: pathlib.Path | None = None*, *throw=False*, *quiet=False*, *must_exist=True*, *config=None*, ***config_kwargs*)

archivebox.misc.db

Database utility functions for ArchiveBox.

Post-bootstrap: requires archivebox.config constants and uses Django lazily (`from django.db import ...` inside functions). Not safe to import pre-bootstrap.

Module Contents

Functions

is_postgres	True if this process started with the PostgreSQL database backend.
postgres_db_params	Postgres connection params from config (NAME/USER/PASSWORD/HOST/PORT).
_psycopg_connect	
database_exists	True if this collection's database has been initialized.
database_display_location	Human-readable location of the database (file path or postgres DSN).
ensure_database_ready	Make sure a database server is reachable before running migrations.
approximate_row_counts	Cheap per-table approximate row counts from the backend's optimizer stats.
truncate_overlong_charfields	Clamp a model instance's CharField values to their declared max_length.
rebuild_models_from_migration_state	(non-sqlite only) Drop and recreate the given models' tables from the current migration state.
drop_models_on_postgres	Reverse companion to `rebuild_models_from_migration_state`.
run_db_analyze_batch	Refresh approximate SQLite planner stats for one table per idle tick.

continues on next page

Table 28 – continued from previous page

compact_command	
sqlite_lock_holders	
log_sqlite_lock_holders	
sqlite_lock_error	
retry_sqlite_locks	
migration_lock	
migration_state	Cheaply compare migration files to django_migrations without invoking migrate.
pending_migrations	Return migration files on disk that have not been applied yet.
apply_migrations	Apply pending Django migrations

Data

_IS_POSTGRES
HISTORICAL_GHOST_MIGRATIONS

API

archivebox.misc.db.**_IS_POSTGRES: bool | None**

None

archivebox.misc.db.**is_postgres**(*config: Any | None = None*) → bool

True if this process started with the PostgreSQL database backend.

archivebox.misc.db.**postgres_db_params**() → dict[str, str]

Postgres connection params from config (NAME/USER/PASSWORD/HOST/PORT).

archivebox.misc.db.**_psycopg_connect**(*dbname: str | None = None*, *connect_timeout: int = 5*)

archivebox.misc.db.**database_exists**() → bool

True if this collection's database has been initialized.

sqlite: the index.sqlite3 file exists on disk. postgres: the configured database is reachable and contains the django_migrations table. Safe to call before Django is set up.

archivebox.misc.db.**database_display_location**() → str

Human-readable location of the database (file path or postgres DSN).

archivebox.misc.db.**ensure_database_ready**() → None

Make sure a database server is reachable before running migrations.

sqlite: no-op (the file is created on first connection). postgres: verify the server accepts connections and create the configured database if it does not exist yet. Raises SystemExit with a helpful message if the server is unreachable.

archivebox.misc.db.**approximate_row_counts**(*connection*) → dict[str, int]

Cheap per-table approximate row counts from the backend's optimizer stats.

sqlite: reads sqlite_stat1 (populated by ANALYZE). postgres: reads pg_class.reltuples (maintained by autovacuum/ANALYZE). Returns {} on any failure; tables never analyzed may be absent.

archivebox.misc.db.**truncate_overlong_charfields**(*instance=None*, ***kwargs*) → None

Clamp a model instance's CharField values to their declared max_length.

SQLite never enforces VARCHAR(n) limits, so ArchiveBox has always stored overlong values (e.g. long crawl labels or page titles) untruncated. PostgreSQL enforces them and would raise DataError on save instead. Truncating keeps writes succeeding identically on both backends.

Dual-use: works as a `pre_save` receiver (Django passes `instance=` and `sender=` as kwargs; registered in `CoreConfig.ready()`) and as a plain `truncate_overlong_charfields(obj)` call for `bulk_create` paths, which bypass signals.

archivebox.misc.db.**rebuild_models_from_migration_state**(*apps*, *schema_editor*, *app_label: str*, *model_names: list[str]*) → None

(non-sqlite only) Drop and recreate the given models' tables from the current migration state.

ArchiveBox's historical sqlite migrations rebuild tables with raw SQL that intentionally diverges from Django migration state (state-only AddFields reconciled by later sqlite rebuilds). Postgres support postdates all of them, so a non-sqlite database can never contain legacy data at these points in history: every affected table is empty, and dropping + recreating it from state is always equivalent, keeping the real schema in lockstep with migration state at each divergence point. No-op on sqlite.

archivebox.misc.db.**drop_models_on_postgres**(*apps*, *schema_editor*, *app_label: str*, *model_names: list[str]*) → None

Reverse companion to `rebuild_models_from_migration_state`.

Drops the given models' tables on non-sqlite backends (in the order given, so callers pass reverse-dependency order). No-op on sqlite, whose reverse is handled by the gated `RunSQL` reverse_sql instead.

archivebox.misc.db.**run_db_analyze_batch**(*remaining: list[str] | None*) → list[str]

Refresh approximate SQLite planner stats for one table per idle tick.

A full ANALYZE scans every index while holding a write transaction; moving it into the runner did not prevent it from blocking submissions. Sampling at most about 1000 entries per index keeps maintenance independent of collection size. Approximate stats are sufficient for choosing indexes. PostgreSQL owns its statistics through autovacuum/autoanalyze.

archivebox.misc.db.**compact_command**(*cmdline: list[str] | None*, *fallback: str = ''*) → str

archivebox.misc.db.**sqlite_lock_holders**(*db_path: pathlib.Path = CONSTANTS.DATABASE_FILE*) → list[str]

archivebox.misc.db.**log_sqlite_lock_holders**(*console: Any*, ***, *db_path: pathlib.Path = CONSTANTS.DATABASE_FILE*, *limit: int = 8*) → None

archivebox.misc.db.**sqlite_lock_error**(*error: BaseException*) → bool

archivebox.misc.db.**retry_sqlite_locks**(*action: collections.abc.Callable[[], Any]*, ***, *label: str*, *stderr: TextIO | None = None*) → Any

archivebox.misc.db.**migration_lock**(*stdout: TextIO | None = None*)

archivebox.misc.db.**HISTORICAL_GHOST_MIGRATIONS: frozenset[tuple[str, str]]**

'frozenset(...)'

archivebox.misc.db.**migration_state**(*out_dir: pathlib.Path = CONSTANTS.DATA_DIR*) → tuple[list[str], list[str], dict[str, str]]

Cheaply compare migration files to django_migrations without invoking migrate.

archivebox.misc.db.**pending_migrations**(*out_dir: pathlib.Path = CONSTANTS.DATA_DIR*) → list[str]

Return migration files on disk that have not been applied yet.

archivebox.misc.db.**apply_migrations**(*out_dir: pathlib.Path = CONSTANTS.DATA_DIR*, *stdout: TextIO | None = None*, *stderr: TextIO | None = None*, *verbosity: int = 1*) → list[str]

Apply pending Django migrations

archivebox.misc.serve_static

Module Contents

Classes

_StreamingQueueWriter	Expose a write-only file-like object so zipfile can stream into a queue.
RangedFileReader	Wraps a file like object with an iterator that runs over part (or all) of the file defined by start and stop. Blocks of block_size will be returned from the starting position, up to, but not including the stop point. https://github.com/satchamo/django/commit/2ce75c5c4bee2a858c0214d136bfcd351fcde11d

Functions

add_replay_image_fallback	Add a best-effort responsive-image fallback to the locally served replay worker.
_load_hash_map	
_hash_for_path	
_resolve_archive_path	
_cache_policy	
_format_direntry_timestamp	
_safe_zip_stem	
_iter_visible_files	Yield non-hidden files in a stable order so ZIP output is deterministic.
_build_directory_zip_response	
_stream_ranged_file_async	
_render_directory_index	
_extract_markdown_candidate	
_looks_like_markdown	
_render_text_preview_document	
_render_image_preview_document	
_encoded_responses_image_url	
_index_responses_paths_for_html_images	
_rewrite_html_image_sources_to_responses	
_rewrite_html_image_sources_for_request	
_apply_transformed_html_preview_style	
_set_transformed_response_headers	
_render_markdown_fallback	
_render_markdown_document	
_content_type_base	
_is_risky_replay_document	
_apply_archive_replay_headers	

continues on next page

Table 31 – continued from previous page

_is_asgi_request	
serve_static_with_byterange_support	Overrides Django's built-in django.views.static.serve function to support byte range requests. This allows you to do things like seek into the middle of a huge mp4 or WACZ without downloading the whole file. https://github.com/satchamo/django/commit/2ce75c5c4bee2a858c0214d136bfcd351fcde11d
serve_static	Serve static files below a given point in the directory structure or from locations inferred from the staticfiles finders.
parse_range_header	Parses a range header into a list of two-tuples (start, stop) where `start` is the starting byte of the range (inclusive) and `stop` is the ending byte position of the range (exclusive). Returns None if the value of the header is not syntactically valid. https://github.com/satchamo/django/commit/2ce75c5c4bee2a858c0214d136bfcd351fcde11d

Data

_HASHES_CACHE
DIRECTORY_PREVIEW_MAX_BYTES
FAVICON_CACHE_CONTROL
IMG_SRC_ATTR_RE
TRANSFORMED_HTML_PREVIEW_STYLE
MARKDOWN_INLINE_LINK_RE
MARKDOWN_INLINE_IMAGE_RE
MARKDOWN_BOLD_RE
MARKDOWN_ITALIC_RE
HTML_TAG_RE
HTML_BODY_RE
RISKY_REPLAY_MIMETYPES
RISKY_REPLAY_EXTENSIONS
RISKY_REPLAY_MARKERS

API

archivebox.misc.serve_static.**_HASHES_CACHE: dict[pathlib.Path, tuple[float, dict[str, str]]]**

None

archivebox.misc.serve_static.**DIRECTORY_PREVIEW_MAX_BYTES**

None

archivebox.misc.serve_static.**FAVICON_CACHE_CONTROL**

'public, max-age=31536000, s-maxage=31536000, immutable'

archivebox.misc.serve_static.**IMG_SRC_ATTR_RE**

'compile(…)'

archivebox.misc.serve_static.**TRANSFORMED_HTML_PREVIEW_STYLE = <Multiline-String>**

archivebox.misc.serve_static.**add_replay_image_fallback**(*response: django.http.HttpResponse*) → django.http.HttpResponse

Add a best-effort responsive-image fallback to the locally served replay worker.

WHY: a recorder saves the image Chrome selected, not every srcset candidate. Replaying at another viewport/DPR can select an uncaptured URL (the motivating case was Drive's missing 2x logo with its 1x logo already in the same WACZ). WACZ requests are answered inside Webrecorder's service worker, so a Django 404 handler cannot repair them. Append this adapter when serving that worker; do not change captures, request more live assets, or resize the recorded page.

Only an image GET that already returned 404 is eligible. The worker reads the referring HTML from the SAME archive and tries src/srcset candidates declared on that img or its enclosing picture. It never guesses filename substitutions or borrows an unrelated nearby image. The original response wins on absent metadata, unsupported worker APIs, parse errors, timeout, or failed candidates. This deliberately does not promise to fix JS-only srcsets, CSS backgrounds, corrupt images returning 200, or pages whose archived HTML is too large.

Keep this self-contained in the static-file server: it is a replay repair, independent of the capturing plugin/site. Feature detection makes upstream worker changes disable the repair rather than break ordinary replay. The archive's HTML/ZIP bytes and successful HTTP responses remain untouched.

archivebox.misc.serve_static.**_load_hash_map**(*snapshot_dir: pathlib.Path*) → dict[str, str] | None

archivebox.misc.serve_static.**_hash_for_path**(*document_root: pathlib.Path*, *rel_path: str*) → str | None

archivebox.misc.serve_static.**_resolve_archive_path**(*document_root: str | pathlib.Path*, *rel_path: str*) → tuple[pathlib.Path, str]

archivebox.misc.serve_static.**_cache_policy**(*config=None*, ***config_kwargs*) → str

archivebox.misc.serve_static.**_format_direntry_timestamp**(*stat_result: os.stat_result*) → str

archivebox.misc.serve_static.**_safe_zip_stem**(*name: str*) → str

class archivebox.misc.serve_static.**_StreamingQueueWriter**(*output_queue: queue.Queue[bytes | BaseException | object]*)

Expose a write-only file-like object so zipfile can stream into a queue.

Initialization

write(*data: bytes*) → int

tell() → int

flush() → None

close() → None

writable() → bool

seekable() → bool

archivebox.misc.serve_static.**_iter_visible_files**(*root: pathlib.Path*)

Yield non-hidden files in a stable order so ZIP output is deterministic.

archivebox.misc.serve_static.**_build_directory_zip_response**(*fullpath: pathlib.Path*, *path: str*, ***, *is_archive_replay: bool*, *use_async_stream: bool*, *config=None*) → django.http.StreamingHttpResponse

async archivebox.misc.serve_static.**_stream_ranged_file_async**(*ranged_file:* RangedFileReader)

archivebox.misc.serve_static.**_render_directory_index**(*request*, *path: str*, *fullpath: pathlib.Path*) → django.http.HttpResponse

archivebox.misc.serve_static.**MARKDOWN_INLINE_LINK_RE**
'compile(…)'

archivebox.misc.serve_static.**MARKDOWN_INLINE_IMAGE_RE**
'compile(…)'

archivebox.misc.serve_static.**MARKDOWN_BOLD_RE**
'compile(…)'

archivebox.misc.serve_static.**MARKDOWN_ITALIC_RE**
'compile(…)'

archivebox.misc.serve_static.**HTML_TAG_RE**
'compile(…)'

archivebox.misc.serve_static.**HTML_BODY_RE**
'compile(…)'

archivebox.misc.serve_static.**RISKY_REPLAY_MIMETYPES**
None

archivebox.misc.serve_static.**RISKY_REPLAY_EXTENSIONS**
None

archivebox.misc.serve_static.**RISKY_REPLAY_MARKERS**
('<!doctype html', '<html', '<svg')

archivebox.misc.serve_static.**_extract_markdown_candidate**(*text: str*) → str

archivebox.misc.serve_static.**_looks_like_markdown**(*text: str*) → bool

archivebox.misc.serve_static.**_render_text_preview_document**(*text: str*, *title: str*) → str

archivebox.misc.serve_static.**_render_image_preview_document**(*image_url: str*, *title: str*) → str

archivebox.misc.serve_static.**_encoded_responses_image_url**(*image_url: str*, *page_url: str | None*) → str | None

archivebox.misc.serve_static.**_index_responses_paths_for_html_images**(*snapshot_root: pathlib.Path*, *html_rel_path: str*, *encoded_urls: set[str]*) → dict[str, str]

archivebox.misc.serve_static.**_rewrite_html_image_sources_to_responses**(*html_text: str*, *snapshot_root: pathlib.Path*, *html_rel_path: str*, *page_url: str | None*) → tuple[str, int]

archivebox.misc.serve_static.**_rewrite_html_image_sources_for_request**(*request*, *html_text: str*, *document_root: pathlib.Path | None*, *html_rel_path: str*) → tuple[str, int]

archivebox.misc.serve_static.**_apply_transformed_html_preview_style**(*html_text: str*) → str

archivebox.misc.serve_static.**_set_transformed_response_headers**(*response*, *fullpath: pathlib.Path*, *statobj: os.stat_result*, *encoding: str | None*, *cache_policy: str*) → None

archivebox.misc.serve_static.**_render_markdown_fallback**(*text: str*) → str

archivebox.misc.serve_static.**_render_markdown_document**(*markdown_text: str*) → str

archivebox.misc.serve_static.**_content_type_base**(*content_type: str*) → str

archivebox.misc.serve_static.**_is_risky_replay_document**(*fullpath: pathlib.Path*, *content_type: str*) → bool

archivebox.misc.serve_static.**_apply_archive_replay_headers**(*response: django.http.HttpResponse*, ***, *fullpath: pathlib.Path*, *content_type: str*, *is_archive_replay: bool*, *config=None*, ***config_kwargs*) → django.http.HttpResponse

archivebox.misc.serve_static.**_is_asgi_request**(*request*) → bool

archivebox.misc.serve_static.**serve_static_with_byterange_support**(*request*, *path*, *document_root=None*, *show_indexes=False*, *is_archive_replay: bool = False*)

Overrides Django's built-in django.views.static.serve function to support byte range requests. This allows you to do things like seek into the middle of a huge mp4 or WACZ without downloading the whole file. https://github.com/satchamo/django/commit/2ce75c5c4bee2a858c0214d136bfcd351fcde11d

archivebox.misc.serve_static.**serve_static**(*request*, *path*, ***kwargs*)

Serve static files below a given point in the directory structure or from locations inferred from the staticfiles finders.

To use, put a URL pattern such as::

```
from django.contrib.staticfiles import views

path('<path:path>', views.serve)
```

in your URLconf.

It uses the django.views.static.serve() view to serve the found files.

archivebox.misc.serve_static.**parse_range_header**(*header*, *resource_size*)

Parses a range header into a list of two-tuples (start, stop) where `start` is the starting byte of the range (inclusive) and `stop` is the ending byte position of the range (exclusive). Returns None if the value of the header is not syntactically valid. https://github.com/satchamo/django/commit/2ce75c5c4bee2a858c0214d136bfcd351fcde11d

class archivebox.misc.serve_static.**RangedFileReader**(*file_like*, *start=0*, *stop=float('inf')*, *block_size=None*)

Wraps a file like object with an iterator that runs over part (or all) of the file defined by start and stop. Blocks of block_size will be returned from the starting position, up to, but not including the stop point. https://github.com/satchamo/django/commit/2ce75c5c4bee2a858c0214d136bfcd351fcde11d

Initialization

block_size

8192

__iter__()

close()

archivebox.misc.jsonl

JSONL (JSON Lines) utilities for ArchiveBox.

Provides functions for reading, writing, and processing typed JSONL records. All CLI commands that accept stdin can read both plain URLs and typed JSONL.

CLI Pipeline: archivebox crawl URL -> {"type": "Crawl", "id": "…", "urls": "…", …} archivebox snapshot -> {"type": "Snapshot", "id": "…", "url": "…", …} archivebox extract -> {"type": "ArchiveResult", "id": "…", "snapshot_id": "…", …}

Typed JSONL Format: {"type": "Crawl", "id": "…", "urls": "…", "max_depth": 0, …} {"type": "Snapshot", "id": "…", "url": "https://example.com", "title": "…", …} {"type": "ArchiveResult", "id": "…", "snapshot_id": "…", "plugin": "…", …} {"type": "Tag", "name": "…"}

Plain URLs (also supported): https://example.com https://foo.com

Local filesystem paths and file:// URLs are intentionally not accepted here. Pipe file contents through stdin instead of passing a path/URI as a crawl URL.

Module Contents

Functions

parse_line	Parse a single line of input as either JSONL or plain URL.
read_stdin	Read JSONL or plain URLs from stdin.
read_args_or_stdin	Read from CLI arguments if provided, otherwise from stdin.
write_record	Write a single JSONL record to stdout (or provided stream).
write_records	Stream bulk JSONL output with bounded buffering, returning its row count.

Data

TYPE_SNAPSHOT
TYPE_ARCHIVERESULT

continues on next page

Table 34 – continued from previous page

TYPE_TAG
TYPE_CRAWL
TYPE_BINARYREQUEST
TYPE_BINARY
TYPE_PROCESS
TYPE_MACHINE
VALID_TYPES

API

archivebox.misc.jsonl.**TYPE_SNAPSHOT**

'Snapshot'

archivebox.misc.jsonl.**TYPE_ARCHIVERESULT**

'ArchiveResult'

archivebox.misc.jsonl.**TYPE_TAG**

'Tag'

archivebox.misc.jsonl.**TYPE_CRAWL**

'Crawl'

archivebox.misc.jsonl.**TYPE_BINARYREQUEST**

'BinaryRequest'

archivebox.misc.jsonl.**TYPE_BINARY**

'Binary'

archivebox.misc.jsonl.**TYPE_PROCESS**

'Process'

archivebox.misc.jsonl.**TYPE_MACHINE**

'Machine'

archivebox.misc.jsonl.**VALID_TYPES**

None

archivebox.misc.jsonl.**parse_line**(*line: str*) → dict[str, Any] | None

Parse a single line of input as either JSONL or plain URL.

Returns a dict with at minimum {'type': '...', 'url': '...'} or None if invalid.

archivebox.misc.jsonl.**read_stdin**(*stream: TextIO | None = None*) → collections.abc.Iterator[dict[str, Any]]

Read JSONL or plain URLs from stdin.

Yields parsed records as dicts. Supports both JSONL format and plain URLs (one per line).

archivebox.misc.jsonl.**read_args_or_stdin**(*args: collections.abc.Iterable[str]*, *stream: TextIO | None = None*) → collections.abc.Iterator[dict[str, Any]]

Read from CLI arguments if provided, otherwise from stdin.

Handles both URLs and JSONL from either source. Does not expand local file path arguments; pipe file contents via stdin.

archivebox.misc.jsonl.**write_record**(*record: dict[str, Any]*, *stream: TextIO | None = None*) → None

Write a single JSONL record to stdout (or provided stream).

archivebox.misc.jsonl.**write_records**(*records: collections.abc.Iterable[dict[str, Any]]*, *stream: TextIO | None = None*) → int

Stream bulk JSONL output with bounded buffering, returning its row count.

archivebox.misc.monkey_patches

Module Contents

Classes

ModifiedAccessLogGenerator	Clutge workaround until daphne uses the Python logging framework. https://github.com/django/daphne/pull/473/files

Data

SENSITIVE_QUERY_PARAM_RE

API

archivebox.misc.monkey_patches.**SENSITIVE_QUERY_PARAM_RE**

'compile(...)'

class archivebox.misc.monkey_patches.**ModifiedAccessLogGenerator**

Bases: daphne.access.AccessLogGenerator

Clutge workaround until daphne uses the Python logging framework. https://github.com/django/daphne/pull/473/files

__call__(*protocol*, *action*, *details*)

write_entry(*host*, *date*, *request*, *status=None*, *length=None*, *ident=None*, *user=None*, *time_taken=None*)

archivebox.misc.util

Module Contents

Classes

ExtendedEncoder	Extended json serializer that supports serializing several model fields and objects

Functions

filter_queryset_by_uuid_substring	Filter a queryset to UUID-column matches by prefix or suffix (case-insensitive).
sanitize_html_text	Strip all HTML from user-editable text before storing it.
ts_to_date_str	

continues on next page

Table 38 – continued from previous page

sanitize_extracted_url	Use the standalone parser's quote rules, then trim dangling prose punctuation.
validate_url_length	
validate_url	
parens_are_matched	check that all parentheses in a string are balanced and nested properly
split_comma_separated_urls	
find_all_urls	
parse_filesize_to_bytes	Parse a byte count from an integer or human-readable string like 45mb or 2 GB.
enforce_types	Enforce function arg and kwarg types at runtime using its python3 type hints Simpler version of pydantic @validate_call decorator
docstring	attach the given docstring to the decorated function
parse_date	Parse unix timestamps, iso format, and human-readable strings
download_url	Download the contents of a remote url and return the text
ansi_to_html	Based on: https://stackoverflow.com/questions/19212665/python-converting-ansi-color-codes-to-html Simple way to render colored CLI stdout/stderr in HTML properly, Textual/rich is probably better though.
to_json	Serialize object to JSON string with extended type support

Data

scheme
without_scheme
without_query
without_fragment
without_path
path
basename
domain
query
fragment
extension
base_url
urlencode
urldecode
htmlencode
htmldecode
COLOR_REGEX
MAX_URL_LENGTH
FILESIZE_UNITS

API

archivebox.misc.util.**filter_queryset_by_uuid_substring**(*queryset*, *slug: str*, *field: str = 'id'*)

Filter a queryset to UUID-column matches by prefix or suffix (case-insensitive).

Avoids `id__icontains` (an unindexed full-table scan over the UUID column) by stripping non-hex chars from `slug` and matching with `istartswith` / `iendswith`. Returns an empty queryset for inputs with fewer than 8 hex chars to avoid overly broad matches. A full 32-char hex string falls back to an exact-equality lookup.

archivebox.misc.util.**scheme**

None

archivebox.misc.util.**without_scheme**

None

archivebox.misc.util.**without_query**

None

archivebox.misc.util.**without_fragment**

None

archivebox.misc.util.**without_path**

None

archivebox.misc.util.**path**

None

archivebox.misc.util.**basename**

None

archivebox.misc.util.**domain**

None

archivebox.misc.util.**query**

None

archivebox.misc.util.**fragment**

None

archivebox.misc.util.**extension**

None

archivebox.misc.util.**base_url**

None

archivebox.misc.util.**urlencode**

None

archivebox.misc.util.**urldecode**

None

archivebox.misc.util.**htmlencode**

None

archivebox.misc.util.**htmldecode**

None

archivebox.misc.util.**sanitize_html_text**(*value: Any*) → str

Strip all HTML from user-editable text before storing it.

archivebox.misc.util.**ts_to_date_str**(*ts: Any*) → str | None

archivebox.misc.util.**COLOR_REGEX**

'compile(...)'

archivebox.misc.util.**MAX_URL_LENGTH**

65535

archivebox.misc.util.**FILESIZE_UNITS: dict[str, int]**

None

archivebox.misc.util.**sanitize_extracted_url**(*url: str*) → str

Use the standalone parser's quote rules, then trim dangling prose punctuation.

archivebox.misc.util.**validate_url_length**(*url: str*) → str

archivebox.misc.util.**validate_url**(*url: str*) → str

archivebox.misc.util.**parens_are_matched**(*string: str*, *open_char='('*, *close_char=')'*)

check that all parentheses in a string are balanced and nested properly

archivebox.misc.util.**split_comma_separated_urls**(*url: str*)

archivebox.misc.util.**find_all_urls**(*urls_str: str*)

archivebox.misc.util.**parse_filesize_to_bytes**(*value: str | int | float | None*) → int

Parse a byte count from an integer or human-readable string like 45mb or 2 GB.

archivebox.misc.util.**enforce_types**(*func*)

Enforce function arg and kwarg types at runtime using its python3 type hints Simpler version of pydantic @validate_call decorator

archivebox.misc.util.**docstring**(*text: str | None*)

attach the given docstring to the decorated function

archivebox.misc.util.**parse_date**(*date: Any*) → datetime.datetime | None

Parse unix timestamps, iso format, and human-readable strings

archivebox.misc.util.**download_url**(*url: str*, *timeout: int | None = None*, *config=None*, ***config_kwargs*) → str

Download the contents of a remote url and return the text

archivebox.misc.util.**ansi_to_html**(*text: str*) → str

Based on: https://stackoverflow.com/questions/19212665/python-converting-ansi-color-codes-to-html Simple way to render colored CLI stdout/stderr in HTML properly, Textual/rich is probably better though.

class archivebox.misc.util.**ExtendedEncoder**(***, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: `json.JSONEncoder`

Extended json serializer that supports serializing several model fields and objects

Initialization

Constructor for JSONEncoder, with sensible defaults.

If skipkeys is false, then it is a TypeError to attempt encoding of keys that are not str, int, float, bool or None. If skipkeys is True, such items are simply skipped.

If ensure_ascii is true, the output is guaranteed to be str objects with all incoming non-ASCII characters escaped. If ensure_ascii is false, the output can contain non-ASCII characters.

If check_circular is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an RecursionError). Otherwise, no such check takes place.

If allow_nan is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a ValueError to encode such floats.

If sort_keys is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If indent is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, separators should be an (item_separator, key_separator) tuple. The default is (', ', ': ') if *indent* is `None` and (',', ': ') otherwise. To get the most compact JSON representation, you should specify (',', ':') to eliminate whitespace.

If specified, default is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a `TypeError`.

default(*o*)

archivebox.misc.util.**to_json**(*obj: Any*, *indent: int | None = 4*, *sort_keys: bool = True*) → str

Serialize object to JSON string with extended type support

archivebox.cli

Submodules

archivebox.cli.archivebox_remove

Module Contents

Functions

remove	Remove the specified URLs from the archive
main	Remove the specified URLs from the archive

Data

__command__

API

archivebox.cli.archivebox_remove.**__command__**

'archivebox remove'

archivebox.cli.archivebox_remove.**remove**(*filter_patterns: collections.abc.Iterable[str] = ()*, *filter_type: str = 'exact'*, *snapshots: django.db.models.QuerySet | None = None*, *after: float | None = None*, *before: float | None = None*, *yes: bool = False*, *delete: bool = False*, *out_dir: pathlib.Path = CONSTANTS.DATA_DIR*, *timeout: float | None = None*, ***kwargs*) → dict[str, object]

Remove the specified URLs from the archive

archivebox.cli.archivebox_remove.**main**(***kwargs*)

Remove the specified URLs from the archive

archivebox.cli.archivebox_crawl

archivebox crawl <action> [args...] [–filters]

Manage Crawl records.

Actions: create - Create Crawl jobs from URLs list - List Crawls as JSONL (with optional filters) update - Update Crawls from stdin JSONL delete - Delete Crawls from stdin JSONL

Examples: # Create archivebox crawl create https://example.com https://foo.com –depth=1 archivebox crawl create –tag=news https://example.com

```
# List with filters
archivebox crawl list --status=queued
archivebox crawl list --urls__icontains=example.com

# Update
archivebox crawl list --status=started | archivebox crawl update --status=queued

# Delete
archivebox crawl list --urls__icontains=spam.com | archivebox crawl delete --yes

# Full pipeline
archivebox crawl create https://example.com | archivebox snapshot create | archivebox run
```

Module Contents

Functions

create_crawl	Create a Crawl job from URLs.
list_crawls	List Crawls as JSONL with optional filters.
update_crawls	Update Crawls from stdin JSONL.
delete_crawls	Delete Crawls from stdin JSONL.
main	Manage Crawl records.
create_cmd	Create a Crawl job from URLs or stdin.
list_cmd	List Crawls as JSONL.
update_cmd	Update Crawls from stdin JSONL.
delete_cmd	Delete Crawls from stdin JSONL.

Data

__command__

API

archivebox.cli.archivebox_crawl.**__command__**

'archivebox crawl'

archivebox.cli.archivebox_crawl.**create_crawl**(*urls: collections.abc.Iterable[str]*, *depth: int = 0*, *tag: str = ''*, *status: str = 'queued'*, *created_by_id: int | None = None*) → int

Create a Crawl job from URLs.

Takes URLs as args or stdin, creates one Crawl with all URLs, outputs JSONL. Pass-through: Records that are not URLs are output unchanged (for piping).

Exit codes: 0: Success 1: Failure

archivebox.cli.archivebox_crawl.**list_crawls**(*status: str | None = None*, *urls__icontains: str | None = None*, *max_depth: int | None = None*, *limit: int | None = None*) → int

List Crawls as JSONL with optional filters.

Exit codes: 0: Success (even if no results)

archivebox.cli.archivebox_crawl.**update_crawls**(*status: str | None = None*, *max_depth: int | None = None*) → int

Update Crawls from stdin JSONL.

Reads Crawl records from stdin and applies updates. Uses PATCH semantics - only specified fields are updated.

Exit codes: 0: Success 1: No input or error

archivebox.cli.archivebox_crawl.**delete_crawls**(*yes: bool = False*, *dry_run: bool = False*) → int

Delete Crawls from stdin JSONL.

Requires –yes flag to confirm deletion.

Exit codes: 0: Success 1: No input or missing –yes flag

archivebox.cli.archivebox_crawl.**main**()

Manage Crawl records.

archivebox.cli.archivebox_crawl.**create_cmd**(*urls: tuple*, *depth: int*, *tag: str*, *status: str*)

Create a Crawl job from URLs or stdin.

archivebox.cli.archivebox_crawl.**list_cmd**(*status: str | None*, *urls__icontains: str | None*, *max_depth: int | None*, *limit: int | None*)

List Crawls as JSONL.

archivebox.cli.archivebox_crawl.**update_cmd**(*status: str | None*, *max_depth: int | None*)

Update Crawls from stdin JSONL.

archivebox.cli.archivebox_crawl.**delete_cmd**(*yes: bool*, *dry_run: bool*)

Delete Crawls from stdin JSONL.

archivebox.cli.archivebox_schedule

Module Contents

Functions

schedule	Manage database-backed scheduled crawls processed by the crawl runner.
main	Manage database-backed scheduled crawls processed by the crawl runner.

API

archivebox.cli.archivebox_schedule.**schedule**(*add: bool = False*, *show: bool = False*, *clear: bool = False*, *foreground: bool = False*, *run_all: bool = False*, *quiet: bool = False*, *every: str | None = None*, *tag: str = ''*, *depth: int | str = 0*, *import_path: str | None = None*, *config: dict[str, object] | None = None*)

Manage database-backed scheduled crawls processed by the crawl runner.

archivebox.cli.archivebox_schedule.**main**(***kwargs*)

Manage database-backed scheduled crawls processed by the crawl runner.

archivebox.cli.archivebox_search

Module Contents

Data

__command__

API

archivebox.cli.archivebox_search.**__command__**

'archivebox search'

archivebox.cli.archivebox_run

archivebox run [–daemon] [–crawl-id=…] [–snapshot-id=…] [–binary-id=…]

Unified command for processing queued work on the shared abx-dl bus.

Modes: - With stdin JSONL: Process piped records, exit when complete - Without stdin (TTY): Run the background runner in foreground until killed - –crawl-id: Run the crawl runner for a specific crawl only - –snapshot-id: Run a specific snapshot through its parent crawl - –binary-id: Emit a BinaryRequestEvent for a specific Binary row

Examples: # Run the background runner in foreground archivebox run

```
# Run as daemon (don't exit on idle)
archivebox run --daemon

# Process specific records (pipe any JSONL type, exits when done)
```

(continues on next page)

(continued from previous page)

```
archivebox snapshot list --status=queued | archivebox run
archivebox archiveresult list --status=failed | archivebox run
archivebox crawl list --status=queued | archivebox run

# Mixed types work too
cat mixed_records.jsonl | archivebox run

# Run the crawl runner for a specific crawl
archivebox run --crawl-id=019b7e90-04d0-73ed-adec-aad9cfcd863e

# Run one snapshot from an existing crawl
archivebox run --snapshot-id=019b7e90-5a8e-712c-9877-2c70eebe80ad

# Run one queued binary install directly on the bus
archivebox run --binary-id=019b7e90-5a8e-712c-9877-2c70eebe80ad
```

Module Contents

Functions

_exit_daemon_runner_on_signal	
process_stdin_records	Process JSONL records from stdin.
run_runner	Run the background runner loop.
main	Process queued work.
run_snapshot_worker	

Data

__command__
RUNNER_DAEMON_ENV

API

archivebox.cli.archivebox_run.**__command__**

‘archivebox run’

archivebox.cli.archivebox_run.**RUNNER_DAEMON_ENV**

‘ARCHIVEBOX_RUNNER_DAEMON’

archivebox.cli.archivebox_run.**_exit_daemon_runner_on_signal**(*sig: signal.Signals*) → None

archivebox.cli.archivebox_run.**process_stdin_records**() → int

Process JSONL records from stdin.

Create-or-update behavior:

- Records WITHOUT id: Create via Model.from_json(), then queue
- Records WITH id: Lookup existing, re-queue for processing

Outputs JSONL of all processed records (for chaining).

Handles Crawl and Snapshot work records. ArchiveResult records are accepted as references to their parent Snapshot and plugin.

Returns exit code (0 = success, 1 = error).

archivebox.cli.archivebox_run.**run_runner**(*daemon: bool = False*, *crawl_id: str | None = None*, *maintenance_only: bool = False*) → int

Run the background runner loop.

Args: daemon: Run forever (don't exit when idle)

Returns exit code (0 = success, 1 = error).

archivebox.cli.archivebox_run.**main**(*daemon: bool*, *crawl_id: str*, *snapshot_id: str*, *binary_id: str*, *maintenance_only: bool*, *no_stdin: bool*)

Process queued work.

Modes:

- No args + stdin piped: Process piped JSONL records
- No args + TTY: Run the crawl runner for all work
- –crawl-id: Run the crawl runner for that crawl only
- –snapshot-id: Run one snapshot through its crawl only
- –binary-id: Run one queued binary install directly on the bus

archivebox.cli.archivebox_run.**run_snapshot_worker**(*snapshot_id: str*, *, *daemon: bool = False*) → int

archivebox.cli.archivebox_help

Module Contents

Functions

_command_doc	
help	Print the ArchiveBox help message and usage
main	Print the ArchiveBox help message and usage

Data

__command__

API

archivebox.cli.archivebox_help.**__command__**

'archivebox help'

archivebox.cli.archivebox_help.**_command_doc**(*cmd: str*, *import_path: str*) → str

archivebox.cli.archivebox_help.**help**() → None

Print the ArchiveBox help message and usage

archivebox.cli.archivebox_help.**main**(***kwargs*)

Print the ArchiveBox help message and usage

archivebox.cli.archivebox_binary

archivebox binary <action> [args...] [–filters]

Manage Binary records (detected executables like chrome, wget, etc.).

Actions: create - Create/register a Binary list - List Binaries as JSONL (with optional filters) update - Update Binaries from stdin JSONL delete - Delete Binaries from stdin JSONL

Examples: # List all binaries archivebox binary list

```
# List specific binary
archivebox binary list --name=chrome

# List binaries with specific version
archivebox binary list --version__icontains=120

# Delete old binary entries
archivebox binary list --name=chrome | archivebox binary delete --yes
```

Module Contents

Functions

create_binary	Create/register a Binary.
list_binaries	List Binaries as JSONL with optional filters.
update_binaries	Update Binaries from stdin JSONL.
delete_binaries	Delete Binaries from stdin JSONL.
main	Manage Binary records (detected executables).
create_cmd	Create/register a Binary.
list_cmd	List Binaries as JSONL.
update_cmd	Update Binaries from stdin JSONL.
delete_cmd	Delete Binaries from stdin JSONL.

Data

__command__

API

archivebox.cli.archivebox_binary.**__command__**

'archivebox binary'

archivebox.cli.archivebox_binary.**create_binary**(*name: str*, *abspath: str*, *version: str = ''*) → int

Create/register a Binary.

Exit codes: 0: Success 1: Failure

archivebox.cli.archivebox_binary.**list_binaries**(*name: str | None = None*, *abspath__icontains: str | None = None*, *version__icontains: str | None = None*, *limit: int | None = None*) → int

List Binaries as JSONL with optional filters.

Exit codes: 0: Success (even if no results)

archivebox.cli.archivebox_binary.**update_binaries**(*version: str | None = None*, *abspath: str | None = None*) → int

Update Binaries from stdin JSONL.

Reads Binary records from stdin and applies updates. Uses PATCH semantics - only specified fields are updated.

Exit codes: 0: Success 1: No input or error

archivebox.cli.archivebox_binary.**delete_binaries**(*yes: bool = False*, *dry_run: bool = False*) → int

Delete Binaries from stdin JSONL.

Requires –yes flag to confirm deletion.

Exit codes: 0: Success 1: No input or missing –yes flag

archivebox.cli.archivebox_binary.**main**()

Manage Binary records (detected executables).

archivebox.cli.archivebox_binary.**create_cmd**(*name: str*, *abspath: str*, *version: str*)

Create/register a Binary.

archivebox.cli.archivebox_binary.**list_cmd**(*name: str | None*, *abspath__icontains: str | None*, *version__icontains: str | None*, *limit: int | None*)

List Binaries as JSONL.

archivebox.cli.archivebox_binary.**update_cmd**(*version: str | None*, *abspath: str | None*)

Update Binaries from stdin JSONL.

archivebox.cli.archivebox_binary.**delete_cmd**(*yes: bool*, *dry_run: bool*)

Delete Binaries from stdin JSONL.

archivebox.cli.archivebox_update

Module Contents

Functions

_get_snapshot_crawl	
_get_search_indexing_plugins	
_build_filtered_snapshots_queryset	
reindex_snapshots	
run_scheduled_maintenance	Queue stale filesystem rows and backfill missing search facts.
update	Update the database without forcing a filesystem scan by default.
drain_old_archive_dirs	Drain old archive/ directories (0.8.x → 0.9.x migration).
process_all_db_snapshots	Queue only snapshots whose indexed filesystem version is stale.
process_filtered_snapshots	Process snapshots matching filters (DB query only).
print_stats	Print statistics for filtered mode.
print_combined_stats	Print statistics for full mode.
print_index_stats	
main	

API

archivebox.cli.archivebox_update.**_get_snapshot_crawl**(*snapshot:* archivebox.core.models.Snapshot) → *archivebox.crawls.models.Crawl* | None

archivebox.cli.archivebox_update.**_get_search_indexing_plugins**() → list[str]

archivebox.cli.archivebox_update.**_build_filtered_snapshots_queryset**(***kwargs*)

archivebox.cli.archivebox_update.**reindex_snapshots**(*snapshots: django.db.models.QuerySet[*archivebox.core.models.Snapshot, archivebox.core.models.Snapshot*]*, ***, *search_plugins: list[str]*, *batch_size: int*, *collect_ids: bool = False*, *reverse: bool = False*, *wait_for_turn=None*) → dict[str, Any]

archivebox.cli.archivebox_update.**run_scheduled_maintenance**(***, *batch_size: int = 500*) → dict[str, Any]

Queue stale filesystem rows and backfill missing search facts.

archivebox.cli.archivebox_update.**update**(*filter_patterns: collections.abc.Iterable[str] = ()*, *filter_type: str = 'exact'*, *status: str | None = None*, *url__icontains: str | None = None*, *url__istartswith: str | None = None*, *tag: str | None = None*, *crawl_id: str | None = None*, *limit: int | None = None*, *sort: str | None = None*, *search: str | None = None*, *before: float | None = None*, *after: float | None = None*, *resume: str | None = None*, *batch_size: int = 500*, *continuous: bool = False*, *index_only: bool = False*, *migrate_only: bool = False*, *rescan: bool = False*, *reverse: bool = False*, *stop_daemon_stack: bool = True*) → None

Update the database without forcing a filesystem scan by default.

–migrate-only: eagerly finish pending filesystem migrations for known rows. –index-only: reconcile known snapshot assets/metadata and backfill search. –rescan: discover filesystem orphans first, then repair known snapshots. Explicit scans default to newest first; –reverse selects oldest first. Filters select existing database snapshots for targeted updates.

archivebox.cli.archivebox_update.**drain_old_archive_dirs**(*resume_from: str | None = None*, *batch_size: int = 500*, *reverse: bool = False*) → dict[str, int]

Drain old archive/ directories (0.8.x → 0.9.x migration).

Removes obsolete timestamp symlinks and processes real legacy directories. For each old dir found in archive/:

1. Load or create DB snapshot
2. Trigger fs migration on save() to move to data/archive/users/{user}/...
3. Remove the old timestamp path after the verified migration commits

After this drains, current snapshot data exists only under archive/users/.

archivebox.cli.archivebox_update.**process_all_db_snapshots**(*batch_size: int = 500*, *resume: str | None = None*, *wait_for_turn=None*, *reverse: bool = False*, *migrate: bool = False*) → dict[str, int]

Queue only snapshots whose indexed filesystem version is stale.

archivebox.cli.archivebox_update.**process_filtered_snapshots**(*filter_patterns: collections.abc.Iterable[str], filter_type: str, status: str | None, url__icontains: str | None, url__istartswith: str | None, tag: str | None, crawl_id: str | None, limit: int | None, sort: str | None, search: str | None, before: float | None, after: float | None, resume: str | None, batch_size: int, queue_for_archiving: bool = True, reverse: bool = False, wait_for_turn=None*) → dict[str, Any]

Process snapshots matching filters (DB query only).

archivebox.cli.archivebox_update.**print_stats**(*stats: dict*)

Print statistics for filtered mode.

archivebox.cli.archivebox_update.**print_combined_stats**(*stats_combined: dict*)

Print statistics for full mode.

archivebox.cli.archivebox_update.**print_index_stats**(*stats: dict[str, Any]*) → None

archivebox.cli.archivebox_update.**main**(***kwargs*)

archivebox.cli.archivebox_machine

archivebox machine <action> [–filters]

Manage Machine records (system-managed, mostly read-only).

Machine records track the host machines where ArchiveBox runs. They are created automatically by the system and are primarily for debugging.

Actions: list - List Machines as JSONL (with optional filters)

Examples: # List all machines archivebox machine list

```
# List machines by hostname
archivebox machine list --hostname__icontains=myserver
```

Module Contents

Functions

list_machines	List Machines as JSONL with optional filters.
main	Manage Machine records (read-only, system-managed).
list_cmd	List Machines as JSONL.

Data

__command__

API

archivebox.cli.archivebox_machine.**__command__**

'archivebox machine'

archivebox.cli.archivebox_machine.**list_machines**(*hostname__icontains: str | None = None*, *os_platform: str | None = None*, *limit: int | None = None*) → int

List Machines as JSONL with optional filters.

Exit codes: 0: Success (even if no results)

archivebox.cli.archivebox_machine.**main**()

Manage Machine records (read-only, system-managed).

archivebox.cli.archivebox_machine.**list_cmd**(*hostname__icontains: str | None*, *os_platform: str | None*, *limit: int | None*)

List Machines as JSONL.

archivebox.cli.archivebox_shell

Module Contents

Functions

shell	Enter an interactive ArchiveBox Django shell
main	

API

archivebox.cli.archivebox_shell.**shell**(*args: collections.abc.Iterable[str] = ()*) → None

Enter an interactive ArchiveBox Django shell

archivebox.cli.archivebox_shell.**main**(*args: collections.abc.Iterable[str] = ()*) → None

archivebox.cli.archivebox_persona

archivebox persona <action> [args...] [–filters]

Manage Persona records (browser profiles for archiving).

Actions: create - Create Personas list - List Personas as JSONL (with optional filters) update - Update Personas from stdin JSONL delete - Delete Personas from stdin JSONL

Examples: # Create a new persona archivebox persona create work archivebox persona create –import=chrome personal archivebox persona create –import=edge work

```
# List all personas
archivebox persona list

# Delete a persona
archivebox persona list --name=old | archivebox persona delete --yes
```

Module Contents

Functions

validate_persona_name	Validate persona name to prevent path traversal attacks.
ensure_path_within_personas_dir	Verify that a persona path is within PERSONAS_DIR.
create_personas	Create Personas from names.
list_personas	List Personas as JSONL with optional filters.
update_personas	Update Personas from stdin JSONL.
delete_personas	Delete Personas from stdin JSONL.
main	Manage Persona records (browser profiles).
create_cmd	Create Personas, optionally importing from a browser profile.
list_cmd	List Personas as JSONL.
open_cmd	Open a persona's browser, including its imported logins.
update_cmd	Update Personas from stdin JSONL.
delete_cmd	Delete Personas from stdin JSONL.

Data

__command__

API

archivebox.cli.archivebox_persona.**__command__**

'archivebox persona'

archivebox.cli.archivebox_persona.**validate_persona_name**(*name: str*) → tuple[bool, str]

Validate persona name to prevent path traversal attacks.

Returns: (is_valid, error_message): tuple indicating if name is valid

archivebox.cli.archivebox_persona.**ensure_path_within_personas_dir**(*persona_path: pathlib.Path*) → bool

Verify that a persona path is within PERSONAS_DIR.

This is a safety check to prevent path traversal attacks where a malicious persona name could cause operations on paths outside the expected PERSONAS_DIR.

Returns: True if path is safe, False otherwise

archivebox.cli.archivebox_persona.**create_personas**(*names: collections.abc.Iterable[str]*, *import_from: str | None = None*, *profile: str | None = None*, *source: str | None = None*, *browser_binary: str | None = None*, *permissions: str | None = None*) → int

Create Personas from names.

If –import is specified, copy the browser profile to the persona directory and extract cookies.

Exit codes: 0: Success 1: Failure

archivebox.cli.archivebox_persona.**list_personas**(*name: str | None = None*, *name__icontains: str | None = None*, *limit: int | None = None*) → int

List Personas as JSONL with optional filters.

Exit codes: 0: Success (even if no results)

archivebox.cli.archivebox_persona.**update_personas**(*name: str | None = None*) → int

Update Personas from stdin JSONL.

Reads Persona records from stdin and applies updates. Uses PATCH semantics - only specified fields are updated.

Exit codes: 0: Success 1: No input or error

archivebox.cli.archivebox_persona.**delete_personas**(*yes: bool = False*, *dry_run: bool = False*) → int

Delete Personas from stdin JSONL.

Requires –yes flag to confirm deletion.

Exit codes: 0: Success 1: No input or missing –yes flag

archivebox.cli.archivebox_persona.**main**()

Manage Persona records (browser profiles).

archivebox.cli.archivebox_persona.**create_cmd**(*names: tuple*, *import_from: str | None*, *profile: str | None*, *source: str | None*, *browser_binary: str | None*, *permissions: str | None*)

Create Personas, optionally importing from a browser profile.

archivebox.cli.archivebox_persona.**list_cmd**(*name: str | None*, *name__icontains: str | None*, *limit: int | None*)

List Personas as JSONL.

archivebox.cli.archivebox_persona.**open_cmd**(*name: str*)

Open a persona's browser, including its imported logins.

archivebox.cli.archivebox_persona.**update_cmd**(*name: str | None*)

Update Personas from stdin JSONL.

archivebox.cli.archivebox_persona.**delete_cmd**(*yes: bool*, *dry_run: bool*)

Delete Personas from stdin JSONL.

archivebox.cli.archivebox_mcp

archivebox mcp

Start the Model Context Protocol (MCP) server in stdio mode. Exposes all ArchiveBox CLI commands as MCP tools for AI agents.

Module Contents

Functions

mcp	Start the MCP server in stdio mode for AI agent control.
main	Start the MCP server in stdio mode

Data

__command__

API

archivebox.cli.archivebox_mcp.**__command__**

'archivebox mcp'

archivebox.cli.archivebox_mcp.**mcp**()

Start the MCP server in stdio mode for AI agent control.

The MCP (Model Context Protocol) server exposes all ArchiveBox CLI commands as tools that AI agents can discover and execute. It communicates via JSON-RPC 2.0 over stdin/stdout.

Example usage with an MCP client: archivebox mcp < requests.jsonl > responses.jsonl

Or interactively: archivebox mcp {"jsonrpc":"2.0","id":1,"method":"initialize","params":{}} {"jsonrpc":"2.0","id":2,"method":"tools/list","params":{}}

archivebox.cli.archivebox_mcp.**main**(***kwargs*)

Start the MCP server in stdio mode

archivebox.cli.archivebox_version

Module Contents

Functions

_format_binary_abspath	
_render_binary_abspath	
_binary_row_dedupe_key	
version	Print the ArchiveBox version, debug metadata, and installed dependency versions
main	

API

archivebox.cli.archivebox_version.**_format_binary_abspath**(*abspath: str*, ***, *pwd: pathlib.Path*, *lib_dir: pathlib.Path*, *personas_dir: pathlib.Path*, *home: pathlib.Path*) → str

archivebox.cli.archivebox_version.**_render_binary_abspath**(*abspath: str*)

archivebox.cli.archivebox_version.**_binary_row_dedupe_key**(***, *display_name: str*, *valid: bool*, *version: str*, *provider: str*, *abspath: str*) → tuple[str, str, str, str]

archivebox.cli.archivebox_version.**version**(*quiet: bool = False*, *binaries: collections.abc.Iterable[str] = ()*, *no_cache: bool = False*) → list[str]

Print the ArchiveBox version, debug metadata, and installed dependency versions

archivebox.cli.archivebox_version.**main**(***kwargs*)

archivebox.cli.archivebox_config

Module Contents

Functions

_format_toml	
config	Get and set your ArchiveBox project configuration values
main	

API

archivebox.cli.archivebox_config.**_format_toml**(*config: dict*) → str

archivebox.cli.archivebox_config.**config**(**keys*, *get: bool = False*, *set: bool = False*, *search: bool = False*, *reset: bool = False*, ***kwargs*) → None

Get and set your ArchiveBox project configuration values

archivebox.cli.archivebox_config.**main**(***kwargs*) → None

archivebox.cli.archivebox_oneshot

Module Contents

Functions

main	Download URLs using abx-dl

Data

__command__

API

archivebox.cli.archivebox_oneshot.**__command__**

'archivebox oneshot'

archivebox.cli.archivebox_oneshot.**main**(*args: tuple[str, ...] = ()*) → None

Download URLs using abx-dl

archivebox.cli.archivebox_init

Module Contents

Functions

_display_data_path

continues on next page

Table 64 – continued from previous page

init	Initialize a new ArchiveBox collection in the current directory
main	

API

archivebox.cli.archivebox_init.**_display_data_path**(*path: pathlib.Path*, *data_dir: pathlib.Path*) → str

archivebox.cli.archivebox_init.**init**(*force: bool = False*, *quick: bool = False*, *install: bool = False*) → None

Initialize a new ArchiveBox collection in the current directory

archivebox.cli.archivebox_init.**main**(***kwargs*) → None

archivebox.cli.archivebox_archiveresult

archivebox archiveresult <action> [args...] [–filters]

Manage ArchiveResult records (plugin extraction results).

Actions: create - Emit plugin extraction request records for Snapshots list - List ArchiveResults as JSONL (with optional filters) update - Update ArchiveResults from stdin JSONL delete - Delete ArchiveResults from stdin JSONL

Examples: # Emit extraction requests; `archivebox run` schedules their parent snapshots archivebox snapshot list –status=queued | archivebox archiveresult create archivebox archiveresult create –plugin=screenshot –snapshot-id=<uuid>

```
# List with filters
archivebox archiveresult list --status=failed
archivebox archiveresult list --plugin=screenshot --status=succeeded

# Delete
archivebox archiveresult list --plugin=singlefile | archivebox archiveresult delete --yes

# Re-run failed extractions
archivebox archiveresult list --status=failed | archivebox run
```

Module Contents

Functions

build_archiveresult_request	
create_archiveresults	Create ArchiveResult request records for Snapshots.
list_archiveresults	List ArchiveResults as JSONL with optional filters.
update_archiveresults	Update ArchiveResults from stdin JSONL.
delete_archiveresults	Delete ArchiveResults from stdin JSONL.
main	Manage ArchiveResult records (plugin extraction results).
create_cmd	Emit Snapshot plugin extraction requests as JSONL.
list_cmd	List ArchiveResults as JSONL.
update_cmd	Update ArchiveResults from stdin JSONL.
delete_cmd	Delete ArchiveResults from stdin JSONL.

Data

__command__

API

archivebox.cli.archivebox_archiveresult.**__command__**

'archivebox archiveresult'

archivebox.cli.archivebox_archiveresult.**build_archiveresult_request**(*snapshot_id: str*, *plugin: str*, *hook_name: str = ''*, *status: str = 'queued'*) → dict

archivebox.cli.archivebox_archiveresult.**create_archiveresults**(*snapshot_id: str | None = None*, *plugin: str | None = None*, *status: str = 'queued'*) → int

Create ArchiveResult request records for Snapshots.

Reads Snapshot records from stdin and emits ArchiveResult request JSONL. Pass-through: Non-Snapshot/ArchiveResult records are output unchanged. If –plugin is specified, only emits requests for that plugin. Otherwise, emits requests for all enabled snapshot hooks.

Exit codes: 0: Success 1: Failure

archivebox.cli.archivebox_archiveresult.**list_archiveresults**(*status: str | None = None*, *plugin: str | None = None*, *snapshot_id: str | None = None*, *limit: int | None = None*) → int

List ArchiveResults as JSONL with optional filters.

Exit codes: 0: Success (even if no results)

archivebox.cli.archivebox_archiveresult.**update_archiveresults**(*status: str | None = None*) → int

Update ArchiveResults from stdin JSONL.

Reads ArchiveResult records from stdin and applies updates. Uses PATCH semantics - only specified fields are updated.

Exit codes: 0: Success 1: No input or error

archivebox.cli.archivebox_archiveresult.**delete_archiveresults**(*yes: bool = False*, *dry_run: bool = False*) → int

Delete ArchiveResults from stdin JSONL.

Requires –yes flag to confirm deletion.

Exit codes: 0: Success 1: No input or missing –yes flag

archivebox.cli.archivebox_archiveresult.**main**()

Manage ArchiveResult records (plugin extraction results).

archivebox.cli.archivebox_archiveresult.**create_cmd**(*snapshot_id: str | None*, *plugin: str | None*, *status: str*)

Emit Snapshot plugin extraction requests as JSONL.

archivebox.cli.archivebox_archiveresult.**list_cmd**(*status: str | None*, *plugin: str | None*, *snapshot_id: str | None*, *limit: int | None*)

List ArchiveResults as JSONL.

archivebox.cli.archivebox_archiveresult.**update_cmd**(*status: str | None*)

Update ArchiveResults from stdin JSONL.

archivebox.cli.archivebox_archiveresult.**delete_cmd**(*yes: bool*, *dry_run: bool*)

Delete ArchiveResults from stdin JSONL.

archivebox.cli.archivebox_extract

Run abx-dl snapshot hooks for existing ArchiveBox snapshots.

Module Contents

Functions

_resolve_requests	Resolve CLI records to snapshot-level execution requests.
_run_snapshot_requests	
run_plugins	Execute selected plugins through the snapshot-level runner.
process_archiveresult_by_id	Re-run the parent Snapshot plugin referenced by an ArchiveResult.
main	Run plugins on Snapshots; ArchiveResult IDs select their parent plugin.

Data

__command__

API

archivebox.cli.archivebox_extract.**__command__**

'archivebox extract'

archivebox.cli.archivebox_extract.**_resolve_requests**(*records: list[dict]*, *plugins: str*) → tuple[dict[str, set[str]], set[str]]

Resolve CLI records to snapshot-level execution requests.

ArchiveResult input is accepted as a convenient reference to its parent Snapshot and plugin. It is never reset or converted into queued work.

archivebox.cli.archivebox_extract.**_run_snapshot_requests**(*requested: dict[str, set[str]]*, ***, *wait: bool*, *show_progress: bool*) → int

archivebox.cli.archivebox_extract.**run_plugins**(*args: tuple*, *records: list[dict] | None = None*, *plugins: str = ''*, *wait: bool = True*, *emit_results: bool = True*, *show_progress: bool = True*) → int

Execute selected plugins through the snapshot-level runner.

archivebox.cli.archivebox_extract.**process_archiveresult_by_id**(*archiveresult_id: str*) → int

Re-run the parent Snapshot plugin referenced by an ArchiveResult.

archivebox.cli.archivebox_extract.**main**(*plugins: str*, *wait: bool*, *args: tuple*)

Run plugins on Snapshots; ArchiveResult IDs select their parent plugin.

archivebox.cli.rescan

Explicit discovery of portable snapshots; no checkpoint or default tree walk.

Module Contents

Functions

_directories	
_snapshot_directories	
_fields	
_dates	
_restore_dates	
_reconcile_directory	
reconcile_known_snapshots	Explicit metadata / asset repair; normal updates never call this scan.
rescan_snapshots	

API

archivebox.cli.rescan.**_directories**(*parent: pathlib.Path*)

archivebox.cli.rescan.**_snapshot_directories**(*reverse=False*)

archivebox.cli.rescan.**_fields**(*record*, *names*)

archivebox.cli.rescan.**_dates**(*record*)

archivebox.cli.rescan.**_restore_dates**(*obj*, *record*)

archivebox.cli.rescan.**_reconcile_directory**(*directory: pathlib.Path*) → str

archivebox.cli.rescan.**reconcile_known_snapshots**(*snapshots*, *wait_for_turn=None*)

Explicit metadata / asset repair; normal updates never call this scan.

archivebox.cli.rescan.**rescan_snapshots**(*wait_for_turn=None*, *reverse=False*, *scan_legacy=None*)

archivebox.cli.archivebox_manage

Module Contents

Functions

manage	Run an ArchiveBox Django management command
main	

API

archivebox.cli.archivebox_manage.**manage**(*args: list[str] | None = None*) → None

Run an ArchiveBox Django management command

archivebox.cli.archivebox_manage.**main**(*args: list[str] | None = None*) → None

archivebox.cli.archivebox_process

archivebox process <action> [–filters]

Manage Process records (system-managed, mostly read-only).

Process records track executions of binaries during extraction. They are created automatically by the system and are primarily for debugging.

Actions: list - List Processes as JSONL (with optional filters)

Examples: # List all processes archivebox process list

```
# List processes by binary
archivebox process list --binary-name=chrome

# List recent processes
archivebox process list --limit=10
```

Module Contents

Functions

list_processes	List Processes as JSONL with optional filters.
main	Manage Process records (read-only, system-managed).
list_cmd	List Processes as JSONL.

Data

__command__

API

archivebox.cli.archivebox_process.**__command__**

'archivebox process'

archivebox.cli.archivebox_process.**list_processes**(*binary_name: str | None = None*, *machine_id: str | None = None*, *limit: int | None = None*) → int

List Processes as JSONL with optional filters.

Exit codes: 0: Success (even if no results)

archivebox.cli.archivebox_process.**main**()

Manage Process records (read-only, system-managed).

archivebox.cli.archivebox_process.**list_cmd**(*binary_name: str | None*, *machine_id: str | None*, *limit: int | None*)

List Processes as JSONL.

archivebox.cli.archivebox_status

Module Contents

Functions

status	Print out some info and statistics about the archive collection
main	Print out some info and statistics about the archive collection

Data

MAX_STATUS_FS_DIR_SCAN

API

archivebox.cli.archivebox_status.**MAX_STATUS_FS_DIR_SCAN**
5000

archivebox.cli.archivebox_status.**status**(*out_dir: pathlib.Path = CONSTANTS.DATA_DIR*) → None
Print out some info and statistics about the archive collection

archivebox.cli.archivebox_status.**main**(***kwargs*)
Print out some info and statistics about the archive collection

archivebox.cli.archivebox_install

Module Contents

Functions

_resolve_install_targets	Resolve plugin names and declared binary aliases, leaving unknown tokens raw.
_install_raw_binary_names	Install user-requested standalone binaries through the Binary lifecycle.
ensure_data_dir_lib_symlink	Expose the shared abxpkg lib dir at DATA_DIR/lib after install.
install	Detect and install ArchiveBox dependencies by running the abx-dl install flow
main	

API

archivebox.cli.archivebox_install.**_resolve_install_targets**(*requested_names: tuple[str, ...]*) → tuple[list[str], list[str]]
Resolve plugin names and declared binary aliases, leaving unknown tokens raw.

archivebox.cli.archivebox_install.**_install_raw_binary_names**(*binary_names: list[str]*, *binproviders: str*) → None
Install user-requested standalone binaries through the Binary lifecycle.

archivebox.cli.archivebox_install.**ensure_data_dir_lib_symlink**(*data_dir: pathlib.Path*, *abxpkg_lib_dir: pathlib.Path*) → pathlib.Path | None

Expose the shared abxpkg lib dir at DATA_DIR/lib after install.

archivebox.cli.archivebox_install.**install**(*binaries: tuple[str, ...] = ()*, *binproviders: str = '*'*, *dry_run: bool = False*) → None

Detect and install ArchiveBox dependencies by running the abx-dl install flow

Examples: archivebox install # Install all dependencies archivebox install wget curl # Install only wget and curl archivebox install –binproviders=pip yt-dlp # Install yt-dlp using only pip archivebox install –binproviders=brew,apt # Install all deps using only brew or apt

archivebox.cli.archivebox_install.**main**(***kwargs*) → None

archivebox.cli.archivebox_snapshot

archivebox snapshot <action> [args…] [–filters]

Manage Snapshot records.

Actions: create - Create Snapshots from URLs or Crawl JSONL list - List Snapshots as JSONL (with optional filters) update - Update Snapshots from stdin JSONL delete - Delete Snapshots from stdin JSONL

Examples: # Create archivebox snapshot create https://example.com –tag=news archivebox crawl create https://example.com | archivebox snapshot create

```
# List with filters
archivebox snapshot list --status=queued
archivebox snapshot list --url__icontains=example.com

# Update
archivebox snapshot list --tag=old | archivebox snapshot update --tag=new

# Delete
archivebox snapshot list --url__icontains=spam.com | archivebox snapshot delete --yes
```

Module Contents

Functions

iter_snapshot_json	
create_snapshots	Create Snapshots from URLs or stdin JSONL (Crawl or Snapshot records). Pass-through: Records that are not Crawl/Snapshot/URL are output unchanged.
snapshot_filter_options	
snapshot_output_options	
build_snapshot_queryset	
list_snapshots	List Snapshots as JSONL with optional filters.
update_snapshots	Update Snapshots from stdin JSONL.
delete_snapshots	Delete Snapshots from stdin JSONL.
main	Manage Snapshot records.
create_cmd	Create Snapshots from URLs or stdin JSONL.
list_cmd	List Snapshots as JSONL.

continues on next page

Table 76 – continued from previous page

update_cmd	Update Snapshots from stdin JSONL.
delete_cmd	Delete Snapshots from stdin JSONL.

Data

__command__
SNAPSHOT_FILTER_TYPE_CHOICES
SNAPSHOT_LIST_CHUNK_SIZE

API

archivebox.cli.archivebox_snapshot.**__command__**

'archivebox snapshot'

archivebox.cli.archivebox_snapshot.**SNAPSHOT_FILTER_TYPE_CHOICES**

('exact', 'substring', 'regex', 'domain', 'tag', 'timestamp')

archivebox.cli.archivebox_snapshot.**SNAPSHOT_LIST_CHUNK_SIZE**

5000

archivebox.cli.archivebox_snapshot.**iter_snapshot_json**(*queryset: django.db.models.QuerySet*) → collections.abc.Iterator[dict[str, object]]

archivebox.cli.archivebox_snapshot.**create_snapshots**(*urls: collections.abc.Iterable[str]*, *tag: str = ''*, *status: str = 'queued'*, *depth: int = 0*, *created_by_id: int | None = None*) → int

Create Snapshots from URLs or stdin JSONL (Crawl or Snapshot records). Pass-through: Records that are not Crawl/Snapshot/URL are output unchanged.

Exit codes: 0: Success 1: Failure

archivebox.cli.archivebox_snapshot.**snapshot_filter_options**(*, *default_filter_type: str*)

archivebox.cli.archivebox_snapshot.**snapshot_output_options**(*func*)

archivebox.cli.archivebox_snapshot.**build_snapshot_queryset**(***kwargs*) → django.db.models.QuerySet

archivebox.cli.archivebox_snapshot.**list_snapshots**(*csv: str | None = None*, *as_json: bool = False*, *as_html: bool = False*, *with_headers: bool = False*, ***kwargs*) → int

List Snapshots as JSONL with optional filters.

Exit codes: 0: Success (even if no results)

archivebox.cli.archivebox_snapshot.**update_snapshots**(*status: str | None = None*, *tag: str | None = None*) → int

Update Snapshots from stdin JSONL.

Reads Snapshot records from stdin and applies updates. Uses PATCH semantics - only specified fields are updated.

Exit codes: 0: Success 1: No input or error

archivebox.cli.archivebox_snapshot.**delete_snapshots**(*yes: bool = False*, *dry_run: bool = False*) → int

Delete Snapshots from stdin JSONL.

Requires –yes flag to confirm deletion.

Exit codes: 0: Success 1: No input or missing –yes flag

archivebox.cli.archivebox_snapshot.**main**()

Manage Snapshot records.

archivebox.cli.archivebox_snapshot.**create_cmd**(*urls: tuple*, *tag: str*, *status: str*, *depth: int*)

Create Snapshots from URLs or stdin JSONL.

archivebox.cli.archivebox_snapshot.**list_cmd**(***kwargs*)

List Snapshots as JSONL.

archivebox.cli.archivebox_snapshot.**update_cmd**(*status: str | None*, *tag: str | None*)

Update Snapshots from stdin JSONL.

archivebox.cli.archivebox_snapshot.**delete_cmd**(*yes: bool*, *dry_run: bool*)

Delete Snapshots from stdin JSONL.

archivebox.cli.archivebox_server

Module Contents

Functions

_is_ipv4_literal	
_is_ipv6_literal	
_bind_host_looks_like_ip	
_split_bind_spec	Split a `host:port` / `host` / `port` spec into `(host, port)`.
_parse_and_validate_bind_spec	Resolve a CLI/config bind spec to `(host, port)` or hard-error.
_print_server_startup_warnings	Print startup-time security / routing warnings for the server command.
server	Run the ArchiveBox HTTP server
main	

Data

_IPV4_RE
_IPV6_CHARS_RE
_LOCAL_BIND_HOSTS

API

archivebox.cli.archivebox_server.**_IPV4_RE**

'compile(…)'

archivebox.cli.archivebox_server.**_IPV6_CHARS_RE**

'compile(…)'

archivebox.cli.archivebox_server.**_LOCAL_BIND_HOSTS**

'frozenset(...)'

archivebox.cli.archivebox_server.**_is_ipv4_literal**(*host: str*) → bool

archivebox.cli.archivebox_server.**_is_ipv6_literal**(*host: str*) → bool

archivebox.cli.archivebox_server.**_bind_host_looks_like_ip**(*host: str*) → bool

archivebox.cli.archivebox_server.**_split_bind_spec**(*spec: str*) → tuple[str, str]

Split a `host:port` / `host` / `port` spec into `(host, port)`.

The empty strings stand in for "not provided"; the caller fills in defaults. Bracketed IPv6 literals like `[::1]:5797` are handled.

archivebox.cli.archivebox_server.**_parse_and_validate_bind_spec**(*spec: str*) → tuple[str, str]

Resolve a CLI/config bind spec to `(host, port)` or hard-error.

Accepts only IP literals (v4 or v6) or the special string `localhost` (normalized to `127.0.0.1`). Bare hostnames are rejected because the bind address feeds Daphne, which has to listen on a numeric address; public hostnames belong in `BASE_URL` instead. Empty values fall back to `127.0.0.1` / `5797`.

archivebox.cli.archivebox_server.**_print_server_startup_warnings**(*config*, *host: str*, *port: str*) → None

Print startup-time security / routing warnings for the server command.

Runs only from `archivebox server` so other entry points (manage shell, plugin lookups, etc.) don't repeat this banner on every config load.

archivebox.cli.archivebox_server.**server**(*runserver_args: collections.abc.Iterable[str] | None = None*, *reload: bool = False*, *debug: bool = False*, *daemonize: bool = False*, *nothreading: bool = False*, *createsuperuser: bool = False*) → None

Run the ArchiveBox HTTP server

archivebox.cli.archivebox_server.**main**(***kwargs*)

archivebox.cli.archivebox_add

Module Contents

Functions

_collect_input_urls	
add	Add a URL list or imported URL document to a new Crawl.
main	Add a new URL or list of URLs to your archive

Data

__command__

API

archivebox.cli.archivebox_add.**__command__**

'archivebox add'

archivebox.cli.archivebox_add.**_collect_input_urls**(*args: tuple[str, ...]*, *, *parser: str = 'auto'*) → list[str]

archivebox.cli.archivebox_add.**add**(*urls: str | list[str]*, *snapshot_ids: list[str] | None = None*, *depth: int | str = 0*, *max_urls: int = 0*, *crawl_max_size: int | str = 0*, *crawl_timeout: int = 0*, *snapshot_max_size: int | str = 0*, *crawl_max_concurrent_snapshots: int | None = None*, *tag: str = ''*, *url_allowlist: str = ''*, *url_denylist: str = ''*, *parser: str = 'auto'*, *plugins: str = ''*, *persona: str = 'Default'*, *index_only: bool = False*, *bg: bool = False*, *created_by_id: int | None = None*, *config: dict[str, Any] | None = None*) → tuple[*archivebox.crawls.models.Crawl*, django.db.models.QuerySet[*archivebox.core.models.Snapshot*]]

Add a URL list or imported URL document to a new Crawl.

archivebox.cli.archivebox_add.**main**(***kwargs*)

Add a new URL or list of URLs to your archive

archivebox.cli.archivebox_pluginmap

Module Contents

Functions

pluginmap	Show the current abx-dl event phases and their associated plugin hooks.
main	

Data

EVENT_FLOW_DIAGRAM

API

archivebox.cli.archivebox_pluginmap.**EVENT_FLOW_DIAGRAM = <Multiline-String>**

archivebox.cli.archivebox_pluginmap.**pluginmap**(*show_disabled: bool = False*, *event: str | None = None*, *quiet: bool = False*) → dict

Show the current abx-dl event phases and their associated plugin hooks.

This command reflects the new bus-driven runtime, not the legacy ArchiveBox event runtime. Event names are normalized to hook prefixes by stripping a trailing `Event`, then ArchiveBox checks whether any matching `on_{EventFamily}__*` scripts actually exist.

archivebox.cli.archivebox_pluginmap.**main**(***kwargs*)

archivebox.cli.cli_util

Shared CLI utilities for ArchiveBox commands.

This module contains common utilities used across multiple CLI commands, extracted to avoid code duplication.

Module Contents

Functions

apply_filters	Apply Django-style filters from CLI kwargs to a QuerySet.

API

archivebox.cli.cli_util.**apply_filters**(*queryset*, *filter_kwargs: dict*, *limit: int | None = None*)

Apply Django-style filters from CLI kwargs to a QuerySet.

Supports: –status=queued, –url__icontains=example, –id__in=uuid1,uuid2

Args: queryset: Django QuerySet to filter filter_kwargs: Dict of filter key-value pairs from CLI limit: Optional limit on results

Returns: Filtered QuerySet

Example: queryset = Snapshot.objects.all() filter_kwargs = {'status': 'queued', 'url__icontains': 'example.com'} filtered = apply_filters(queryset, filter_kwargs, limit=10)

archivebox.cli.archivebox_tag

archivebox tag <action> [args...] [–filters]

Manage Tag records.

Actions: create - Create Tags list - List Tags as JSONL (with optional filters) update - Update Tags from stdin JSONL delete - Delete Tags from stdin JSONL

Examples: # Create archivebox tag create news tech science archivebox tag create "important stuff"

```
# List
archivebox tag list
archivebox tag list --name__icontains=news

# Update (rename tags)
archivebox tag list --name=oldname | archivebox tag update --name=newname

# Delete
archivebox tag list --name=unused | archivebox tag delete --yes
```

Module Contents

Functions

create_tags	Create Tags from names.

continues on next page

Table 85 – continued from previous page

list_tags	List Tags as JSONL with optional filters.
update_tags	Update Tags from stdin JSONL.
delete_tags	Delete Tags from stdin JSONL.
main	Manage Tag records.
create_cmd	Create Tags from names.
list_cmd	List Tags as JSONL.
update_cmd	Update Tags from stdin JSONL.
delete_cmd	Delete Tags from stdin JSONL.

Data

__command__

API

archivebox.cli.archivebox_tag.**__command__**

'archivebox tag'

archivebox.cli.archivebox_tag.**create_tags**(*names: collections.abc.Iterable[str]*) → int

Create Tags from names.

Exit codes: 0: Success 1: Failure

archivebox.cli.archivebox_tag.**list_tags**(*name: str | None = None*, *name__icontains: str | None = None*, *limit: int | None = None*) → int

List Tags as JSONL with optional filters.

Exit codes: 0: Success (even if no results)

archivebox.cli.archivebox_tag.**update_tags**(*name: str | None = None*) → int

Update Tags from stdin JSONL.

Reads Tag records from stdin and applies updates. Uses PATCH semantics - only specified fields are updated.

Exit codes: 0: Success 1: No input or error

archivebox.cli.archivebox_tag.**delete_tags**(*yes: bool = False*, *dry_run: bool = False*) → int

Delete Tags from stdin JSONL.

Requires –yes flag to confirm deletion.

Exit codes: 0: Success 1: No input or missing –yes flag

archivebox.cli.archivebox_tag.**main**()

Manage Tag records.

archivebox.cli.archivebox_tag.**create_cmd**(*names: tuple*)

Create Tags from names.

archivebox.cli.archivebox_tag.**list_cmd**(*name: str | None*, *name__icontains: str | None*, *limit: int | None*)

List Tags as JSONL.

archivebox.cli.archivebox_tag.**update_cmd**(*name: str | None*)

Update Tags from stdin JSONL.

archivebox.cli.archivebox_tag.**delete_cmd**(*yes: bool*, *dry_run: bool*)

Delete Tags from stdin JSONL.

archivebox.cli.archivebox_list

Module Contents

Functions

main	List Snapshots.

Data

__command__

API

archivebox.cli.archivebox_list.**__command__**

'archivebox list'

archivebox.cli.archivebox_list.**main**(***kwargs*) → None

List Snapshots.

Package Contents

Classes

ArchiveBoxGroup	lazy loading click group for archivebox commands

Functions

cli	ArchiveBox: The self-hosted internet archive
main	

Data

__command__
STDERR

API

archivebox.cli.**__command__**

'archivebox'

archivebox.cli.**STDERR**

'Console(...)'

class archivebox.cli.**ArchiveBoxGroup**

Bases: rich_click.Group

lazy loading click group for archivebox commands

meta_commands

None

setup_commands

None

model_commands

None

archive_commands

None

all_subcommands

None

renamed_commands

None

classmethod get_canonical_name(*cmd_name*)

classmethod _needs_django_for_lazy_import(*cmd_name: str*) → bool

classmethod _setup_django_for_lazy_import(*cmd_name: str*) → None

get_command(*ctx*, *cmd_name*)

classmethod _lazy_load(*cmd_name_or_path*)

archivebox.cli.**cli**(*ctx*, *help=False*)

ArchiveBox: The self-hosted internet archive

archivebox.cli.**main**(*args=None*, *prog_name=None*)

archivebox.api

Submodules

archivebox.api.apps

Module Contents

Classes

APIConfig

Functions

register_admin

API

class archivebox.api.apps.**APIConfig**

Bases: django.apps.AppConfig

name

'archivebox.api'

label

'api'

ready()

archivebox.api.apps.**register_admin**(*admin_site*)

archivebox.api.v1_crawls

Module Contents

Classes

CrawlSchema
CrawlUpdateSchema
CrawlCreateSchema
CrawlDeleteResponseSchema

Functions

normalize_tag_list	
get_crawls	
create_crawl	
get_crawl_by_ref	
get_crawl	Get a specific Crawl by id.
crawl_file	
crawl_file_root	
crawl_file_nested_1	
crawl_file_nested_2	
patch_crawl	Update a crawl (e.g., set status=sealed to cancel queued work).
delete_crawl	

Data

router

API

archivebox.api.v1_crawls.**router**

'Router(...)'

class archivebox.api.v1_crawls.CrawlSchema

Bases: ninja.Schema

TYPE: str

'crawls.models.Crawl'

id: uuid.UUID

None

modified_at: datetime.datetime

None

created_at: datetime.datetime

None

created_by_id: str

None

created_by_username: str

None

status: str

None

retry_at: datetime.datetime | None

None

is_paused: bool

None

urls: str

None

max_depth: int

None

tags_str: str

None

config: dict

None

static resolve_created_by_id(*obj*)

static resolve_created_by_username(*obj*)

static resolve_config(*obj*)

static resolve_snapshots(*obj*, *context*)

class archivebox.api.v1_crawls.CrawlUpdateSchema

Bases: ninja.Schema

action: str | None

None

status: str | None

None

retry_at: datetime.datetime | None

None

tags: list[str] | None

None

tags_str: str | None

None

class archivebox.api.v1_crawls.**CrawlCreateSchema**

Bases: ninja.Schema

urls: list[str]

None

max_depth: int

0

tags: list[str] | None

None

tags_str: str = <Multiline-String>

label: str = <Multiline-String>

notes: str = <Multiline-String>

config: dict

None

class archivebox.api.v1_crawls.**CrawlDeleteResponseSchema**

Bases: ninja.Schema

success: bool

None

crawl_id: str

None

deleted_count: int

None

deleted_snapshots: int

None

archivebox.api.v1_crawls.**normalize_tag_list**(*tags: list[str] | None = None*, *tags_str: str = ''*) → list[str]

archivebox.api.v1_crawls.**get_crawls**(*request: django.http.HttpRequest*)

archivebox.api.v1_crawls.**create_crawl**(*request: django.http.HttpRequest*, *data:* archivebox.api.v1_crawls.CrawlCreateSchema)

archivebox.api.v1_crawls.**get_crawl_by_ref**(*crawl_id: str*)

archivebox.api.v1_crawls.**get_crawl**(*request: django.http.HttpRequest*, *crawl_id: str*, *as_rss: bool = False*, *with_snapshots: bool = False*, *with_archiveresults: bool = False*)

Get a specific Crawl by id.

archivebox.api.v1_crawls.**crawl_file**(*request: django.http.HttpRequest*, *crawl_id: str*, *path: str*)

archivebox.api.v1_crawls.**crawl_file_root**(*request: django.http.HttpRequest*, *crawl_id: str*, *filename: str*)

archivebox.api.v1_crawls.**crawl_file_nested_1**(*request: django.http.HttpRequest*, *crawl_id: str*, *folder: str*, *filename: str*)

archivebox.api.v1_crawls.**crawl_file_nested_2**(*request: django.http.HttpRequest*, *crawl_id: str*, *folder: str*, *subfolder: str*, *filename: str*)

archivebox.api.v1_crawls.**patch_crawl**(*request: django.http.HttpRequest*, *crawl_id: str*, *data:* archivebox.api.v1_crawls.CrawlUpdateSchema)

Update a crawl (e.g., set status=sealed to cancel queued work).

archivebox.api.v1_crawls.**delete_crawl**(*request: django.http.HttpRequest*, *crawl_id: str*)

archivebox.api.urls

Module Contents

Functions

archive_redirect_view

Data

urlpatterns

API

archivebox.api.urls.**archive_redirect_view**(*request: django.http.HttpRequest*, *url: str*) → django.http.HttpResponseRedirect

archivebox.api.urls.**urlpatterns**

None

archivebox.api.webhooks

Module Contents

Functions

load_deferred_fields_for_delete_webhook	Materialize deferred fields while DELETE webhook rows still exist.
register_delete_webhook_field_loader	
warning_error_handler	
transaction_on_commit_task_handler	

Data

logger

API

archivebox.api.webhooks.**logger**

'getLogger(...)'

archivebox.api.webhooks.**load_deferred_fields_for_delete_webhook**(*sender*, *instance*, *using*, ***kwargs*) → None

Materialize deferred fields while DELETE webhook rows still exist.

Admin lists defer large fields, but signal_webhooks serializes all concrete fields in post_delete. Loading one then would query an already-deleted row and drop the event. Hydrate only deferred fields on webhook-enabled models here, preserving the normal list-query optimization.

archivebox.api.webhooks.**register_delete_webhook_field_loader**() → None

archivebox.api.webhooks.**warning_error_handler**(*hook: Any*, *error: Exception | None*) → None

archivebox.api.webhooks.**transaction_on_commit_task_handler**(*hook: collections.abc.Callable[..., None]*, ***kwargs: Any*) → None

archivebox.api.v1_api

Module Contents

Classes

NinjaAPIWithIOCapture

Functions

register_urls
generic_exception_handler

Data

COMMIT_HASH
html_description
api
urls

API

archivebox.api.v1_api.**COMMIT_HASH**

None

archivebox.api.v1_api.**html_description**

None

archivebox.api.v1_api.**register_urls**(*api: ninja.NinjaAPI*) → ninja.NinjaAPI

class archivebox.api.v1_api.**NinjaAPIWithIOCapture**

Bases: ninja.NinjaAPI

get_openapi_schema(*, *path_prefix: str | None = None*, *path_params: dict[str, Any] | None = None*) → ninja.openapi.schema.OpenAPISchema

_schema_for_prefix(*path_prefix: str*) → ninja.openapi.schema.OpenAPISchema

create_temporal_response(*request: django.http.HttpRequest*) → django.http.HttpResponse

archivebox.api.v1_api.**api**

'NinjaAPIWithIOCapture(...)'

archivebox.api.v1_api.**urls**

None

archivebox.api.v1_api.**generic_exception_handler**(*request*, *err*)

archivebox.api.v1_personas

Module Contents

Classes

PersonaBrowserSettingsSchema	
PersonaSyncSchema	
PersonaSchema	
PersonaSyncResponseSchema	

Functions

browser_settings_to_config	
find_persona	
get_personas	List personas available on this ArchiveBox server.
sync_persona	Create or update a Persona from a browser extension profile export.

Data

router

API

archivebox.api.v1_personas.**router**

'Router(...)'

class archivebox.api.v1_personas.**PersonaBrowserSettingsSchema**

Bases: ninja.Schema

user_agent: str = <Multiline-String>

viewport_size: str = <Multiline-String>

viewport_device_scale_factor: float | None

None

color_scheme: Literal[, light, dark] = <Multiline-String>

language: str = <Multiline-String>

timezone: str = <Multiline-String>

geolocation: dict[str, Any] | None

None

class archivebox.api.v1_personas.**PersonaSyncSchema**

Bases: ninja.Schema

extension_persona_id: str

None

name: str

None

settings: *archivebox.api.v1_personas.PersonaBrowserSettingsSchema*

'Field(...)'

cookies_txt: str = <Multiline-String>

auth_json: dict[str, Any]

'Field(...)'

class archivebox.api.v1_personas.**PersonaSchema**

Bases: ninja.Schema

TYPE: str

'personas.models.Persona'

id: uuid.UUID

None

name: str

None

created_at: datetime.datetime

None

created_by_id: str

None

created_by_username: str

None

config: dict[str, Any] | None

None

static resolve_created_by_id(*obj*)

static resolve_created_by_username(*obj*) → str

static resolve_config(*obj*)

class archivebox.api.v1_personas.**PersonaSyncResponseSchema**

Bases: ninja.Schema

success: **bool**

None

created: **bool**

None

persona: *archivebox.api.v1_personas.PersonaSchema*

None

cookies_file_written: **bool**

None

auth_file_written: **bool**

None

archivebox.api.v1_personas.**browser_settings_to_config**(*extension_persona_id: str*, *settings:* archivebox.api.v1_personas.PersonaBrowserSettingsSchema) → dict[str, Any]

archivebox.api.v1_personas.**find_persona**(*extension_persona_id: str*, *name: str*) → *archivebox.personas.models.Persona* | None

archivebox.api.v1_personas.**get_personas**(*request: django.http.HttpRequest*)

List personas available on this ArchiveBox server.

archivebox.api.v1_personas.**sync_persona**(*request: django.http.HttpRequest*, *payload:* archivebox.api.v1_personas.PersonaSyncSchema)

Create or update a Persona from a browser extension profile export.

The extension sends browser settings plus portable auth artifacts. The server keeps browser override settings in Persona.config and writes cookies.txt / auth.json into the persona directory for extractors to consume.

archivebox.api.v1_core

Module Contents

Classes

CustomPagination
MinimalArchiveResultSchema
ArchiveResultSchema
ArchiveResultFilterSchema
SnapshotSchema
SnapshotUpdateSchema
SnapshotCreateSchema
SnapshotDeleteResponseSchema

continues on next page

Table 107 – continued from previous page

SnapshotFilterSchema
TagSchema
TagAutocompleteSchema
TagCreateSchema
TagCreateResponseSchema
TagSearchSnapshotSchema
TagSearchCardSchema
TagSearchResponseSchema
TagUpdateSchema
TagUpdateResponseSchema
TagDeleteResponseSchema
TagSnapshotRequestSchema
TagSnapshotResponseSchema

Functions

_with_archiveresult_processes	
get_archiveresults	List all ArchiveResult entries matching these filters.
_uuid_ref_query	
get_archiveresult	Get a specific ArchiveResult by id.
_normalize_uploaded_archiveresult_plugin	
_normalize_uploaded_archiveresult_output_pat	
_parse_archiveresult_output_json	
_get_archiveresult_upload_data	
_get_archiveresult_upload_files	
_get_archiveresult_upload_form_values	
_get_archiveresult_upload_form_value	
_parse_archiveresult_upload_int	
_summarize_archiveresult_output_files	
_get_snapshot_by_ref	
_queue_archiveresult_snapshot_maintenance	Mark an uploaded ArchiveResult's Snapshot as dirty without finalizing it.
_write_archiveresult_files	
create_archiveresult	Create or update an ArchiveResult with one or more output files.
patch_archiveresult	Append or replace files on an existing ArchiveResult.
normalize_tag_list	
_parse_rss_before	
_filter_snapshots_for_rss	
_snapshots_rss_response	
get_snapshots	List all Snapshot entries matching these filters.
get_snapshots_rss	Return matching snapshots as an RSS feed, newest first.
get_snapshot	Get a specific Snapshot by id.
create_snapshot	
patch_snapshot	Update a snapshot (e.g., set status=sealed to cancel queued work).
delete_snapshot	Queue deletion; the normal server runner stops writers and removes files.
get_tags	
get_tag	

continues on next page

Table 108 – continued from previous page

get_any	Get any object by its ID (e.g. snapshot, archiveresult, tag, crawl, etc.).
_get_snapshot_for_tag_edit	
search_tags	Return detailed tag cards for admin/live-search UIs.
_public_tag_listing_enabled	
_request_has_tag_autocomplete_access	
tags_autocomplete	Return tags matching the query for autocomplete.
tags_create	Create a new tag or return existing one.
rename_tag	
delete_tag	
tag_urls_export	
tag_snapshots_export	
tags_add_to_snapshot	Add a tag to a snapshot. Creates the tag if it doesn't exist.
tags_remove_from_snapshot	Remove a tag from a snapshot.

Data

router
ARCHIVERESULT_UPLOAD_HOOK_NAME
ARCHIVERESULT_UPLOAD_PLUGIN_RE

API

archivebox.api.v1_core.**router**

'Router(...)'

archivebox.api.v1_core.**ARCHIVERESULT_UPLOAD_HOOK_NAME**

None

archivebox.api.v1_core.**ARCHIVERESULT_UPLOAD_PLUGIN_RE**

'compile(...)'

class archivebox.api.v1_core.**CustomPagination**

Bases: ninja.pagination.PaginationBase

class Input

Bases: ninja.pagination.PaginationBase.Input

limit: int

200

offset: int

0

page: int

0

class Output

Bases: ninja.pagination.PaginationBase.Output

count: int

None

total_items: int
None

total_pages: int
None

page: int
None

limit: int
None

offset: int
None

num_items: int
None

items: list[Any]
None

paginate_queryset(*queryset*, *pagination:* Input, *request: django.http.HttpRequest*, ***params*)

class archivebox.api.v1_core.**MinimalArchiveResultSchema**

Bases: ninja.Schema

TYPE: str
'core.models.ArchiveResult'

id: uuid.UUID
None

created_at: datetime.datetime | None
None

modified_at: datetime.datetime | None
None

created_by_id: str
None

created_by_username: str
None

status: str
None

retry_at: datetime.datetime | None
None

plugin: str
None

hook_name: str
None

process_id: uuid.UUID | None
None

cmd_version: str | None
None

cmd: list[str] | None
None

pwd: str | None
None

output_str: str
None

output_json: dict[str, Any] | None
None

output_files: dict[str, dict[str, Any]] | None
None

output_size: int
None

output_mimetypes: str
None

start_ts: datetime.datetime | None
None

end_ts: datetime.datetime | None
None

static resolve_created_by_id(*obj*)

static resolve_created_by_username(*obj*) → str

static resolve_output_files(*obj*)

static resolve_output_mimetypes(*obj*) → str

class archivebox.api.v1_core.**ArchiveResultSchema**
Bases: *archivebox.api.v1_core.MinimalArchiveResultSchema*

TYPE: str
'core.models.ArchiveResult'

snapshot_id: uuid.UUID
None

snapshot_timestamp: str
None

snapshot_url: str
None

snapshot_tags: list[str]
None

static resolve_snapshot_timestamp(*obj*)

static resolve_snapshot_url(*obj*)

static resolve_snapshot_id(*obj*)

static resolve_snapshot_tags(*obj*)

class archivebox.api.v1_core.ArchiveResultFilterSchema

Bases: ninja.FilterSchema

id: Annotated[str | None, FilterLookup(['id__startswith', 'snapshot__id__startswith', 'snapshot__timestamp__startswith'])]

None

search: Annotated[str | None, FilterLookup(['snapshot__url__icontains', 'snapshot__title__icontains', 'snapshot__tags__name__icontains', 'plugin', 'output_str__icontains', 'id__startswith', 'snapshot__id__startswith', 'snapshot__timestamp__startswith'])]

None

snapshot_id: Annotated[str | None, FilterLookup(['snapshot__id__startswith', 'snapshot__timestamp__startswith'])]

None

snapshot_url: Annotated[str | None, FilterLookup('snapshot__url__icontains')]

None

snapshot_tag: Annotated[str | None, FilterLookup('snapshot__tags__name__icontains')]

None

status: Annotated[str | None, FilterLookup('status')]

None

output_str: Annotated[str | None, FilterLookup('output_str__icontains')]

None

plugin: Annotated[str | None, FilterLookup('plugin__icontains')]

None

hook_name: Annotated[str | None, FilterLookup('hook_name__icontains')]

None

process_id: Annotated[str | None, FilterLookup('process__id__startswith')]

None

cmd: Annotated[str | None, FilterLookup('cmd__0__icontains')]

None

pwd: Annotated[str | None, FilterLookup('pwd__icontains')]

None

cmd_version: Annotated[str | None, FilterLookup('cmd_version')]

None

created_at: Annotated[datetime.datetime | None, FilterLookup('created_at')]

None

created_at__gte: Annotated[datetime.datetime | None, FilterLookup('created_at__gte')]

None

created_at__lt: Annotated[datetime.datetime | None, FilterLookup('created_at__lt')]
None

archivebox.api.v1_core.**_with_archiveresult_processes**(*queryset*)

archivebox.api.v1_core.**get_archiveresults**(*request: django.http.HttpRequest*, *filters: ninja.Query[archivebox.api.v1_core.ArchiveResultFilterSchema]*)

List all ArchiveResult entries matching these filters.

archivebox.api.v1_core.**_uuid_ref_query**(*field_name: str*, *ref: str*) → django.db.models.Q

archivebox.api.v1_core.**get_archiveresult**(*request: django.http.HttpRequest*, *archiveresult_id: str*)

Get a specific ArchiveResult by id.

archivebox.api.v1_core.**_normalize_uploaded_archiveresult_plugin**(*plugin: str*) → str

archivebox.api.v1_core.**_normalize_uploaded_archiveresult_output_path**(*output_path: str*, ***, *filename: str*) → str

archivebox.api.v1_core.**_parse_archiveresult_output_json**(*output_json: str | None*) → dict[str, Any] | None

archivebox.api.v1_core.**_get_archiveresult_upload_data**(*request: django.http.HttpRequest*)

archivebox.api.v1_core.**_get_archiveresult_upload_files**(*request: django.http.HttpRequest*, ***, *allow_empty: bool = False*) → list[ninja.UploadedFile]

archivebox.api.v1_core.**_get_archiveresult_upload_form_values**(*request: django.http.HttpRequest*, **field_names: str*) → list[str]

archivebox.api.v1_core.**_get_archiveresult_upload_form_value**(*request: django.http.HttpRequest*, **field_names: str*) → str

archivebox.api.v1_core.**_parse_archiveresult_upload_int**(*value: str*, *field_name: str*, ***, *default: int | None = None*) → int

archivebox.api.v1_core.**_summarize_archiveresult_output_files**(*output_files: dict[str, dict[str, Any]]*) → tuple[int, str]

archivebox.api.v1_core.**_get_snapshot_by_ref**(*snapshot_id: str*)

archivebox.api.v1_core.**_queue_archiveresult_snapshot_maintenance**(*snapshot: archivebox.core.models.Snapshot*) → None

Mark an uploaded ArchiveResult's Snapshot as dirty without finalizing it.

Upload API handlers are allowed to persist files and ArchiveResult rows, but Snapshot save() side effects, sealing, symlink creation, and index/details rewrites belong to the runner. retry_at is the scheduler signal the runner already watches, so only bump rows that are final or otherwise invisible.

archivebox.api.v1_core.**_write_archiveresult_files**(*request: django.http.HttpRequest*, *snapshot: archivebox.core.models.Snapshot*, *plugin_name: str*, ***, *existing_output_files: dict[str, dict[str, Any]] | None = None*, *allow_empty: bool = False*) → dict[str, dict[str, Any]]

archivebox.api.v1_core.**create_archiveresult**(*request: django.http.HttpRequest*, *snapshot_id: str = Form(...)*, *plugin: str = Form(...)*, *output_str: str = Form('')*, *hook_name: str = Form(ARCHIVERESULT_UPLOAD_HOOK_NAME)*, *status: str = Form(str(ArchiveResult.StatusChoices.SUCCEEDED))*, *output_json: str = Form('')*)

Create or update an ArchiveResult with one or more output files.

archivebox.api.v1_core.**patch_archiveresult**(*request: django.http.HttpRequest*, *archiveresult_id: str*)

Append or replace files on an existing ArchiveResult.

class archivebox.api.v1_core.**SnapshotSchema**

Bases: ninja.Schema

TYPE: str

'core.models.Snapshot'

id: uuid.UUID

None

crawl_id: uuid.UUID

None

persona: str | None

None

created_by_id: str

None

created_by_username: str

None

created_at: datetime.datetime

None

modified_at: datetime.datetime

None

status: str

None

retry_at: datetime.datetime | None

None

bookmarked_at: datetime.datetime

None

downloaded_at: datetime.datetime | None

None

url: str

None

tags: list[str]

None

title: str | None

None

timestamp: str

None

archive_path: str

None

archive_size: int

None

output_size: int

None

num_archiveresults: int

None

archiveresults: list[*archivebox.api.v1_core.MinimalArchiveResultSchema*]

None

static resolve_archive_path(*obj*)

static resolve_persona(*obj*)

static resolve_created_by_id(*obj*)

static resolve_created_by_username(*obj*)

static resolve_tags(*obj*)

static resolve_archive_size(*obj*)

static resolve_output_size(*obj*)

static resolve_num_archiveresults(*obj*, *context*)

static resolve_archiveresults(*obj*, *context*)

class archivebox.api.v1_core.**SnapshotUpdateSchema**

Bases: ninja.Schema

action: str | None

None

status: str | None

None

retry_at: datetime.datetime | None

None

tags: list[str] | None

None

class archivebox.api.v1_core.**SnapshotCreateSchema**

Bases: ninja.Schema

url: str

None

crawl_id: str | None

None

depth: int

0

title: str | None

None

tags: list[str] | None

None

status: str | None

None

class archivebox.api.v1_core.**SnapshotDeleteResponseSchema**

Bases: ninja.Schema

success: bool

None

snapshot_id: str

None

crawl_id: str

None

deleted_count: int

None

queued_count: int

None

archivebox.api.v1_core.**normalize_tag_list**(*tags: list[str] | None = None*) → list[str]

archivebox.api.v1_core.**_parse_rss_before**(*before: str | None*) → datetime.datetime

archivebox.api.v1_core.**_filter_snapshots_for_rss**(**, crawl_id: str = '', created_by: str = '', before: str | None = None, limit: int = 50*)

archivebox.api.v1_core.**_snapshots_rss_response**(*request: django.http.HttpRequest, *, snapshots, title: str = 'ArchiveBox Snapshots'*) → django.http.HttpResponse

class archivebox.api.v1_core.**SnapshotFilterSchema**

Bases: ninja.FilterSchema

id: Annotated[str | None, FilterLookup(['id__istartswith', 'id__iendswith', 'timestamp__startswith'])]

None

created_by_id: Annotated[str | None, FilterLookup('crawl__created_by_id')]

None

created_by_username: Annotated[str | None, FilterLookup('crawl__created_by__username__icontains')]

None

created_at__gte: Annotated[datetime.datetime | None, FilterLookup('created_at__gte')]

None

created_at__lt: Annotated[datetime.datetime | None, FilterLookup('created_at__lt')]

None

created_at: Annotated[datetime.datetime | None, FilterLookup('created_at')]

None

modified_at: Annotated[datetime.datetime | None, FilterLookup('modified_at')]

None

modified_at__gte: Annotated[datetime.datetime | None, FilterLookup('modified_at__gte')]

None

modified_at__lt: Annotated[datetime.datetime | None, FilterLookup('modified_at__lt')]

None

search: str | None

None

search_mode: str | None

None

status: str | None

None

url: Annotated[str | None, FilterLookup('url')]

None

tag: Annotated[str | None, FilterLookup('tags__name')]

None

title: Annotated[str | None, FilterLookup('title__icontains')]

None

timestamp: Annotated[str | None, FilterLookup('timestamp__startswith')]

None

bookmarked_at__gte: Annotated[datetime.datetime | None, FilterLookup('bookmarked_at__gte')]

None

bookmarked_at__lt: Annotated[datetime.datetime | None, FilterLookup('bookmarked_at__lt')]

None

filter_search(*value: str | None*) → django.db.models.Q

filter_search_mode(*value: str | None*) → django.db.models.Q

filter_status(*value: str | None*) → django.db.models.Q

archivebox.api.v1_core.**get_snapshots**(*request: django.http.HttpRequest*, *filters: ninja.Query[archivebox.api.v1_core.SnapshotFilterSchema]*, *with_archiveresults: bool = False*)

List all Snapshot entries matching these filters.

archivebox.api.v1_core.**get_snapshots_rss**(*request: django.http.HttpRequest*, *crawl_id: str = ''*, *created_by: str = ''*, *limit: int = 50*, *before: str | None = None*)

Return matching snapshots as an RSS feed, newest first.

archivebox.api.v1_core.**get_snapshot**(*request: django.http.HttpRequest*, *snapshot_id: str*, *with_archiveresults: bool = True*)

Get a specific Snapshot by id.

archivebox.api.v1_core.**create_snapshot**(*request: django.http.HttpRequest*, *data: archivebox.api.v1_core.SnapshotCreateSchema*)

archivebox.api.v1_core.**patch_snapshot**(*request: django.http.HttpRequest*, *snapshot_id: str*, *data: archivebox.api.v1_core.SnapshotUpdateSchema*)

Update a snapshot (e.g., set status=sealed to cancel queued work).

archivebox.api.v1_core.**delete_snapshot**(*request: django.http.HttpRequest*, *snapshot_id: str*)

Queue deletion; the normal server runner stops writers and removes files.

The snapshot remains readable with status=deleting until cleanup succeeds. queued_count reports acceptance; deleted_count is zero in this response.

class archivebox.api.v1_core.**TagSchema**

Bases: ninja.Schema

TYPE: str

'core.models.Tag'

id: int

None

modified_at: datetime.datetime

None

created_at: datetime.datetime

None

created_by_id: str

None

created_by_username: str

None

name: str

None

num_snapshots: int

None

snapshots: list[*archivebox.api.v1_core.SnapshotSchema*]

None

static resolve_created_by_id(*obj*)

static **resolve_created_by_username**(*obj*)

static **resolve_num_snapshots**(*obj*, *context*)

static **resolve_snapshots**(*obj*, *context*)

archivebox.api.v1_core.**get_tags**(*request: django.http.HttpRequest*)

archivebox.api.v1_core.**get_tag**(*request: django.http.HttpRequest*, *tag_id: str*, *with_snapshots: bool = True*)

archivebox.api.v1_core.**get_any**(*request: django.http.HttpRequest*, *id: str*)

Get any object by its ID (e.g. snapshot, archiveresult, tag, crawl, etc.).

class archivebox.api.v1_core.**TagAutocompleteSchema**

Bases: ninja.Schema

tags: list[dict]

None

class archivebox.api.v1_core.**TagCreateSchema**

Bases: ninja.Schema

name: str

None

class archivebox.api.v1_core.**TagCreateResponseSchema**

Bases: ninja.Schema

success: bool

None

tag_id: int

None

tag_name: str

None

created: bool

None

class archivebox.api.v1_core.**TagSearchSnapshotSchema**

Bases: ninja.Schema

id: str

None

title: str

None

url: str

None

favicon_url: str

None

admin_url: str

None

archive_url: str
None

downloaded_at: str | None
None

class archivebox.api.v1_core.TagSearchCardSchema
Bases: ninja.Schema

id: int
None

name: str
None

slug: str
None

num_snapshots: int
None

filter_url: str
None

edit_url: str
None

export_urls_url: str
None

export_jsonl_url: str
None

rename_url: str
None

delete_url: str
None

snapshots: list[archivebox.api.v1_core.TagSearchSnapshotSchema]
None

class archivebox.api.v1_core.TagSearchResponseSchema
Bases: ninja.Schema

tags: list[archivebox.api.v1_core.TagSearchCardSchema]
None

sort: str
None

created_by: str
None

year: str
None

has_snapshots: str
None

class archivebox.api.v1_core.**TagUpdateSchema**

Bases: ninja.Schema

name: str

None

class archivebox.api.v1_core.**TagUpdateResponseSchema**

Bases: ninja.Schema

success: bool

None

tag_id: int

None

tag_name: str

None

class archivebox.api.v1_core.**TagDeleteResponseSchema**

Bases: ninja.Schema

success: bool

None

tag_id: int

None

deleted_count: int

None

class archivebox.api.v1_core.**TagSnapshotRequestSchema**

Bases: ninja.Schema

snapshot_id: str

None

tag_name: str | None

None

tag_id: int | None

None

class archivebox.api.v1_core.**TagSnapshotResponseSchema**

Bases: ninja.Schema

success: bool

None

tag_id: int

None

tag_name: str

None

archivebox.api.v1_core.**_get_snapshot_for_tag_edit**(*snapshot_ref: str*) → *archivebox.core.models.Snapshot*

archivebox.api.v1_core.**search_tags**(*request: django.http.HttpRequest*, *q: str = ''*, *sort: str = 'created_desc'*, *created_by: str = ''*, *year: str = ''*, *has_snapshots: str = 'all'*)

Return detailed tag cards for admin/live-search UIs.

archivebox.api.v1_core.**_public_tag_listing_enabled**() → bool

archivebox.api.v1_core.**_request_has_tag_autocomplete_access**(*request: django.http.HttpRequest*) → bool

archivebox.api.v1_core.**tags_autocomplete**(*request: django.http.HttpRequest*, *q: str = ''*)

Return tags matching the query for autocomplete.

archivebox.api.v1_core.**tags_create**(*request: django.http.HttpRequest*, *data:* archivebox.api.v1_core.TagCreateSchema)

Create a new tag or return existing one.

archivebox.api.v1_core.**rename_tag**(*request: django.http.HttpRequest*, *tag_id: int*, *data:* archivebox.api.v1_core.TagUpdateSchema)

archivebox.api.v1_core.**delete_tag**(*request: django.http.HttpRequest*, *tag_id: int*)

archivebox.api.v1_core.**tag_urls_export**(*request: django.http.HttpRequest*, *tag_id: int*)

archivebox.api.v1_core.**tag_snapshots_export**(*request: django.http.HttpRequest*, *tag_id: int*)

archivebox.api.v1_core.**tags_add_to_snapshot**(*request: django.http.HttpRequest*, *data:* archivebox.api.v1_core.TagSnapshotRequestSchema)

Add a tag to a snapshot. Creates the tag if it doesn't exist.

archivebox.api.v1_core.**tags_remove_from_snapshot**(*request: django.http.HttpRequest*, *data:* archivebox.api.v1_core.TagSnapshotRequestSchema)

Remove a tag from a snapshot.

archivebox.api.models

Module Contents

Classes

APIToken
OutboundWebhook

Functions

generate_secret_token

API

archivebox.api.models.**generate_secret_token**() → str

class archivebox.api.models.**APIToken**

Bases: `django.db.models.Model`

id

'CompactUUIDField(...)'

created_by

'ForeignKey(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

token

'CharField(...)'

expires

'DateTimeField(...)'

class Meta

Bases: `django_stubs_ext.db.models.TypedModelMeta`

app_label

'api'

verbose_name

'API Key'

verbose_name_plural

'API Keys'

__str__() → str

property token_redacted

is_valid(*for_date=None*)

class archivebox.api.models.**OutboundWebhook**

Bases: `signal_webhooks.models.WebhookBase`

id

'CompactUUIDField(...)'

created_by

'ForeignKey(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

class Meta

Bases: `signal_webhooks.models.WebhookBase.Meta`

app_label

'api'

verbose_name

'API Outbound Webhook'

__str__() → str

archivebox.api.v1_auth

Module Contents

Classes

PasswordAuthSchema	Schema for a /get_api_token request
TokenAuthSchema	Schema for a /check_api_token request

Functions

get_api_token
check_api_token
browser_session

Data

router

API

archivebox.api.v1_auth.**router**

'Router(...)'

class archivebox.api.v1_auth.**PasswordAuthSchema**

Bases: ninja.Schema

Schema for a /get_api_token request

username: str | None

None

password: str | None

None

archivebox.api.v1_auth.**get_api_token**(*request: django.http.HttpRequest*, *auth_data:* archivebox.api.v1_auth.PasswordAuthSchema)

class archivebox.api.v1_auth.**TokenAuthSchema**

Bases: ninja.Schema

Schema for a /check_api_token request

token: str

None

archivebox.api.v1_auth.**check_api_token**(*request: django.http.HttpRequest*, *token_data:* archivebox.api.v1_auth.TokenAuthSchema)

archivebox.api.v1_auth.**browser_session**(*request: django.http.HttpRequest*)

archivebox.api.middleware

Module Contents

Classes

ApiCorsMiddleware	Attach permissive CORS headers for API routes (token-based auth).

API

class archivebox.api.middleware.**ApiCorsMiddleware**(*get_response*)

Attach permissive CORS headers for API routes (token-based auth).

Initialization

__call__(*request*)

_add_cors_headers(*request*, *response*)

archivebox.api.auth

Module Contents

Classes

HeaderTokenAuth	Allow authenticating by passing X-API-Key=xyz as a request header
BearerTokenAuth	Allow authenticating by passing Bearer=xyz as a request header
QueryParamTokenAuth	Allow authenticating by passing api_key=xyz as a GET/POST query parameter

Functions

get_or_create_api_token	
auth_using_token	Given an API token string, check if a corresponding non-expired APIToken exists, and return its user
token_from_request	
authenticated_user_from_request	
auth_using_password	Given a username and password, check if they are valid and return the corresponding user
_require_superuser	

Data

API_AUTH_METHODS

API

archivebox.api.auth.**get_or_create_api_token**(*user: django.contrib.auth.models.User | None*)

archivebox.api.auth.**auth_using_token**(*token: str | None*, *request: django.http.HttpRequest | None = None*) → django.contrib.auth.models.User | None

Given an API token string, check if a corresponding non-expired APIToken exists, and return its user

archivebox.api.auth.**token_from_request**(*request: django.http.HttpRequest*) → str

archivebox.api.auth.**authenticated_user_from_request**(*request: django.http.HttpRequest*) → django.contrib.auth.models.User | None

archivebox.api.auth.**auth_using_password**(*username: str | None*, *password: str | None*, *request: django.http.HttpRequest | None = None*) → django.contrib.auth.models.User | None

Given a username and password, check if they are valid and return the corresponding user

archivebox.api.auth.**_require_superuser**(*user: django.contrib.auth.models.User | None*, *request: django.http.HttpRequest*, *auth_method: str*) → django.contrib.auth.models.User | None

class archivebox.api.auth.**HeaderTokenAuth**

Bases: ninja.security.APIKeyHeader

Allow authenticating by passing X-API-Key=xyz as a request header

param_name

'X-ArchiveBox-API-Key'

authenticate(*request: django.http.HttpRequest*, *key: str | None*) → django.contrib.auth.models.User | None

class archivebox.api.auth.**BearerTokenAuth**

Bases: ninja.security.HttpBearer

Allow authenticating by passing Bearer=xyz as a request header

authenticate(*request: django.http.HttpRequest*, *token: str*) → django.contrib.auth.models.User | None

class archivebox.api.auth.**QueryParamTokenAuth**

Bases: ninja.security.APIKeyQuery

Allow authenticating by passing api_key=xyz as a GET/POST query parameter

param_name

'api_key'

authenticate(*request: django.http.HttpRequest*, *key: str | None*) → django.contrib.auth.models.User | None

archivebox.api.auth.**API_AUTH_METHODS**

None

archivebox.api.v1_machine

Module Contents

Classes

MachineSchema	Schema for Machine model.
MachineFilterSchema	
BinarySchema	Schema for Binary model.
BinaryFilterSchema	

Functions

get_machines	List all machines.
get_current_machine	Get the current machine.
get_machine	Get a specific machine by ID.
get_binaries	List all binaries.
get_binary	Get a specific binary by ID.
get_binaries_by_name	Get all binaries with the given name.

Data

router

API

archivebox.api.v1_machine.**router**

'Router(...)'

class archivebox.api.v1_machine.**MachineSchema**

Bases: ninja.Schema

Schema for Machine model.

TYPE: str

'machine.Machine'

id: uuid.UUID

None

created_at: datetime.datetime

None

modified_at: datetime.datetime

None

guid: str

None

hostname: str

None

hw_in_docker: bool

None

hw_in_vm: bool

None

hw_manufacturer: str
None

hw_product: str
None

hw_uuid: str
None

os_arch: str
None

os_family: str
None

os_platform: str
None

os_release: str
None

os_kernel: str
None

stats: dict
None

num_uses_succeeded: int
None

num_uses_failed: int
None

class archivebox.api.v1_machine.MachineFilterSchema
Bases: ninja.FilterSchema

id: Annotated[str | None, FilterLookup('id__startswith')]
None

hostname: Annotated[str | None, FilterLookup('hostname__icontains')]
None

os_platform: Annotated[str | None, FilterLookup('os_platform__icontains')]
None

os_arch: Annotated[str | None, FilterLookup('os_arch')]
None

hw_in_docker: Annotated[bool | None, FilterLookup('hw_in_docker')]
None

hw_in_vm: Annotated[bool | None, FilterLookup('hw_in_vm')]
None

bin_providers: Annotated[str | None, FilterLookup('bin_providers__icontains')]
None

class archivebox.api.v1_machine.**BinarySchema**

Bases: ninja.Schema

Schema for Binary model.

TYPE: str

'machine.Binary'

id: uuid.UUID

None

created_at: datetime.datetime

None

modified_at: datetime.datetime

None

machine_id: uuid.UUID

None

machine_hostname: str

None

name: str

None

binproviders: str

None

binprovider: str

None

abspath: str

None

version: str

None

sha256: str

None

status: str

None

is_valid: bool

None

num_uses_succeeded: int

None

num_uses_failed: int

None

static resolve_machine_hostname(*obj*) → str

static resolve_is_valid(*obj*) → bool

class archivebox.api.v1_machine.**BinaryFilterSchema**

Bases: ninja.FilterSchema

id: `Annotated[str | None, FilterLookup('id__startswith')]`

None

name: `Annotated[str | None, FilterLookup('name__icontains')]`

None

binprovider: `Annotated[str | None, FilterLookup('binprovider')]`

None

status: `Annotated[str | None, FilterLookup('status')]`

None

machine_id: `Annotated[str | None, FilterLookup('machine_id__startswith')]`

None

version: `Annotated[str | None, FilterLookup('version__icontains')]`

None

archivebox.api.v1_machine.**get_machines**(*request: django.http.HttpRequest*, *filters: ninja.Query[*archivebox.api.v1_machine.MachineFilterSchema*]*)

List all machines.

archivebox.api.v1_machine.**get_current_machine**(*request: django.http.HttpRequest*)

Get the current machine.

archivebox.api.v1_machine.**get_machine**(*request: django.http.HttpRequest*, *machine_id: str*)

Get a specific machine by ID.

archivebox.api.v1_machine.**get_binaries**(*request: django.http.HttpRequest*, *filters: ninja.Query[*archivebox.api.v1_machine.BinaryFilterSchema*]*)

List all binaries.

archivebox.api.v1_machine.**get_binary**(*request: django.http.HttpRequest*, *binary_id: str*)

Get a specific binary by ID.

archivebox.api.v1_machine.**get_binaries_by_name**(*request: django.http.HttpRequest*, *name: str*)

Get all binaries with the given name.

archivebox.api.v1_cli

Module Contents

Classes

CLICommandResponseSchema
AddCommandSchema
SnapshotFilterCommandSchema
UpdateCommandSchema
ScheduleCommandSchema
ListCommandSchema
RemoveCommandSchema

Functions

snapshot_filter_kwargs
cli_add
cli_update
cli_schedule
cli_search
cli_remove

Data

router
JSONType
FILTER_PATTERNS_EXAMPLES
FilterTypeChoices

API

archivebox.api.v1_cli.**router**

'Router(...)'

archivebox.api.v1_cli.**JSONType**

None

archivebox.api.v1_cli.**FILTER_PATTERNS_EXAMPLES**

[['https://example.com']]

class archivebox.api.v1_cli.**CLICommandResponseSchema**

Bases: ninja.Schema

success: bool

None

errors: list[str]

None

result: *archivebox.api.v1_cli.JSONType*

None

result_format: str

'str'

stdout: str

None

stderr: str

None

archivebox.api.v1_cli.**FilterTypeChoices**

'Enum(...)'

class archivebox.api.v1_cli.**AddCommandSchema**

Bases: ninja.Schema

urls: list[str]
None

snapshot_ids: list[str] | None
None

tag: str = <Multiline-String>

depth: int
0

max_urls: int
0

crawl_max_size: int
0

crawl_timeout: int
0

snapshot_max_size: int
0

parser: str
'auto'

plugins: str = <Multiline-String>

persona: str
'Default'

only_new: bool | None
None

update: bool
False

overwrite: bool
False

index_only: bool
False

class archivebox.api.v1_cli.SnapshotFilterCommandSchema
Bases: ninja.Schema

after: float | None
0

before: float | None
None

filter_type: str | None
None

filter_patterns: list[str] | None
'Field(...)'

status: str | None
None

url__icontains: str | None
None

url__istartswith: str | None
None

tag: str | None
None

crawl_id: str | None
None

limit: int | None
None

sort: str | None
None

search: str | None
None

class archivebox.api.v1_cli.**UpdateCommandSchema**
Bases: *archivebox.api.v1_cli.SnapshotFilterCommandSchema*

resume: str | None
None

batch_size: int
100

continuous: bool
False

index_only: bool
False

migrate_only: bool
False

class archivebox.api.v1_cli.**ScheduleCommandSchema**
Bases: ninja.Schema

import_path: str | None
None

add: bool
False

show: bool
False

foreground: bool
False

run_all: bool
False

quiet: bool
False

every: str | None
None

tag: str = <Multiline-String>

depth: int
0

only_new: bool | None
None

update: bool
False

overwrite: bool
False

clear: bool
False

class archivebox.api.v1_cli.**ListCommandSchema**

Bases: *archivebox.api.v1_cli.SnapshotFilterCommandSchema*

as_json: bool
True

as_html: bool
False

as_csv: str | None
'timestamp,url'

with_headers: bool
False

class archivebox.api.v1_cli.**RemoveCommandSchema**

Bases: *archivebox.api.v1_cli.SnapshotFilterCommandSchema*

filter_type: str
None

timeout: float
60.0

archivebox.api.v1_cli.**snapshot_filter_kwargs**(*args:* archivebox.api.v1_cli.SnapshotFilterCommandSchema, *, *default_filter_type: str*) → dict[str, Any]

archivebox.api.v1_cli.**cli_add**(*request: django.http.HttpRequest*, *args:* archivebox.api.v1_cli.AddCommandSchema)

archivebox.api.v1_cli.**cli_update**(*request: django.http.HttpRequest*, *args:* archivebox.api.v1_cli.UpdateCommandSchema)

archivebox.api.v1_cli.**cli_schedule**(*request: django.http.HttpRequest*, *args:* archivebox.api.v1_cli.ScheduleCommandSchema)

archivebox.api.v1_cli.**cli_search**(*request: django.http.HttpRequest*, *args:* archivebox.api.v1_cli.ListCommandSchema)

archivebox.api.v1_cli.**cli_remove**(*request: django.http.HttpRequest*, *args:* archivebox.api.v1_cli.RemoveCommandSchema)

archivebox.api.admin

Module Contents

Classes

APITokenAdmin
OutboundWebhookAdminForm
CustomWebhookAdmin

Functions

_webhook_fields
register_admin

API

archivebox.api.admin.**_webhook_fields**(**names: str*) → tuple[str, ...]

class archivebox.api.admin.**APITokenAdmin**

Bases: *archivebox.base_models.admin.BaseModelAdmin*

list_display

('created_at', 'id', 'created_by', 'token_redacted', 'expires')

sort_fields

('id', 'created_at', 'created_by', 'expires')

readonly_fields

('created_at', 'modified_at')

search_fields

('id', 'created_by__username', 'token')

fieldsets

(('Token',), ('Owner',), ('Timestamps',))

list_filter

('created_by',)

ordering

['-created_at']

list_per_page

100

class archivebox.api.admin.**OutboundWebhookAdminForm**(**args*, ***kwargs*)

Bases: signal_webhooks.admin.WebhookModelForm

Initialization

class archivebox.api.admin.**CustomWebhookAdmin**

Bases: signal_webhooks.admin.WebhookAdmin, *archivebox.base_models.admin.BaseModelAdmin*

form

None

list_display

('created_at', 'created_by', 'id')

sort_fields

'_webhook_fields(...)'

readonly_fields

'_webhook_fields(...)'

fieldsets

(('Webhook',), ('Authentication',), ('Status',), ('Owner',), ('Timestamps',))

lookup_allowed(*lookup: str*, *value: str*, *request: django.http.HttpRequest | None = None*) → bool

Preserve WebhookAdmin's auth token filter with Django's current admin signature.

archivebox.api.admin.**register_admin**(*admin_site: django.contrib.admin.AdminSite*) → None

archivebox.workers

Subpackages

archivebox.workers.management

Subpackages

archivebox.workers.management.commands

Submodules

archivebox.workers.management.commands.runner_watch

Module Contents

Classes

Command

API

class archivebox.workers.management.commands.runner_watch.**Command**

Bases: django.core.management.base.BaseCommand

help

'Watch the debug runserver Process row and restart the background runner on autoreloads.'

add_arguments(*parser*)

handle(**args*, ***kwargs*)

archivebox.workers.management.commands.supervisord_watchdog

Module Contents

Classes

Command

API

class archivebox.workers.management.commands.supervisord_watchdog.**Command**

Bases: `django.core.management.base.BaseCommand`

help

'Stop a foreground-owned supervisord if its exact ArchiveBox owner Process exits.'

add_arguments(*parser*)

handle(**args*, ***kwargs*)

Submodules

archivebox.workers.apps

Module Contents

Classes

WorkersConfig

API

class archivebox.workers.apps.**WorkersConfig**

Bases: `django.apps.AppConfig`

default_auto_field

'django.db.models.BigAutoField'

name

'archivebox.workers'

label

'workers'

archivebox.workers.supervisord_util

Module Contents

Classes

SupervisordConnectionCache	Reuse one XML-RPC proxy until it fails, avoiding hot-loop reconnects.

Functions

_shell_join	
_warn_background_cleanup	
_read_cgroup_limit	
effective_available_memory_bytes	
_require_available_memory	
require_server_worker_memory	
require_dependency_install_memory	
archivebox_cmd	
_archivebox_worker_environment	Keep hook lookup aligned with the pinned ArchiveBox executable.
resolve_env_binary	
_record_supervisord_process	
_fallback_supervisord_process_from_db	
_live_supervisord_processes_from_db	Return live supervisord parents recorded for this DATA_DIR.
_stop_older_supervisord_processes	Stop older supervisord parents for this DATA_DIR after a start race.
RUNNER_WORKER	
RUNNER_ONCE_WORKER	
RUNNER_WATCH_WORKER	
SUPERVISORD_PARENT_WATCHDOG_WORKER	
SERVER_WORKER	
RUNSERVER_WORKER	
is_port_in_use	Check if a port is already in use.
_sonic_worker_bind_target	Read the plugin-owned Sonic config before starting its supervisord worker.
get_sock_file	Get the path to the supervisord socket file.
create_supervisord_config	
_worker_environment_value	
_worker_log_base_dir	
create_worker_config	Create a supervisord worker config file for a given daemon
_current_foreground_supervisord_watchdog_arg	
sync_supervisord_workers	Project desired workers into supervisord from ArchiveBox-owned state.
get_existing_supervisord_process	
stop_existing_supervisord_process	
stop_own_supervisord_process	Stop only the supervisord child started by this Python process.

continues on next page

Table 131 – continued from previous page

reap_foreground_supervisord_process	Reap the supervisord child owned by this foreground parent if it exited.
start_new_supervisord_process	
wait_for_supervisord_ready	Poll for supervisord readiness without a fixed startup sleep.
get_or_create_supervisord_process	
start_worker	
run_runner_worker	
get_worker	
format_runtime_components	
worker_runtime_component	
runtime_components_for_worker_names	
active_supervisord_runtime_components	
build_server_worker_plan	
stop_worker	
tail_multiple_worker_logs	Tail multiple log files simultaneously, interleaving their output.
get_sonic_supervisord_worker_from_plugin	
_proc_cmdline	
_is_sonic_process	
_is_supervisord_process	
_has_live_archivebox_parent	
_terminate_process_tree	
_sonic_listeners	
stop_stale_sonic_processes	
start_server_workers	

Data

LOG_FILE_NAME
CONFIG_FILE_NAME
PID_FILE_NAME
WORKERS_DIR_NAME
_supervisord_proc
_desired_supervisord_workers
_ACTIVE_WORKER_STATES
_RUNTIME_COMPONENT_ORDER
_SUPERVISORD_ERRORS
_PROCESS_STATE_ERRORS
MIN_SERVER_WORKER_AVAILABLE_MEMORY_BYTES
MIN_DEPENDENCY_INSTALL_AVAILABLE_MEMORY_BYTI
_PROC_ACCESS_EXCEPTIONS
_PROC_INFO_EXCEPTIONS

API

archivebox.workers.supervisord_util.**LOG_FILE_NAME**

'supervisord.log'

archivebox.workers.supervisord_util.**CONFIG_FILE_NAME**

'supervisord.conf'

archivebox.workers.supervisord_util.**PID_FILE_NAME**

'supervisord.pid'

archivebox.workers.supervisord_util.**WORKERS_DIR_NAME**

'workers'

archivebox.workers.supervisord_util.**_supervisord_proc**

None

archivebox.workers.supervisord_util.**_desired_supervisord_workers: dict[str, dict[str, str]]**

None

archivebox.workers.supervisord_util.**_ACTIVE_WORKER_STATES**

None

archivebox.workers.supervisord_util.**_RUNTIME_COMPONENT_ORDER**

('orchestrator', 'server', 'sonic')

archivebox.workers.supervisord_util.**_SUPERVISORD_ERRORS**

()

archivebox.workers.supervisord_util.**_PROCESS_STATE_ERRORS**

()

archivebox.workers.supervisord_util.**MIN_SERVER_WORKER_AVAILABLE_MEMORY_BYTES**

None

archivebox.workers.supervisord_util.**MIN_DEPENDENCY_INSTALL_AVAILABLE_MEMORY_BYTES**

None

archivebox.workers.supervisord_util.**_shell_join**(*args: list[str]*) → str

archivebox.workers.supervisord_util.**_warn_background_cleanup**(*context: str*, *err: BaseException*) → None

archivebox.workers.supervisord_util.**_read_cgroup_limit**(*path: pathlib.Path*) → int | None

archivebox.workers.supervisord_util.**effective_available_memory_bytes**() → int

archivebox.workers.supervisord_util.**_require_available_memory**(*operation: str*, *idle_message: str*, *available_bytes: int*, *required_bytes: int*) → None

archivebox.workers.supervisord_util.**require_server_worker_memory**(*available_bytes: int | None = None*) → None

archivebox.workers.supervisord_util.**require_dependency_install_memory**(*available_bytes: int | None = None*) → None

archivebox.workers.supervisord_util.**archivebox_cmd**(**args: str*) → list[str]

archivebox.workers.supervisord_util.**_archivebox_worker_environment**(*command: list[str]*, ***extra: str*) → str

Keep hook lookup aligned with the pinned ArchiveBox executable.

archivebox.workers.supervisord_util.**resolve_env_binary**(*name: str*) → pathlib.Path

archivebox.workers.supervisord_util.**_record_supervisord_process**(*proc: subprocess.Popen*, *config_file: pathlib.Path*, *supervisord_binary: pathlib.Path*) → None

archivebox.workers.supervisord_util.**_fallback_supervisord_process_from_db**()

archivebox.workers.supervisord_util.**_live_supervisord_processes_from_db**()

Return live supervisord parents recorded for this DATA_DIR.

The socket/config files are generated runtime projection and can move when TMP_DIR changes or falls back. Process rows are the durable coordination state, so takeover/shutdown must stop every live supervisord recorded for this collection, not only the one reachable at the current socket path.

archivebox.workers.supervisord_util.**_stop_older_supervisord_processes**(*, *current_pid: int*, *current_started_at: float*, *timeout: float*) → None

Stop older supervisord parents for this DATA_DIR after a start race.

Lazy daemon users such as `archivebox list --search ...` may race to start Sonic. The durable Process table is the arbiter: after this parent is recorded, kill only live supervisord rows that started before this one. If another parent started later, leave it alone so newest healthy owner wins.

archivebox.workers.supervisord_util.**RUNNER_WORKER**()

archivebox.workers.supervisord_util.**RUNNER_ONCE_WORKER**(*args*, *name='worker_runner_once'*)

archivebox.workers.supervisord_util.**RUNNER_WATCH_WORKER**(*bind_url*)

archivebox.workers.supervisord_util.**SUPERVISORD_PARENT_WATCHDOG_WORKER**(*, *owner_pid: int*, *owner_started_at: float*, *supervisord_pid: int*, *supervisord_started_at: float*)

archivebox.workers.supervisord_util.**SERVER_WORKER**(*host: str*, *port: str*) → dict[str, str]

archivebox.workers.supervisord_util.**RUNSERVER_WORKER**(*host: str*, *port: str*, *, *reload: bool*, *nothreading: bool = False*)

archivebox.workers.supervisord_util.**is_port_in_use**(*host: str*, *port: int*) → bool

Check if a port is already in use.

archivebox.workers.supervisord_util.**_sonic_worker_bind_target**(*worker: dict[str, str]*) → tuple[str, int] | None

Read the plugin-owned Sonic config before starting its supervisord worker.

archivebox.workers.supervisord_util.**get_sock_file**()

Get the path to the supervisord socket file.

Supervisord-managed workers inherit SUPERVISOR_SERVER_URL from their parent supervisord. They must keep using that socket so worker code cannot accidentally start a nested supervisord.

archivebox.workers.supervisord_util.**create_supervisord_config**()

archivebox.workers.supervisord_util.**_worker_environment_value**(*daemon: dict[str, str]*, *key: str*) → str | None

archivebox.workers.supervisord_util.**_worker_log_base_dir**(*daemon: dict[str, str]*) → pathlib.Path

archivebox.workers.supervisord_util.**create_worker_config**(*daemon*)

Create a supervisord worker config file for a given daemon

archivebox.workers.supervisord_util.**_current_foreground_supervisord_watchdog_args**()

archivebox.workers.supervisord_util.**sync_supervisord_workers**(*supervisor*, *workers: list[tuple[dict[str, str], bool]]*, ***, *prune: bool = True*)

Project desired workers into supervisord from ArchiveBox-owned state.

The worker conf files are generated supervisor input only. They are never treated as durable ArchiveBox state; callers either pass a complete worker set with prune=True or add one explicit worker with prune=False.

archivebox.workers.supervisord_util.**get_existing_supervisord_process**(***, *quiet: bool = False*)

class archivebox.workers.supervisord_util.**SupervisordConnectionCache**(***, *quiet: bool = False*)

Reuse one XML-RPC proxy until it fails, avoiding hot-loop reconnects.

Initialization

clear() → None

get()

archivebox.workers.supervisord_util.**stop_existing_supervisord_process**()

archivebox.workers.supervisord_util.**stop_own_supervisord_process**(***, *record_exit: bool = True*)

Stop only the supervisord child started by this Python process.

archivebox.workers.supervisord_util.**reap_foreground_supervisord_process**() → None

Reap the supervisord child owned by this foreground parent if it exited.

archivebox.workers.supervisord_util.**start_new_supervisord_process**(*daemonize=False*)

archivebox.workers.supervisord_util.**wait_for_supervisord_ready**(*max_wait_sec: float = 5.0*, *interval_sec: float = 0.1*, ***, *quiet: bool = False*)

Poll for supervisord readiness without a fixed startup sleep.

archivebox.workers.supervisord_util.**get_or_create_supervisord_process**(*daemonize=False*)

archivebox.workers.supervisord_util.**start_worker**(*supervisor*, *daemon*, *lazy=False*)

archivebox.workers.supervisord_util.**run_runner_worker**(*args: list[str]*, ***, *name: str = 'worker_runner_once'*, *interactive_interrupts: bool = False*, *keep_running=None*, *config=None*) → int

archivebox.workers.supervisord_util.**get_worker**(*supervisor*, *daemon_name*)

archivebox.workers.supervisord_util.**format_runtime_components**(*components: list[str] | tuple[str, ...]*) → str

archivebox.workers.supervisord_util.**worker_runtime_component**(*worker_name: str*) → str | None

archivebox.workers.supervisord_util.**runtime_components_for_worker_names**(*worker_names: set[str] | list[str] | tuple[str, ...]*) → list[str]

archivebox.workers.supervisord_util.**active_supervisord_runtime_components**(*, *supervisor=None*) → list[str]

archivebox.workers.supervisord_util.**build_server_worker_plan**(*, *config*, *host: str*, *port: str*, *debug: bool*, *reload: bool*, *nothreading: bool*, *supervisor=None*)

archivebox.workers.supervisord_util.**stop_worker**(*supervisor*, *daemon_name*)

archivebox.workers.supervisord_util.**tail_multiple_worker_logs**(*log_files: list[str]*, *follow=True*, *proc=None*, *keep_running=None*)

Tail multiple log files simultaneously, interleaving their output.

Args: log_files: List of log file paths to tail follow: Whether to keep following (True) or just read existing content (False) proc: Optional subprocess.Popen object - stop tailing when this process exits

archivebox.workers.supervisord_util.**get_sonic_supervisord_worker_from_plugin**(*config*) → dict[str, str] | None

archivebox.workers.supervisord_util.**_PROC_ACCESS_EXCEPTIONS**

()

archivebox.workers.supervisord_util.**_PROC_INFO_EXCEPTIONS**

()

archivebox.workers.supervisord_util.**_proc_cmdline**(*proc: psutil.Process*) → list[str]

archivebox.workers.supervisord_util.**_is_sonic_process**(*proc: psutil.Process*) → bool

archivebox.workers.supervisord_util.**_is_supervisord_process**(*proc: psutil.Process | None*) → bool

archivebox.workers.supervisord_util.**_has_live_archivebox_parent**(*proc: psutil.Process | None*) → bool

archivebox.workers.supervisord_util.**_terminate_process_tree**(*root: psutil.Process*, *, *timeout: float = 2.0*) → None

archivebox.workers.supervisord_util.**_sonic_listeners**(*host: str*, *port: int*) → list[psutil.Process]

archivebox.workers.supervisord_util.**stop_stale_sonic_processes**(*sonic_worker: dict[str, str]*, *, *supervisor_pid: int | None*, *host: str | None = None*, *port: int | None = None*) → None

archivebox.workers.supervisord_util.**start_server_workers**(*host='0.0.0.0'*, *port='5797'*, *daemonize=False*, *debug=False*, *reload=False*, *nothreading=False*, *keep_running=None*, *should_stop_supervisord=None*, *resumed_from_pid=None*)

archivebox.workers.supervisord_parent_watchdog

Module Contents

Functions

matching_process	
run_watchdog	
main	

API

archivebox.workers.supervisord_parent_watchdog.**matching_process**(*pid: int*, *started_at: float*) → psutil.Process | None

archivebox.workers.supervisord_parent_watchdog.**run_watchdog**(*, *owner_pid: int*, *owner_started_at: float*, *supervisord_pid: int*, *supervisord_started_at: float*, *interval: float = 1.0*) → None

archivebox.workers.supervisord_parent_watchdog.**main**() → None

archivebox.workers.models

Module Contents

Classes

DefaultStatusChoices	
ModelWithQueue	Durable queue fields and atomic lease operations shared by work rows.

Data

default_status_field	
default_retry_at_field	
RETRY_AT_MAX	
ACTIVE_STATE_LEASE_SECONDS	
logger	
MODULE_PATH	
REPO_ROOT	
PACKAGE_ROOT	

API

class archivebox.workers.models.**DefaultStatusChoices**

Bases: django.db.models.TextChoices

QUEUED

('queued', 'Queued')

STARTED

('started', 'Started')

PAUSED

('paused', 'Paused')

SEALED

('sealed', 'Sealed')

archivebox.workers.models.**default_status_field: django.db.models.CharField**

'CharField(...)'

archivebox.workers.models.**default_retry_at_field: django.db.models.DateTimeField**

'DateTimeField(...)'

archivebox.workers.models.**RETRY_AT_MAX**

'datetime(...)'

archivebox.workers.models.**ACTIVE_STATE_LEASE_SECONDS**

60

archivebox.workers.models.**logger**

'getLogger(...)'

archivebox.workers.models.**MODULE_PATH**

'resolve(...)'

archivebox.workers.models.**REPO_ROOT**

None

archivebox.workers.models.**PACKAGE_ROOT**

None

class archivebox.workers.models.**ModelWithQueue**

Bases: django.db.models.Model

Durable queue fields and atomic lease operations shared by work rows.

Concrete models own lifecycle transitions. This mixin only owns the common database queue protocol: status, retry_at, pause/resume, and claims.

StatusChoices: ClassVar[*type*[django.db.models.TextChoices]]

None

INITIAL_STATE: ClassVar[str]

None

ACTIVE_STATE: ClassVar[str]

None

FINAL_STATES: ClassVar[tuple[str, ...]]

()

warn_on_save_outside_runner: ClassVar[bool]

True

status: django.db.models.CharField

'CharField(...)'

retry_at: django.db.models.DateTimeField

'DateTimeField(...)'

class Meta

Bases: `django_stubs_ext.db.models.TypedModelMeta`

app_label

'workers'

abstract

True

FINAL_OR_ACTIVE_STATES: ClassVar[tuple[str, ...]]

()

classmethod status_counts(*queryset: django.db.models.QuerySet | None = None*, *statuses: collections.abc.Iterable[str] | None = None*) → dict[str, int]

property RETRY_AT: datetime.datetime | None

property STATE: str

bump_retry_at(*seconds: int = 10*) → None

property is_paused: bool

safe_update(*update_fields: dict[str, Any]*, ***, *refresh: bool = True*, *extra_filter: dict[str, Any] | None = None*) → bool

save(**args: Any*, ***kwargs: Any*) → None

pause(***, *save: bool = True*) → bool

resume(***, *when: datetime.datetime | None = None*, *save: bool = True*) → bool

update_and_requeue(***, *refresh: bool = True*, ***kwargs: Any*) → bool

classmethod get_queue()

classmethod claim_for_worker(*obj:* archivebox.workers.models.ModelWithQueue, *lock_seconds: int = 60*) → bool

claim_processing_lock(*lock_seconds: int = 60*) → bool

classmethod extend_choices(*base_choices:* type*[django.db.models.TextChoices]*)

classmethod StatusField(***kwargs: Any*) → django.db.models.CharField

classmethod RetryAtField(***kwargs: Any*) → django.db.models.DateTimeField

archivebox.workers.admin

Workers admin module.

Background runner processes do not need Django admin registration.

Module Contents

Functions

register_admin	No models to register - workers are process-based, not Django models.

API

archivebox.workers.admin.**register_admin**(*admin_site*)

No models to register - workers are process-based, not Django models.

Package Contents

Functions

register_admin

Data

__order__

API

archivebox.workers.**__order__**

100

archivebox.workers.**register_admin**(*admin_site*)

archivebox.core

Subpackages

archivebox.core.templatetags

Submodules

archivebox.core.templatetags.core_tags

Module Contents

Functions

snapshot_url_text	Emphasize the hostname while preserving the exact selectable URL.
plugin_ui	Optional plugin UI must never prevent the host page from rendering.

continues on next page

Table 139 – continued from previous page

_is_web_subdomain_alias	Hide the informational mismatch banner on the conventional public web alias.
_normalize_output_files	
_snapshot_id	
_coerce_output_file_size	
_count_media_files	
_list_media_files	
_resolve_snapshot_output_file	
_is_text_preview_path	
_is_image_preview_path	
_is_root_snapshot_output_path	
_static_snapshot_base_url	
_snapshot_base_url_for_context	
_snapshot_url_for_context	
_build_snapshot_preview_url	
_render_text_preview	
_render_fallback_card	
_render_text_file_preview	
split	
index	
file_size	
grid_file_size	
intcomma	
result_list	Monkey patched result
_machine_health_stats	Cached wrapper around the Machine-admin stats util.
system_warnings_banner	Render the top-of-page warning banner for one of the conditions below, in priority order (highest first):
url_replace	
admin_base_url	
web_base_url	
static_export_root_url	
snapshot_base_url	
snapshot_detail_url	Keep the trusted snapshot detail page on web.*, not the replay host.
snapshot_url	
snapshot_files_url	
snapshot_zip_url	
snapshot_role_output_url	
snapshot_result_file_url	Build a saved-file URL from one ArchiveResult declaration.
snapshot_thumbnail	
snapshot_index_row	
snapshot_preview_url	
plugin_icon	Render the icon for a plugin.
plugin_card	Render the card template for an archive result.
render_plugin_card_document	Render one plugin card template for the trusted snapshot-origin frame.
output_card	
plugin_full	Render the full template for an archive result.
plugin_name	Get the base name of a plugin (strips numeric prefix).

continues on next page

Table 139 – continued from previous page

api_token	Return an API token string for the logged-in user, creating one if needed.

Data

register
CARD_OVERFLOW_STYLE
_TEXT_PREVIEW_EXTS
_IMAGE_PREVIEW_EXTS
_STATIC_URL_SAFE
_VIDEO_FILE_EXTS
_AUDIO_FILE_EXTS
_MEDIA_FILE_EXTS
_BROWSER_VIDEO_FILE_EXTS
_BROWSER_AUDIO_FILE_EXTS
_LOW_DISK_THRESHOLD_GB
_HIGH_MEMORY_THRESHOLD_PCT
_HIGH_LOAD_MULTIPLE
_HEALTH_CHECK_INTERVAL_SECONDS
_health_cache

API

archivebox.core.templatetags.core_tags.**register**

'Library(…)'

archivebox.core.templatetags.core_tags.**CARD_OVERFLOW_STYLE**

'<style>html,body{height:100%!important;min-height:0!important}html,body,body *{overflow:clip!importa…'

archivebox.core.templatetags.core_tags.**snapshot_url_text**(*url*)

Emphasize the hostname while preserving the exact selectable URL.

archivebox.core.templatetags.core_tags.**plugin_ui**(*context*, *plugin: str*, *slot: str*)

Optional plugin UI must never prevent the host page from rendering.

archivebox.core.templatetags.core_tags.**_TEXT_PREVIEW_EXTS**

('.json', '.jsonl', '.txt', '.csv', '.tsv', '.xml', '.yml', '.yaml', '.md', '.log')

archivebox.core.templatetags.core_tags.**_IMAGE_PREVIEW_EXTS**

('.png', '.jpg', '.jpeg', '.gif', '.webp', '.bmp', '.ico', '.avif')

archivebox.core.templatetags.core_tags.**_STATIC_URL_SAFE**

"/@-._~!$&'()*+,;="

archivebox.core.templatetags.core_tags.**_VIDEO_FILE_EXTS**

None

archivebox.core.templatetags.core_tags.**_AUDIO_FILE_EXTS**

None

archivebox.core.templatetags.core_tags.**_MEDIA_FILE_EXTS**

None

archivebox.core.templatetags.core_tags.**_BROWSER_VIDEO_FILE_EXTS**

None

archivebox.core.templatetags.core_tags.**_BROWSER_AUDIO_FILE_EXTS**

None

archivebox.core.templatetags.core_tags.**_is_web_subdomain_alias**(*browser_url: str*, *configured_base_url: str*) → bool

Hide the informational mismatch banner on the conventional public web alias.

This is deliberately presentation-only: request routing and the restriction on state-changing requests continue to use the configured BASE_URL.

archivebox.core.templatetags.core_tags.**_normalize_output_files**(*output_files: Any*) → dict[str, dict[str, Any]]

archivebox.core.templatetags.core_tags.**_snapshot_id**(*value: Any*) → Any

archivebox.core.templatetags.core_tags.**_coerce_output_file_size**(*value: Any*) → int | None

archivebox.core.templatetags.core_tags.**_count_media_files**(*result*) → int

archivebox.core.templatetags.core_tags.**_list_media_files**(*result*, *, *include_filesystem_fallback=True*) → list[dict]

archivebox.core.templatetags.core_tags.**_resolve_snapshot_output_file**(*snapshot_dir: str | pathlib.Path | None*, *raw_output_path: str | None*) → pathlib.Path | None

archivebox.core.templatetags.core_tags.**_is_text_preview_path**(*raw_output_path: str | None*) → bool

archivebox.core.templatetags.core_tags.**_is_image_preview_path**(*raw_output_path: str | None*) → bool

archivebox.core.templatetags.core_tags.**_is_root_snapshot_output_path**(*raw_output_path: str | None*) → bool

archivebox.core.templatetags.core_tags.**_static_snapshot_base_url**(*context*, *snapshot*) → str

archivebox.core.templatetags.core_tags.**_snapshot_base_url_for_context**(*context*, *snapshot*) → str

archivebox.core.templatetags.core_tags.**_snapshot_url_for_context**(*context*, *snapshot*, *path: str = ''*) → str

archivebox.core.templatetags.core_tags.**_build_snapshot_preview_url**(*snapshot_id: str*, *path: str = ''*, *request=None*, *config=None*, *plugin: str = ''*, *base_url: str | None = None*) → str

archivebox.core.templatetags.core_tags.**_render_text_preview**(*plugin: str*, *icon_html: str*, *snippet: str*) → str

archivebox.core.templatetags.core_tags.**_render_fallback_card**(*plugin: str*, *icon_html: str*, *fallback_label: str*) → str

archivebox.core.templatetags.core_tags.**_render_text_file_preview**(*snapshot_dir: str | pathlib.Path | None*, *raw_output_path: str | None*, *plugin: str*, *icon_html: str*) → str | None

archivebox.core.templatetags.core_tags.**split**(*value*, *separator: str = ','*)

archivebox.core.templatetags.core_tags.**index**(*value*, *position*)

archivebox.core.templatetags.core_tags.**file_size**(*num_bytes: float*, *decimal_places: int = 1*) → str

archivebox.core.templatetags.core_tags.**grid_file_size**(*num_bytes: float | None*) → str

archivebox.core.templatetags.core_tags.**intcomma**(*value: int | str | None*) → str

archivebox.core.templatetags.core_tags.**result_list**(*context*, *cl*)

Monkey patched result

archivebox.core.templatetags.core_tags.**_LOW_DISK_THRESHOLD_GB**

1.0

archivebox.core.templatetags.core_tags.**_HIGH_MEMORY_THRESHOLD_PCT**

95.0

archivebox.core.templatetags.core_tags.**_HIGH_LOAD_MULTIPLE**

3

archivebox.core.templatetags.core_tags.**_HEALTH_CHECK_INTERVAL_SECONDS**

30

archivebox.core.templatetags.core_tags.**_health_cache: dict**

None

archivebox.core.templatetags.core_tags.**_machine_health_stats**() → dict

Cached wrapper around the Machine-admin stats util.

The Machine list/change pages already render disk / mem / load via `archivebox.machine.detect.get_host_stats()` — we reuse the same call so the banner thresholds line up 1:1 with what's shown on /admin/machine/. Cached for 30s because the inclusion tag fires on every page render and `get_host_stats` shells into several psutil probes.

archivebox.core.templatetags.core_tags.**system_warnings_banner**(*context*)

Render the top-of-page warning banner for one of the conditions below, in priority order (highest first):

1. `mode="unconfigured"`— `BASE_URL` is empty and the operator has not selected safe single-domain routing. That mode intentionally uses each device's connection address, including LAN and Tailscale IPs.
2. `mode="base_url_mismatch"` — the browser reached ArchiveBox through an origin that differs from the configured canonical `BASE_URL`.
3. `mode="unsafe"` — `SERVER_SECURITY_MODE` explicitly enables a lower-security replay mode.
4. `mode="low_disk"` — `DATA_DIR` has <1 GiB free; new archive jobs will start failing on ENOSPC.
5. `mode="high_memory"` — virtual memory utilization at/above 95%; the host is one OOM-kill from a crash.
6. `mode="high_load"` — 15-minute load average exceeds 3 × CPU count (the kernel's own sustained-load EMA, so no rolling buffer of ours is needed).

Config/security warnings come first because they affect correctness + security and need explicit operator action; host-health warnings come after and reuse `machine.detect.get_host_stats` (the same function that populates the Machine admin page), cached for 30s.

archivebox.core.templatetags.core_tags.**url_replace**(*context*, ***kwargs*)

archivebox.core.templatetags.core_tags.**admin_base_url**(*context*) → str

archivebox.core.templatetags.core_tags.**web_base_url**(*context*) → str

archivebox.core.templatetags.core_tags.**static_export_root_url**(*context*) → str

archivebox.core.templatetags.core_tags.**snapshot_base_url**(*context*, *snapshot*) → str

archivebox.core.templatetags.core_tags.**snapshot_detail_url**(*context*, *snapshot*) → str

Keep the trusted snapshot detail page on web.*, not the replay host.

archivebox.core.templatetags.core_tags.**snapshot_url**(*context*, *snapshot*, *path: str = ''*) → str

archivebox.core.templatetags.core_tags.**snapshot_files_url**(*context*, *snapshot*, *path: str = ''*) → str

archivebox.core.templatetags.core_tags.**snapshot_zip_url**(*context*, *snapshot*, *path: str = ''*) → str

archivebox.core.templatetags.core_tags.**snapshot_role_output_url**(*context*, *snapshot*, *role: str*, *fallback_to_default: bool = False*) → str

archivebox.core.templatetags.core_tags.**snapshot_result_file_url**(*context*, *snapshot*, *result*, *path: str*) → str

Build a saved-file URL from one ArchiveResult declaration.

archivebox.core.templatetags.core_tags.**snapshot_thumbnail**(*context*, *snapshot*, *admin=False*, *width='100px'*, *height='100px'*)

archivebox.core.templatetags.core_tags.**snapshot_index_row**(*context*, *link*) → str

archivebox.core.templatetags.core_tags.**snapshot_preview_url**(*context*, *snapshot*, *path: str = ''*, *result=None*) → str

archivebox.core.templatetags.core_tags.**plugin_icon**(*plugin: str*) → str

Render the icon for a plugin.

Usage: {% plugin_icon “screenshot” %}

archivebox.core.templatetags.core_tags.**plugin_card**(*context*, *result*) → str

Render the card template for an archive result.

Usage: {% plugin_card result %}

Context variables passed to template: - result: ArchiveResult object - snapshot: Parent Snapshot object - output_path: Path to output relative to snapshot dir (from embed_path()) - plugin: Plugin base name

archivebox.core.templatetags.core_tags.**render_plugin_card_document**(*context*, *result*) → str

Render one plugin card template for the trusted snapshot-origin frame.

archivebox.core.templatetags.core_tags.**output_card**(*snapshot*, *output_path: str*, *plugin: str*) → str

archivebox.core.templatetags.core_tags.**plugin_full**(*context*, *result*) → str

Render the full template for an archive result.

Usage: {% plugin_full result %}

archivebox.core.templatetags.core_tags.**plugin_name**(*value: str*) → str

Get the base name of a plugin (strips numeric prefix).

Usage: {{ result.plugin|plugin_name }}

archivebox.core.templatetags.core_tags.**api_token**(*context*) → str

Return an API token string for the logged-in user, creating one if needed.

archivebox.core.management

Subpackages

archivebox.core.management.commands

Submodules

archivebox.core.management.commands.archivebox

Module Contents

Classes

Command

API

class archivebox.core.management.commands.archivebox.**Command**

Bases: `django.core.management.base.BaseCommand`

help

'Run an ArchiveBox CLI subcommand (e.g. add, remove, list, etc)'

add_arguments(*parser*)

handle(**args*, ***kwargs*)

archivebox.core.sqlite_backend

Submodules

archivebox.core.sqlite_backend.base

Module Contents

Classes

SQLiteCursorWrapper
DatabaseWrapper

Functions

_sqlite_lock_retry_timeout
_sqlite_lock_retry_interval
_format_sql
_log_locked_database
_connection_in_transaction
_recover_non_atomic_connection
_is_inside_atomic
_abort_locked_database
_retry_locked_database

API

archivebox.core.sqlite_backend.base.**_sqlite_lock_retry_timeout**() → float

archivebox.core.sqlite_backend.base.**_sqlite_lock_retry_interval**() → float

archivebox.core.sqlite_backend.base.**_format_sql**(*query: str*, *params=None*) → str

archivebox.core.sqlite_backend.base.**_log_locked_database**(*query: str*, *params=None*, *, *attempt: int*, *elapsed: float*, *retry_interval: float*) → None

archivebox.core.sqlite_backend.base.**_connection_in_transaction**(*connection*) → bool

archivebox.core.sqlite_backend.base.**_recover_non_atomic_connection**(*db_wrapper*, *query: str*) → None

archivebox.core.sqlite_backend.base.**_is_inside_atomic**(*db_wrapper*) → bool

archivebox.core.sqlite_backend.base.**_abort_locked_database**(*query: str*, *params=None*, *, *elapsed: float*, *db_wrapper=None*) → None

archivebox.core.sqlite_backend.base.**_retry_locked_database**(*action*, *query: str*, *params=None*, *, *db_wrapper=None*)

class archivebox.core.sqlite_backend.base.**SQLiteCursorWrapper**

Bases: django.db.backends.sqlite3.base.SQLiteCursorWrapper

execute(*query*, *params=None*)

executemany(*query*, *param_list*)

class archivebox.core.sqlite_backend.base.**DatabaseWrapper**

Bases: django.db.backends.sqlite3.base.DatabaseWrapper

create_cursor(*name=None*)

_commit()

_rollback()

Submodules

archivebox.core.admin_archiveresults

Module Contents

Classes

ArchiveResultInline	
ArchiveResultAdmin	

Functions

_get_replay_source_url	
build_abx_dl_display_command	
build_abx_dl_replay_command	
get_plugin_admin_url	
get_process_link_label	
render_archiveresults_list	Render a nice inline list view of archive results with status, plugin, output, and actions.
register_admin	

API

archivebox.core.admin_archiveresults.**_get_replay_source_url**(*result:* archivebox.core.models.ArchiveResult) → str

archivebox.core.admin_archiveresults.**build_abx_dl_display_command**(*result:* archivebox.core.models.ArchiveResult) → str

archivebox.core.admin_archiveresults.**build_abx_dl_replay_command**(*result:* archivebox.core.models.ArchiveResult, *config=None*) → str

archivebox.core.admin_archiveresults.**get_plugin_admin_url**(*plugin_name: str*) → str

archivebox.core.admin_archiveresults.**get_process_link_label**(*process*) → str

archivebox.core.admin_archiveresults.**render_archiveresults_list**(*archiveresults_qs*, *limit=50*, *config=None*, *can_delete=False*)

Render a nice inline list view of archive results with status, plugin, output, and actions.

class archivebox.core.admin_archiveresults.**ArchiveResultInline**

Bases: `django.contrib.admin.TabularInline`

name

'Archive Results'

model

None

parent_model

None

extra

0

sort_fields

('end_ts', 'plugin', 'output_str', 'status', 'cmd_version')

readonly_fields

('id', 'result_id', 'completed', 'command', 'version')

fields

('start_ts', 'end_ts')

ordering

('end_ts',)

show_change_link

True

get_parent_object_from_request(*request*)

completed(*obj*)

result_id(*obj*)

command(*obj*)

version(*obj*)

get_formset(*request*, *obj=None*, ***kwargs*)

get_readonly_fields(*request*, *obj=None*)

class archivebox.core.admin_archiveresults.**ArchiveResultAdmin**

Bases: *archivebox.base_models.admin.BaseModelAdmin*

list_select_related

()

list_display

('details_link', 'zip_link', 'created_at', 'snapshot_info', 'tags_inline', 'status_badge', 'plugin_w...

list_display_links

None

sort_fields

('id', 'created_at', 'plugin', 'status')

readonly_fields

('cmd', 'cmd_version', 'pwd', 'cmd_str', 'admin_actions', 'snapshot_info', 'tags_str', 'created_at',...

search_fields

('snapshot__id', 'snapshot__url', 'snapshot__tags__name', 'snapshot__crawl_id', 'plugin', 'hook_name...

autocomplete_fields

('snapshot',)

fieldsets

((‘Snapshot’,), (‘Plugin’,), (‘Timing’,), (‘Command’,), (‘Output’,))

list_filter

(‘status’, ‘plugin’, ‘start_ts’)

ordering

(‘-start_ts’,)

list_per_page

50

paginator

None

save_on_top

True

show_full_result_count

False

actions

(‘delete_selected’,)

class Meta

verbose_name

‘Archive Result’

verbose_name_plural

‘Archive Results’

change_view(*request*, *object_id*, *form_url=''*, *extra_context=None*)

get_admin_toolbar_actions(*request*, *obj*)

changelist_view(*request*, *extra_context=None*)

get_queryset(*request*)

get_search_results(*request*, *queryset*, *search_term*)

get_snapshot_view_url(*result:* archivebox.core.models.ArchiveResult) → str

get_output_view_url(*result:* archivebox.core.models.ArchiveResult) → str

get_output_files_url(*result:* archivebox.core.models.ArchiveResult) → str

get_output_zip_url(*result:* archivebox.core.models.ArchiveResult) → str

details_link(*result*)

zip_link(*result*)

admin_actions(*result*)

snapshot_info(*result*)

tags_str(*result*)

tags_inline(*result*)

status_badge(*result*)

plugin_with_icon(*result*)

process_link(*result*)

machine_link(*result*)

cmd_str(*result*)

output_display(*result*)

output_str_display(*result*)

output_summary(*result*)

archivebox.core.admin_archiveresults.**register_admin**(*admin_site*)

archivebox.core.widgets

Module Contents

Classes

TagEditorWidget	A widget that renders tags as clickable pills with inline editing.
URLFiltersWidget	Render URL allowlist / denylist controls with same-domain autofill.
InlineTagEditorWidget	Inline version of TagEditorWidget for use in list views. Includes AJAX save functionality for immediate persistence.

API

class archivebox.core.widgets.**TagEditorWidget**(*attrs=None*, *snapshot_id=None*)

Bases: `django.forms.Widget`

A widget that renders tags as clickable pills with inline editing.

- Displays existing tags alphabetically as styled pills with X remove button
- Text input with HTML5 datalist for autocomplete suggestions
- Press Enter or Space to create new tags (auto-creates if doesn't exist)
- Uses AJAX for autocomplete and tag creation

Initialization

template_name = <Multiline-String>

class Media

css

None

js

[]

_escape(*value*)

Escape HTML entities in value.

_normalize_id(*value*)

Normalize IDs for HTML + JS usage (letters, digits, underscore; JS-safe start).

_json_for_inline_script(*value*)

Serialize JSON so it cannot close the surrounding inline <script> tag.

_tag_style(*value*)

Compute a stable pastel color style for a tag value.

render(*name*, *value*, *attrs=None*, *renderer=None*)

Render the tag editor widget.

Args: name: Field name value: Can be: - QuerySet of Tag objects (from M2M field) - List of tag names - Comma-separated string of tag names - None attrs: HTML attributes renderer: Not used

class archivebox.core.widgets.**URLFiltersWidget**(*attrs=None*, ***, *source_selector='textarea[name="url"]'*)

Bases: `django.forms.Widget`

Render URL allowlist / denylist controls with same-domain autofill.

Initialization

template_name = <Multiline-String>

render(*name*, *value*, *attrs=None*, *renderer=None*)

value_from_datadict(*data*, *files*, *name*)

class archivebox.core.widgets.**InlineTagEditorWidget**(*attrs=None*, *snapshot_id=None*, *editable=True*)

Bases: *archivebox.core.widgets.TagEditorWidget*

Inline version of TagEditorWidget for use in list views. Includes AJAX save functionality for immediate persistence.

Initialization

render(*name*, *value*, *attrs=None*, *renderer=None*, *snapshot_id=None*)

Render inline tag editor with AJAX save.

archivebox.core.settings

Module Contents

Data

IS_MIGRATING
IS_TESTING
IS_SHELL
IS_GETTING_VERSION_OR_HELP

continues on next page

Table 147 – continued from previous page

PACKAGE_DIR
logger
ALL_PLUGINS
LOADED_PLUGINS
WSGI_APPLICATION
ASGI_APPLICATION
ROOT_URLCONF
LOGIN_URL
LOGOUT_REDIRECT_URL
PASSWORD_RESET_URL
APPEND_SLASH
DEBUG
INSTALLED_APPS
DJANGO_OBJECT_ACTIONS_DEFAULT_HTTP_METHOD
MIDDLEWARE
AUTHENTICATION_BACKENDS
STATIC_URL
TEMPLATES_DIR_NAME
CUSTOM_TEMPLATES_ENABLED
STATICFILES_DIRS
TEMPLATE_DIRS
TEMPLATES
DATABASE_NAME
SQLITE_JOURNAL_MODE
SQLITE_MMAP_SIZE
SQLITE_CONNECTION_OPTIONS
MIGRATION_MODULES
DEFAULT_AUTO_FIELD
DATABASE_ROUTERS
CACHES
EMAIL_BACKEND
STORAGES
CHANNEL_LAYERS
SECRET_KEY
ALLOWED_HOSTS
CSRF_TRUSTED_ORIGINS
admin_base_url
api_base_url
SECURE_BROWSER_XSS_FILTER
SECURE_CONTENT_TYPE_NOSNIFF
SECURE_REFERRER_POLICY
BASE_URL_IS_HTTPS
CSRF_COOKIE_SECURE
SESSION_COOKIE_SECURE
SESSION_COOKIE_HTTPONLY
SESSION_COOKIE_NAME
CSRF_COOKIE_NAME
SESSION_COOKIE_DOMAIN
CSRF_COOKIE_DOMAIN
SESSION_COOKIE_AGE
SESSION_EXPIRE_AT_BROWSER_CLOSE
SESSION_SAVE_EVERY_REQUEST

continues on next page

Table 147 – continued from previous page

SESSION_ENGINE
AUTH_PASSWORD_VALIDATORS
DATA_UPLOAD_MAX_NUMBER_FIELDS
DATA_UPLOAD_MAX_MEMORY_SIZE
SHELL_PLUS
SHELL_PLUS_PRINT_SQL
IPYTHON_ARGUMENTS
IPYTHON_KERNEL_DISPLAY_NAME
LANGUAGE_CODE
USE_I18N
USE_TZ
DATETIME_FORMAT
SHORT_DATETIME_FORMAT
TIME_ZONE
LOGGING
SIGNAL_WEBHOOKS_CUSTOM_MODEL
SIGNAL_WEBHOOKS
ADMIN_DATA_VIEWS
DEBUG_TOOLBAR
DEBUG_REQUESTS_TRACKER

API

archivebox.core.settings.**IS_MIGRATING**

None

archivebox.core.settings.**IS_TESTING**

None

archivebox.core.settings.**IS_SHELL**

None

archivebox.core.settings.**IS_GETTING_VERSION_OR_HELP**

None

archivebox.core.settings.**PACKAGE_DIR**

None

archivebox.core.settings.**logger**

'getLogger(...)'

archivebox.core.settings.**ALL_PLUGINS**

None

archivebox.core.settings.**LOADED_PLUGINS**

None

archivebox.core.settings.**WSGI_APPLICATION**

'archivebox.core.wsgi.application'

archivebox.core.settings.**ASGI_APPLICATION**

'archivebox.core.asgi.application'

archivebox.core.settings.**ROOT_URLCONF**

'archivebox.core.urls'

archivebox.core.settings.**LOGIN_URL**
'/accounts/login/'

archivebox.core.settings.**LOGOUT_REDIRECT_URL**
None

archivebox.core.settings.**PASSWORD_RESET_URL**
'/accounts/password_reset/'

archivebox.core.settings.**APPEND_SLASH**
True

archivebox.core.settings.**DEBUG**
None

archivebox.core.settings.**INSTALLED_APPS**
['daphne', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.…

archivebox.core.settings.**DJANGO_OBJECT_ACTIONS_DEFAULT_HTTP_METHOD**
'POST'

archivebox.core.settings.**MIDDLEWARE**
['archivebox.core.middleware.TimezoneMiddleware', 'django.middleware.security.SecurityMiddleware', '…

archivebox.core.settings.**AUTHENTICATION_BACKENDS**
['django.contrib.auth.backends.RemoteUserBackend', 'django.contrib.auth.backends.ModelBackend']

archivebox.core.settings.**STATIC_URL**
'/static/'

archivebox.core.settings.**TEMPLATES_DIR_NAME**
'templates'

archivebox.core.settings.**CUSTOM_TEMPLATES_ENABLED**
None

archivebox.core.settings.**STATICFILES_DIRS**
None

archivebox.core.settings.**TEMPLATE_DIRS**
None

archivebox.core.settings.**TEMPLATES**
None

archivebox.core.settings.**DATABASE_NAME**
None

archivebox.core.settings.**SQLITE_JOURNAL_MODE**
None

archivebox.core.settings.**SQLITE_MMAP_SIZE**
None

archivebox.core.settings.**SQLITE_CONNECTION_OPTIONS**
None

archivebox.core.settings.**MIGRATION_MODULES**
None

archivebox.core.settings.**DEFAULT_AUTO_FIELD**
'django.db.models.BigAutoField'

archivebox.core.settings.**DATABASE_ROUTERS**
[]

archivebox.core.settings.**CACHES**
None

archivebox.core.settings.**EMAIL_BACKEND**
'django.core.mail.backends.console.EmailBackend'

archivebox.core.settings.**STORAGES**
None

archivebox.core.settings.**CHANNEL_LAYERS**
None

archivebox.core.settings.**SECRET_KEY**
None

archivebox.core.settings.**ALLOWED_HOSTS**
None

archivebox.core.settings.**CSRF_TRUSTED_ORIGINS**
'list(...)'

archivebox.core.settings.**admin_base_url**
'normalize_base_url(...)'

archivebox.core.settings.**api_base_url**
'normalize_base_url(...)'

archivebox.core.settings.**SECURE_BROWSER_XSS_FILTER**
True

archivebox.core.settings.**SECURE_CONTENT_TYPE_NOSNIFF**
True

archivebox.core.settings.**SECURE_REFERRER_POLICY**
'strict-origin-when-cross-origin'

archivebox.core.settings.**BASE_URL_IS_HTTPS**
'startswith(...)'

archivebox.core.settings.**CSRF_COOKIE_SECURE**
None

archivebox.core.settings.**SESSION_COOKIE_SECURE**
None

archivebox.core.settings.**SESSION_COOKIE_HTTPONLY**
True

archivebox.core.settings.**SESSION_COOKIE_NAME**
None

archivebox.core.settings.**CSRF_COOKIE_NAME**
None

archivebox.core.settings.**SESSION_COOKIE_DOMAIN**
None

archivebox.core.settings.**CSRF_COOKIE_DOMAIN**
None

archivebox.core.settings.**SESSION_COOKIE_AGE**
1209600

archivebox.core.settings.**SESSION_EXPIRE_AT_BROWSER_CLOSE**
False

archivebox.core.settings.**SESSION_SAVE_EVERY_REQUEST**
False

archivebox.core.settings.**SESSION_ENGINE**
'django.contrib.sessions.backends.db'

archivebox.core.settings.**AUTH_PASSWORD_VALIDATORS**
None

archivebox.core.settings.**DATA_UPLOAD_MAX_NUMBER_FIELDS**
None

archivebox.core.settings.**DATA_UPLOAD_MAX_MEMORY_SIZE**
26214400

archivebox.core.settings.**SHELL_PLUS**
'ipython'

archivebox.core.settings.**SHELL_PLUS_PRINT_SQL**
False

archivebox.core.settings.**IPYTHON_ARGUMENTS**
['–no-confirm-exit', '–no-banner']

archivebox.core.settings.**IPYTHON_KERNEL_DISPLAY_NAME**
'ArchiveBox Django Shell'

archivebox.core.settings.**LANGUAGE_CODE**
'en-us'

archivebox.core.settings.**USE_I18N**
True

archivebox.core.settings.**USE_TZ**
True

archivebox.core.settings.**DATETIME_FORMAT**
'Y-m-d h:i:s A'

archivebox.core.settings.**SHORT_DATETIME_FORMAT**
'Y-m-d h:i:s A'

archivebox.core.settings.**TIME_ZONE**
None

archivebox.core.settings.**LOGGING**
None

archivebox.core.settings.**SIGNAL_WEBHOOKS_CUSTOM_MODEL**

'archivebox.api.models.OutboundWebhook'

archivebox.core.settings.**SIGNAL_WEBHOOKS: dict[str, object]**

None

archivebox.core.settings.**ADMIN_DATA_VIEWS**

None

archivebox.core.settings.**DEBUG_TOOLBAR**

False

archivebox.core.settings.**DEBUG_REQUESTS_TRACKER**

None

archivebox.core.forms

Module Contents

Classes

AddLinkForm	
TagWidget	
TagField	

Functions

_split_strip	
parse_tag_string	Parse the legacy tag-editing format used by the admin tag field.
edit_string_for_tag_names	

Data

DEPTH_CHOICES	

API

archivebox.core.forms.**DEPTH_CHOICES**

(('0', 'depth = 0 (archive just these URLs)'), ('1', 'depth = 1 (+ URLs one hop away)'), ('2', 'dept…

archivebox.core.forms.**_split_strip**(*value: str*, *delimiter: str*) → list[str]

archivebox.core.forms.**parse_tag_string**(*value: str | None*) → list[str]

Parse the legacy tag-editing format used by the admin tag field.

archivebox.core.forms.**edit_string_for_tag_names**(*tags*) → str

class archivebox.core.forms.**AddLinkForm**(**args*, ***kwargs*)

Bases: *archivebox.plugins.forms.PluginConfigFormMixin*, django.forms.Form

Initialization

allow_crawl_execution_config_fields

False

url

'CharField(...)'

tag

'CharField(...)'

depth

'ChoiceField(...)'

max_urls

'IntegerField(...)'

crawl_max_size

'CharField(...)'

crawl_timeout

'CharField(...)'

timeout

'CharField(...)'

snapshot_max_size

'CharField(...)'

delete_after

'CharField(...)'

crawl_max_concurrent_snapshots

'IntegerField(...)'

notes

'CharField(...)'

url_filters

'Field(...)'

main_plugins

'MultipleChoiceField(...)'

page_setup_plugins

'MultipleChoiceField(...)'

media_plugins

'MultipleChoiceField(...)'

text_plugins

'MultipleChoiceField(...)'

metadata_plugins

'MultipleChoiceField(...)'

postprocessing_plugins

'MultipleChoiceField(...)'

other_plugins

'MultipleChoiceField(...)'

schedule

'CharField(...)'

persona

'ModelChoiceField(...)'

permissions

'ChoiceField(...)'

start_paused

'BooleanField(...)'

config

'JSONField(...)'

clean_persona()

clean()

clean_url()

clean_url_filters()

clean_max_urls()

clean_crawl_max_size()

clean_crawl_timeout()

clean_timeout()

_clean_timeout_seconds(*raw_value*, *field_label: str*, *, *blank_value*)

clean_snapshot_max_size()

clean_delete_after()

clean_crawl_max_concurrent_snapshots()

clean_schedule()

class archivebox.core.forms.**TagWidget**

Bases: django.forms.TextInput

format_value(*value*)

class archivebox.core.forms.**TagField**

Bases: django.forms.CharField

widget

None

clean(*value*)

has_changed(*initial*, *data*)

archivebox.core.apps

Module Contents

Classes

CoreConfig

API

class archivebox.core.apps.**CoreConfig**

Bases: django.apps.AppConfig

name

'archivebox.core'

label

'core'

ready()

Register the archivebox.core.admin_site as the main django admin site

archivebox.core.snapshot_status

Module Contents

Functions

snapshot_status_values
normalize_snapshot_status
filter_snapshots_by_status

API

archivebox.core.snapshot_status.**snapshot_status_values**() → tuple[str, ...]

archivebox.core.snapshot_status.**normalize_snapshot_status**(*status: str | None*) → str | None

archivebox.core.snapshot_status.**filter_snapshots_by_status**(*queryset: django.db.models.QuerySet*, *status: str | None*) → django.db.models.QuerySet

archivebox.core.settings_logging

Module Contents

Classes

NoisyRequestsFilter

continues on next page

Table 153 – continued from previous page

DaphneCloseTimeoutFilter	Drop daphne's noisy "killed slow response after client disconnect" warning.
AsyncioCancelledShieldFilter	Drop asyncio's "CancelledError exception in shielded future" noise.
CustomOutboundWebhookLogFormatter	
StripANSIColorCodesFilter	

Data

IGNORABLE_URL_PATTERNS
ERROR_LOG
LOGS_DIR
LOG_LEVEL_DATABASE
LOG_LEVEL_REQUEST
SETTINGS_LOGGING

API

archivebox.core.settings_logging.**IGNORABLE_URL_PATTERNS**

None

class archivebox.core.settings_logging.**NoisyRequestsFilter**(*name=''*)

Bases: logging.Filter

filter(*record*) → bool

class archivebox.core.settings_logging.**DaphneCloseTimeoutFilter**(*name=''*)

Bases: logging.Filter

Drop daphne's noisy "killed slow response after client disconnect" warning.

Daphne emits this whenever a request handler is still running when the client disconnects (e.g. iframe gets navigated away mid-response while fetching favicon / screenshot / preview html). For our use case these are always benign — the disconnect is the browser cancelling a request, not a server-side fault — so we suppress them outright rather than spamming WARNING. Other daphne.server lines pass through unchanged.

Initialization

Initialize a filter.

Initialize with the name of the logger which, together with its children, will have its events allowed through the filter. If no name is specified, allow every event.

filter(*record*) → bool

class archivebox.core.settings_logging.**AsyncioCancelledShieldFilter**(*name=''*)

Bases: logging.Filter

Drop asyncio's "CancelledError exception in shielded future" noise.

When a browser disconnects mid-request, daphne cancels the asgi task and asgiref's `sync_to_async` shields the synchronous Django view via `asyncio.shield(exec_coro)`. The shield wakes up to find its parent cancelled and re-raises `CancelledError`; asyncio's default exception handler then logs the full traceback via the `asyncio` logger. There's nothing the server can do (the client is already gone), so these tracebacks are pure noise and cause hundreds of lines of spam per disconnect.

We match conservatively: only drop the specific shielded-future message that points back into asgiref's shield path. Other asyncio errors fall through unchanged.

Initialization

Initialize a filter.

Initialize with the name of the logger which, together with its children, will have its events allowed through the filter. If no name is specified, allow every event.

filter(*record*) → bool

class archivebox.core.settings_logging.**CustomOutboundWebhookLogFormatter**(*fmt=None*, *datefmt=None*, *style='%'*, *validate=True*, ***, *defaults=None*)

Bases: logging.Formatter

format(*record*)

class archivebox.core.settings_logging.**StripANSIColorCodesFilter**(*name=''*)

Bases: logging.Filter

_ansi_re

'compile(…)'

_bare_re

'compile(…)'

filter(*record*) → bool

archivebox.core.settings_logging.**ERROR_LOG**

None

archivebox.core.settings_logging.**LOGS_DIR**

None

archivebox.core.settings_logging.**LOG_LEVEL_DATABASE**

'WARNING'

archivebox.core.settings_logging.**LOG_LEVEL_REQUEST**

'WARNING'

archivebox.core.settings_logging.**SETTINGS_LOGGING**

None

archivebox.core.permissions

Module Contents

Functions

normalize_permissions
is_admin_user
get_snapshot_permissions

continues on next page

Table 155 – continued from previous page

can_view_snapshot
_persona_ids_for_permissions
filter_personas_by_permissions
filter_snapshots_by_permissions
public_snapshots_queryset
direct_snapshots_queryset

Data

PERMISSIONS_PUBLIC
PERMISSIONS_UNLISTED
PERMISSIONS_PRIVATE
PERMISSIONS_CHOICES
PERMISSIONS_VALUES
PERMISSIONS_META

API

archivebox.core.permissions.**PERMISSIONS_PUBLIC**

'public'

archivebox.core.permissions.**PERMISSIONS_UNLISTED**

'unlisted'

archivebox.core.permissions.**PERMISSIONS_PRIVATE**

'private'

archivebox.core.permissions.**PERMISSIONS_CHOICES**

((), (), ())

archivebox.core.permissions.**PERMISSIONS_VALUES**

None

archivebox.core.permissions.**PERMISSIONS_META**

None

archivebox.core.permissions.**normalize_permissions**(*permissions: object*, *, *default: str* = *PERMISSIONS_PRIVATE*) → str

archivebox.core.permissions.**is_admin_user**(*request: django.http.HttpRequest*) → bool

archivebox.core.permissions.**get_snapshot_permissions**(*snapshot*) → str

archivebox.core.permissions.**can_view_snapshot**(*request: django.http.HttpRequest*, *snapshot*) → bool

archivebox.core.permissions.**_persona_ids_for_permissions**(*allowed_permissions: set[str]*) → list[str]

archivebox.core.permissions.**filter_personas_by_permissions**(*queryset: django.db.models.QuerySet*, *allowed_permissions: set[str]*) → django.db.models.QuerySet

archivebox.core.permissions.**filter_snapshots_by_permissions**(*queryset: django.db.models.QuerySet*, *, *direct: bool = False*, *allowed_permissions: set[str] | None = None*) → django.db.models.QuerySet

archivebox.core.permissions.**public_snapshots_queryset**(*queryset: django.db.models.QuerySet*) → django.db.models.QuerySet

archivebox.core.permissions.**direct_snapshots_queryset**(*request: django.http.HttpRequest*, *queryset: django.db.models.QuerySet*) → django.db.models.QuerySet

archivebox.core.takeover_util

Coordinate local foreground processes without pretending to provide a distributed lock.

ArchiveBox wants one orchestrator per collection, but Process rows can only prove liveness for PIDs visible on the current machine/PID namespace. We therefore enforce one active runner per `(Machine, DATA_DIR)` locally, retire stale rows from sequential containers on that machine, and only warn about rows owned by a different machine. Foreign-machine rows must never block progress or be killed: multi-machine coordination belongs in the Crawl/Snapshot CAS claim layer, not in process takeover.

These helpers only hand local supervisord/runner ownership between CLI parents. They must not hold database transactions or filesystem locks while work runs.

Module Contents

Functions

current_command	
runtime_stack_owner	Return the live local parent allowed to own the server runtime stack.
command_owns_runtime_stack	
foreground_runner_owner	Return the newest live local parent allowed to borrow runner/sonic.
command_owns_foreground_runner	
ensure_daemon_stack	
live_runner_processes	Return locally verifiable runners and warn about unsupported overlap.
enter_single_runner_gate	Admit one active runner for this Machine and DATA_DIR using Process rows.
standby_until_runtime_stack_needed	
standby_until_foreground_runner_needed	

Data

RUNNER_ACTIVE_WORKER_TYPE
RUNNER_WAITING_WORKER_TYPE
RUNNER_GATE_WORKER_TYPES

API

archivebox.core.takeover_util.**RUNNER_ACTIVE_WORKER_TYPE**
'worker_runner'

archivebox.core.takeover_util.**RUNNER_WAITING_WORKER_TYPE**
'runner_waiting'

archivebox.core.takeover_util.**RUNNER_GATE_WORKER_TYPES**
()

archivebox.core.takeover_util.**current_command**(*process_type: str*, *, *data_dir: str | pathlib.Path*, *url: str | None = None*)

archivebox.core.takeover_util.**runtime_stack_owner**(*, *data_dir: str | pathlib.Path*, *exclude_id=None*)
Return the live local parent allowed to own the server runtime stack.

archivebox.core.takeover_util.**command_owns_runtime_stack**(*command*, *, *data_dir: str | pathlib.Path*) → bool

archivebox.core.takeover_util.**foreground_runner_owner**(*, *data_dir: str | pathlib.Path*, *exclude_id=None*)
Return the newest live local parent allowed to borrow runner/sonic.

archivebox.core.takeover_util.**command_owns_foreground_runner**(*command*, *, *data_dir: str | pathlib.Path*) → bool

archivebox.core.takeover_util.**ensure_daemon_stack**(*, *reason: str = ''*)

archivebox.core.takeover_util.**live_runner_processes**(*, *data_dir: str | pathlib.Path*)
Return locally verifiable runners and warn about unsupported overlap.

A Process row from another machine is observability only: its PID cannot be checked or signalled here, so it neither joins the local election nor gets mutated. A row for this same Machine from another PID namespace represents a previous sequential container under the supported model; warn, retire the unreachable row, and let the new container continue.

archivebox.core.takeover_util.**enter_single_runner_gate**(*command*, *, *data_dir: str | pathlib.Path*, *graceful_timeout: float = 5.0*) → bool
Admit one active runner for this Machine and DATA_DIR using Process rows.

The current process is a real OS process while it waits, so we keep its Process row RUNNING but mark worker_type=runner_waiting. Only the process that wins takeover is promoted to worker_type=worker_runner, which is protected by a partial unique DB constraint scoped to (Machine, DATA_DIR). Older locally verifiable runners are terminated and fully waited out before promotion, so runner work never overlaps on one machine. Foreign machines are intentionally outside this gate and only produce a warning above.

archivebox.core.takeover_util.**standby_until_runtime_stack_needed**(*command*, *, *data_dir: str | pathlib.Path*, *interval: float = 2.0*) → dict[str, object]

archivebox.core.takeover_util.**standby_until_foreground_runner_needed**(*command*, *, *data_dir: str | pathlib.Path*, *interval: float = 2.0*, *work_is_complete: collections.abc.Callable[[], bool] | None = None*) → dict[str, object]

archivebox.core.recovery_util

Module Contents

Functions

_is_signal_interrupted_exit	
_canonical_hook_name	
recover_orchestrator_state	

API

archivebox.core.recovery_util.**_is_signal_interrupted_exit**(*exit_code: int | None*) → bool

archivebox.core.recovery_util.**_canonical_hook_name**(*hook_name: str*) → str

archivebox.core.recovery_util.**recover_orchestrator_state**(*, *include_chrome: bool = False*, *crawl_id: str | None = None*) → dict[str, int]

archivebox.core.context_processors

Module Contents

Functions

get_static_cache_key	Version the admin stylesheet even when the checkout has uncommitted edits.
archivebox_globals	

API

archivebox.core.context_processors.**get_static_cache_key**() → str

Version the admin stylesheet even when the checkout has uncommitted edits.

archivebox.core.context_processors.**archivebox_globals**(*request*)

archivebox.core.views

Module Contents

Classes

SnapshotReplayAuthView	Admin-only handoff that lets a snap host mint its own replay cookie.
HomepageView	
SnapshotView	
SnapshotPathView	Serve snapshots by the new URL scheme: /<username>/<YYYYMMDD>/<domain>/<uuid>/…

continues on next page

Table 161 – continued from previous page

SnapshotHostView	Serve snapshot directory contents on <snapshot-subdomain>.<listen_host>/<path>.
SnapshotReplayView	Serve snapshot directory contents on a one-domain replay path.
OriginalDomainHostView	Serve responses from the most recent snapshot when using <domain>.<listen_host>/<path>.
OriginalDomainReplayView	Serve original-domain replay content on a one-domain replay path.
PublicIndexView	Keep browsing responsive while the same small server is archiving.
AddView	
WebAddView	
HealthCheckView	A Django view that renders plain text "OK" for service discovery tools

Functions

_files_index_target	
_find_snapshot_by_ref	
_admin_login_redirect_or_forbidden	
_replay_cookie_name	
_clean_replay_next	Only allow same-snap relative replay paths; grants must never redirect off-host.
_replay_payload_is_valid	A replay cookie is not its own auth source; it must point at a live admin session.
_has_replay_cookie	
_private_snapshot_auth_redirect	
_replay_auth_response	
_safe_archive_relpath	
_resolve_archiveresult_relpath	Resolve a declared output path and its exact historical fallbacks.
_plugin_full_preview_response	Render an explicit plugin full template as a trusted preview wrapper.
_trusted_snapshot_frame_csp	
_plugin_card_document_response	Serve the existing card template from the snapshot origin, not web.*.
_visible_response_snapshots_for_domain	
_latest_response_match	
_latest_responses_root	
_original_request_url	
_serve_responses_path	
_build_snapshot_replay_response	
_serve_snapshot_replay	
_serve_original_domain_replay	
live_config_list_view	
live_config_value_view	

Data

REPLAY_AUTH_SALT
REPLAY_COOKIE_PREFIX
REPLAY_GRANT_MAX_AGE

API

archivebox.core.views.**_files_index_target**(*snapshot:* archivebox.core.models.Snapshot, *archivefile: str | None*) → str

archivebox.core.views.**_find_snapshot_by_ref**(*snapshot_ref: str*) → *archivebox.core.models.Snapshot* | None

archivebox.core.views.**_admin_login_redirect_or_forbidden**(*request: django.http.HttpRequest*)

archivebox.core.views.**REPLAY_AUTH_SALT**

'archivebox.private-snapshot-replay'

archivebox.core.views.**REPLAY_COOKIE_PREFIX**

None

archivebox.core.views.**REPLAY_GRANT_MAX_AGE**

60

archivebox.core.views.**_replay_cookie_name**(*snapshot:* archivebox.core.models.Snapshot) → str

archivebox.core.views.**_clean_replay_next**(*path: str | None*) → str

Only allow same-snap relative replay paths; grants must never redirect off-host.

archivebox.core.views.**_replay_payload_is_valid**(*payload: dict*, *snapshot:* archivebox.core.models.Snapshot) → bool

A replay cookie is not its own auth source; it must point at a live admin session.

Replayed pages can execute hostile JS, so admin cookies stay host-only on admin.. *The snap host gets only this host-only HttpOnly cookie, and every request checks that the original Django session still exists and still belongs to an active staff user. Logout, session expiry, user deletion/deactivation, or password auth-hash rotation all make the replay cookie inert without needing admin.* to delete a cookie on snap-*.

archivebox.core.views.**_has_replay_cookie**(*request: django.http.HttpRequest*, *snapshot:* archivebox.core.models.Snapshot) → bool

archivebox.core.views.**_private_snapshot_auth_redirect**(*request: django.http.HttpRequest*, *snapshot:* archivebox.core.models.Snapshot, *path: str = ''*, ***, *preserve_query: bool = True*)

archivebox.core.views.**_replay_auth_response**(*request: django.http.HttpRequest*, *snapshot:* archivebox.core.models.Snapshot)

class archivebox.core.views.**SnapshotReplayAuthView**

Bases: `django.views.View`

Admin-only handoff that lets a snap host mint its own replay cookie.

admin.* cannot set a host-only cookie for snap-* (browsers forbid that), and widening the real Django session cookie to *.archivebox.localhost would let XSS in replayed pages hit the admin UI. Instead admin.* proves the user is logged in with a short URL grant, then snap-* validates it and sets a snap-host-only cookie.

get(*request: django.http.HttpRequest*)

class archivebox.core.views.**HomepageView**

Bases: django.views.View

get(*request*)

class archivebox.core.views.**SnapshotView**

Bases: django.views.View

static find_snapshots_for_url(*path: str*, *, *allow_fallback: bool = True*)

Return a queryset of snapshots matching a URL-ish path. URL only — never tries ID matching.

Use `find_snapshots_for_id` separately if you also want to match by snapshot UUID.

static find_snapshots_for_id(*slug: str*)

Return a queryset of snapshots matching a (possibly truncated) UUID via prefix or suffix.

Strips non-hex characters from `slug` (so input with or without hyphens both work). Requires at least 8 hex chars — shorter inputs return an empty queryset to avoid scanning the entire snapshots table on too-broad matches.

static render_live_index(*request*, *snapshot*)

get(*request*, *path*)

class archivebox.core.views.**SnapshotPathView**

Bases: django.views.View

Serve snapshots by the new URL scheme: /<username>/<YYYYMMDD>/<domain>/<uuid>/...

get(*request*, *username: str*, *date: str | None = None*, *domain: str | None = None*, *snapshot_id: str | None = None*, *path: str = ''*, *url: str | None = None*)

archivebox.core.views.**_safe_archive_relpath**(*path: str*) → str | None

archivebox.core.views.**_resolve_archiveresult_relpath**(*snapshot:* archivebox.core.models.Snapshot, *rel_path: str*) → tuple[str, *archivebox.core.models.ArchiveResult* | None, tuple[str, ...]]

Resolve a declared output path and its exact historical fallbacks.

archivebox.core.views.**_plugin_full_preview_response**(*request: django.http.HttpRequest*, *snapshot:* archivebox.core.models.Snapshot, *rel_path: str*, *result:* archivebox.core.models.ArchiveResult | *None*) → django.http.HttpResponse | None

Render an explicit plugin full template as a trusted preview wrapper.

archivebox.core.views.**_trusted_snapshot_frame_csp**(*request: django.http.HttpRequest*) → str

archivebox.core.views.**_plugin_card_document_response**(*request: django.http.HttpRequest*, *snapshot:* archivebox.core.models.Snapshot, *result_id: str*) → django.http.HttpResponse

Serve the existing card template from the snapshot origin, not web.*.

srcdoc and inline card scripts inherit the containing page's origin. This frame route gives them same-origin access to this snapshot's saved files without opening CORS on raw archives or sharing the admin session cookie. The parent web page receives only the one-bit login hint; delete buttons navigate to admin.* for a real authenticated confirmation.

archivebox.core.views.**_visible_response_snapshots_for_domain**(*request: django.http.HttpRequest*, *domain: str*) → list[*archivebox.core.models.Snapshot*]

archivebox.core.views.**_latest_response_match**(*snapshots: list[*archivebox.core.models.Snapshot*]*, *domain: str*, *rel_path: str*) → tuple[*archivebox.core.models.Snapshot*, pathlib.Path, pathlib.Path] | None

archivebox.core.views.**_latest_responses_root**(*snapshots: list[*archivebox.core.models.Snapshot*]*, *domain: str*) → tuple[*archivebox.core.models.Snapshot*, pathlib.Path] | None

archivebox.core.views.**_original_request_url**(*domain: str*, *path: str = ''*, *query_string: str = ''*) → str

archivebox.core.views.**_serve_responses_path**(*request*, *responses_root: pathlib.Path*, *rel_path: str*, *show_indexes: bool*)

archivebox.core.views.**_build_snapshot_replay_response**(*request: django.http.HttpRequest*, *snapshot:* archivebox.core.models.Snapshot, *path: str = ''*)

archivebox.core.views.**_serve_snapshot_replay**(*request: django.http.HttpRequest*, *snapshot:* archivebox.core.models.Snapshot, *path: str = ''*)

archivebox.core.views.**_serve_original_domain_replay**(*request: django.http.HttpRequest*, *domain: str*, *path: str = ''*)

class archivebox.core.views.**SnapshotHostView**

Bases: django.views.View

Serve snapshot directory contents on <snapshot-subdomain>.<listen_host>/<path>.

get(*request*, *snapshot_id: str*, *path: str = ''*)

class archivebox.core.views.**SnapshotReplayView**

Bases: django.views.View

Serve snapshot directory contents on a one-domain replay path.

get(*request*, *snapshot_id: str*, *path: str = ''*)

class archivebox.core.views.**OriginalDomainHostView**

Bases: django.views.View

Serve responses from the most recent snapshot when using <domain>.<listen_host>/<path>.

get(*request*, *domain: str*, *path: str = ''*)

class archivebox.core.views.**OriginalDomainReplayView**

Bases: django.views.View

Serve original-domain replay content on a one-domain replay path.

get(*request*, *domain: str*, *path: str = ''*)

class archivebox.core.views.**PublicIndexView**

Bases: django.views.generic.list.ListView

Keep browsing responsive while the same small server is archiving.

Performance takes priority over perfect initial totals: on a 1 vCPU / 1 GB host with hundreds of thousands of snapshots, even an indexed count can consume the entire roughly 0.5-0.7 second page-load budget. Benchmark with the runner active, since it shares the CPU, memory, and SQLite database.

The unfiltered list therefore renders rows first and accepts missing/stale display totals. A later ordinary reload can show the background-cached count; do not restore a synchronous COUNT or add polling just to fill it in.

template_name

'public_index.html'

model

None

ordering: ClassVar[list[str]]

['-bookmarked_at', '-created_at']

paginator_class

None

get_paginate_by(*queryset*)

_base_public_snapshot_fields() → tuple[str, ...]

_ordered_public_page_from_order_index(*, *page_number: int*, *page_size: int*) → list[*archivebox.core.models.Snapshot*]

paginate_queryset(*queryset*, *page_size*)

get_context_data(***kwargs*)

get_queryset(***kwargs*)

get(**args*, ***kwargs*)

class archivebox.core.views.**AddView**

Bases: django.contrib.auth.mixins.UserPassesTestMixin, django.views.generic.FormView

template_name

'add.html'

form_class

None

get_initial()

Prefill the AddLinkForm with the 'url' GET parameter

get_form_kwargs()

test_func()

post(*request: django.http.HttpRequest*, **args: object*, ***kwargs: object*)

_can_override_crawl_config() → bool

_get_custom_config_overrides(*form:* archivebox.core.forms.AddLinkForm) → dict

get_context_data(***kwargs*)

_create_crawl_from_form(*form*, *, *created_by_id=None*) → *archivebox.crawls.models.Crawl*

form_valid(*form*)

class archivebox.core.views.**WebAddView**

Bases: *archivebox.core.views.AddView*

_latest_snapshot_for_url(*requested_url: str*)

_normalize_add_url(*requested_url: str*) → str

dispatch(*request*, **args*, ***kwargs*)

get(*request: django.http.HttpRequest*, **args: object*, ***kwargs: object*)

class archivebox.core.views.**HealthCheckView**

Bases: django.views.View

A Django view that renders plain text "OK" for service discovery tools

get(*request*)

Handle a GET request

archivebox.core.views.**live_config_list_view**(*request: django.http.HttpRequest*, ***kwargs*) → admin_data_views.typing.TableContext

archivebox.core.views.**live_config_value_view**(*request: django.http.HttpRequest*, *key: str*, ***kwargs*) → admin_data_views.typing.ItemContext

archivebox.core.urls

Module Contents

Functions

_raise_test_error

Data

urlpatterns

API

archivebox.core.urls.**urlpatterns**

None

archivebox.core.urls.**_raise_test_error**(*_request: django.http.HttpRequest*)

archivebox.core.shutdown_util

Shared foreground signal and child-process cleanup contracts.

Shutdown intent and process exit are separate: takeover must preserve the replacement owner's work, while a user abort must not later resume the crawl. Callers own that policy; this module preserves the received signal and stops only the process objects they explicitly supply.

Cooperative boundaries are essential, not just quieter exception handling. A real Linux CI regression interrupted Popen.poll() with SIGINT and stranded its wait lock. SIGKILL stopped the child (it became a zombie), but Popen.wait() still timed out. Raising a different exception or waiting longer cannot repair that interrupted critical section. Polling loops must record the signal first and honor it between operations; async owners may cancel their own task while their loop is running. Once exiting, the last-resort process cleanup must stay synchronous: the original loop may be closed, and creating a replacement loop cannot recover tasks or resources owned by it.

Keep immediate signal notices and repeated-signal force exit independent of the event bus: it may already be draining or closed during shutdown. Resumable pause/retry/skip belongs to the existing crawl controller and must not set sticky command-shutdown state. See foreground_shutdown_signals below for both cooperative and legacy blocking callers; do not change all callers' policy at once without supplying their interruption boundaries.

Module Contents

Classes

ShutdownSignalState	Tracks the exact OS signal that asked a foreground command to exit.

Functions

raise_if_shutdown_requested	Honor sticky shutdown intent at a caller-owned interruption boundary.
configured_stopwaitsecs	Return a deterministic shutdown bound from generated worker definitions.
wait_popen_and_kill_children	Reap our own Popen child and stop its explicitly captured descendants.
wait_psutil_and_kill_children	Wait for a psutil parent and then hard-kill any surviving descendants.
kill_remaining_processes	
foreground_shutdown_signals	Keep shutdown intent visible until the owning foreground command unwinds.
foreground_parent_watchdog	Ask a foreground command to exit if its launcher/wrapper disappears.

Data

_active_shutdown_state

API

class archivebox.core.shutdown_util.**ShutdownSignalState**

Tracks the exact OS signal that asked a foreground command to exit.

signal_name: str | None

None

archivebox.core.shutdown_util.**_active_shutdown_state: *archivebox.core.shutdown_util.ShutdownSignalState* | None**

None

archivebox.core.shutdown_util.**raise_if_shutdown_requested**() → None

Honor sticky shutdown intent at a caller-owned interruption boundary.

archivebox.core.shutdown_util.**configured_stopwaitsecs**(*workers: list[dict[str, str]] | tuple[dict[str, str], ...]*, ***, *default: int = 5*, *buffer: int = 5*) → int

Return a deterministic shutdown bound from generated worker definitions.

archivebox.core.shutdown_util.**wait_popen_and_kill_children**(*proc: subprocess.Popen*, *children: list[psutil.Process]*, ***, *timeout: float*, *kill_timeout: float = 2.0*) → None

Reap our own Popen child and stop its explicitly captured descendants.

A dead child still needs its parent to reap it. Keep wait failures visible; bypassing Popen's lock would hide the interrupted-owner bug described above. Descendants must still be cleaned up when that wait fails.

archivebox.core.shutdown_util.**wait_psutil_and_kill_children**(*proc: psutil.Process*, *children: list[psutil.Process]*, ***, *timeout: float*, *kill_timeout: float = 2.0*) → None

Wait for a psutil parent and then hard-kill any surviving descendants.

archivebox.core.shutdown_util.**kill_remaining_processes**(*processes: list[psutil.Process]*, ***, *timeout: float = 2.0*) → None

archivebox.core.shutdown_util.**foreground_shutdown_signals**(*handled_signals: tuple[signal.Signals, ...] = (signal.SIGHUP, signal.SIGINT, signal.SIGTERM)*, ***, *first_signal_message: str | None = '\n[] Got {signal_name}, stopping gracefully...\n'*, *on_signal: collections.abc.Callable[[signal.Signals], None] | None = None*, *interrupt_handlers: dict[signal.Signals, collections.abc.Callable[[], None]] | None = None*, *raise_on_first_signal: bool = True*) → collections.abc.Iterator[*archivebox.core.shutdown_util.ShutdownS*

Keep shutdown intent visible until the owning foreground command unwinds.

Signal state was added because inner loops can swallow KeyboardInterrupt while their caller still needs to stop its children. Cooperative callers use raise_on_first_signal=False and check raise_if_shutdown_requested() between operations, or cancel their owned async task through on_signal. Raising from the handler can otherwise interrupt library locks or partially sent I/O. Blocking callers without a cooperative boundary retain immediate exceptions.

archivebox.core.shutdown_util.**foreground_parent_watchdog**(*, *enabled: bool = True*, *check_interval: float = 2.0*, *shutdown_signal: signal.Signals = signal.SIGTERM*) → collections.abc.Iterator[None]

Ask a foreground command to exit if its launcher/wrapper disappears.

`uv run archivebox ...` and similar wrappers can be killed without delivering a signal to the real Python child. If that child keeps crawling as an orphan, it can hold SQLite write locks long after the user-facing command timed out. This watchdog is only for foreground command lifetimes; daemon/supervisord workers should not use it because their parent may intentionally hand them off.

archivebox.core.admin_tags

Module Contents

Classes

TagInline
TagAdminForm
TagAdmin

Functions

register_admin

API

class archivebox.core.admin_tags.**TagInline**

Bases: `django.contrib.admin.TabularInline`

model

None

fields

('id', 'tag')

extra

1

max_num

1000

autocomplete_fields

('tag',)

class archivebox.core.admin_tags.**TagAdminForm**

Bases: `django.forms.ModelForm`

class Meta

model

None

fields

'**all**'

widgets

None

clean_name()

class archivebox.core.admin_tags.**TagAdmin**

Bases: *archivebox.base_models.admin.BaseModelAdmin*

form

None

change_list_template

'admin/core/tag/change_list.html'

change_form_template

'admin/core/tag/change_form.html'

list_display

('name', 'num_snapshots', 'created_at', 'created_by')

list_filter

('created_at', 'created_by')

search_fields

('id', 'name')

readonly_fields

('id', 'created_at', 'modified_at', 'snapshots')

actions

['delete_selected']

ordering

['name', 'id']

fieldsets

(('Tag',), ('Metadata',), ('Recent Snapshots',))

add_fieldsets

(('Tag',), ('Metadata',))

get_fieldsets(*request: django.http.HttpRequest*, *obj:* archivebox.core.models.Tag | *None = None*)

changelist_view(*request: django.http.HttpRequest*, *extra_context=None*)

render_change_form(*request*, *context*, *add=False*, *change=False*, *form_url=''*, *obj=None*)

response_add(*request: django.http.HttpRequest*, *obj:* archivebox.core.models.Tag, *post_url_continue=None*)

response_change(*request: django.http.HttpRequest*, *obj:* archivebox.core.models.Tag)

_redirect_to_changelist(*query: str = ''*) → django.http.HttpResponseRedirect

snapshots(*tag:* archivebox.core.models.Tag)

num_snapshots(*tag:* archivebox.core.models.Tag)

archivebox.core.admin_tags.**register_admin**(*admin_site*)

archivebox.core.setup_wizard

Module Contents

Functions

_request_scheme	
get_setup_wizard_context	Build the first-run setup context when BASE_URL is unset.
get_base_url_mismatch_context	Describe a request origin that does not resolve to the configured base.

API

archivebox.core.setup_wizard.**_request_scheme**(*request*) → str

archivebox.core.setup_wizard.**get_setup_wizard_context**(*request*, *config*) → dict[str, Any]

Build the first-run setup context when BASE_URL is unset.

archivebox.core.setup_wizard.**get_base_url_mismatch_context**(*request*, *config*) → dict[str, str] | None

Describe a request origin that does not resolve to the configured base.

archivebox.core.admin_site

Module Contents

Classes

ArchiveBoxLoginView	
ArchiveBoxAdmin	

Functions

register_admin_site	Replace the default admin site with our custom ArchiveBox admin site.

Data

archivebox_admin

API

class archivebox.core.admin_site.**ArchiveBoxLoginView**

Bases: django.contrib.auth.views.LoginView

get_redirect_url() → str

class archivebox.core.admin_site.**ArchiveBoxAdmin**

Bases: django.contrib.admin.AdminSite

site_header

'ArchiveBox'

index_title

'Admin Views'

site_title

'Admin'

namespace

'admin'

each_context(*request: django.http.HttpRequest*) → dict[str, Any]

static _format_object_count(*count: int*) → tuple[int, str, str]

_set_model_object_count(*models_by_table: dict[str, list[dict[str, Any]]]*, *table: str*, *count: int*, *title: str | None = None*) → None

get_app_list(*request: django.http.HttpRequest*, *app_label: str | None = None*) → list[admin_data_views.typing.AppDict]

admin_data_index_view(*request: django.http.HttpRequest*, ***kwargs: Any*) → django.template.response.TemplateResponse

login(*request: django.http.HttpRequest*, *extra_context: dict[str, Any] | None = None*) → django.template.response.TemplateResponse

index(*request: django.http.HttpRequest*, *extra_context: dict[str, Any] | None = None*) → django.template.response.TemplateResponse

get_admin_data_urls() → list[URLResolver | URLPattern]

get_urls() → list[URLResolver | URLPattern]

archivebox.core.admin_site.**archivebox_admin**

'ArchiveBoxAdmin(...)'

archivebox.core.admin_site.**register_admin_site**()

Replace the default admin site with our custom ArchiveBox admin site.

archivebox.core.routes_util

Module Contents

Functions

split_host_port
_normalize_base_url
normalize_base_url
_csrf_trusted_origins

continues on next page

Table 175 – continued from previous page

derive_base_url_from_csrf	Pick a single CSRF_TRUSTED_ORIGINS entry to act as the implicit BASE_URL.
get_listen_host	
_with_port	
strip_role_subdomain	Strip leading `admin.` / `web.` / `api.` / `snap-*.` labels from a host (preserving the port). Strips repeatedly so an already-compounded host like `snap-X.snap-X.<base>` reduces all the way down to `<base>`.
_is_local_bind_host	
canonical_base_host_for_request	Strip role subdomains and remap loopback hostnames to `archivebox.localhost`.
_root_host_from_listen	
get_base_url	
get_base_host	
_build_base_host	
get_admin_host	
get_web_host	
get_api_host	
get_snapshot_subdomain	
get_snapshot_host	
get_original_host	
is_snapshot_subdomain	
get_snapshot_lookup_key	
get_listen_subdomain	
host_matches	
is_allowed_archivebox_redirect_url	Allow login redirects to this host or to the explicit ArchiveBox BASE_URL family.
_scheme_from_request	
_build_base_url_for_host	
get_admin_base_url	
get_web_base_url	
get_api_base_url	
get_snapshot_base_url	
build_admin_url	
build_web_url	
build_snapshot_url	
build_snapshot_detail_path	Build the live Django snapshot-detail path and optional output anchor.
build_snapshot_index_path	Build a path to the literal exported `index.html` snapshot file.
get_snapshot_output_anchor	Map root-relative legacy outputs to the plugin card that displays them.
build_snapshot_detail_url	Build a request-aware URL for trusted snapshot UI on the web origin.
build_snapshot_role_output_url	Build the first available output URL for a plugin-owned snapshot role.
build_snapshot_files_url	Build a request-aware directory-browser URL on the isolated replay origin.
build_snapshot_zip_url	Build a request-aware ZIP download URL for a snapshot directory.
build_original_url	
_build_url	

continues on next page

Table 175 – continued from previous page

_replace_url_query_params

Data

_SNAPSHOT_ID_RE
_SNAPSHOT_SUBDOMAIN_RE
_ROLE_SUBDOMAIN_LABELS
_URL_PATH_SAFE
_URL_FRAGMENT_SAFE

API

archivebox.core.routes_util.**_SNAPSHOT_ID_RE**

'compile(...)'

archivebox.core.routes_util.**_SNAPSHOT_SUBDOMAIN_RE**

'compile(...)'

archivebox.core.routes_util.**_ROLE_SUBDOMAIN_LABELS**

('admin', 'web', 'api')

archivebox.core.routes_util.**_URL_PATH_SAFE**

"/@-._~!$&'()*+,;="

archivebox.core.routes_util.**_URL_FRAGMENT_SAFE**

"/@-._~!$&'()*+,;="

archivebox.core.routes_util.**split_host_port**(*host: str*) → tuple[str, str | None]

archivebox.core.routes_util.**_normalize_base_url**(*value: str | None*) → str

archivebox.core.routes_util.**normalize_base_url**(*value: str | None*) → str

archivebox.core.routes_util.**_csrf_trusted_origins**(*config*) → list[str]

archivebox.core.routes_util.**derive_base_url_from_csrf**(*config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

Pick a single CSRF_TRUSTED_ORIGINS entry to act as the implicit BASE_URL.

0.7.3 → 0.9.0 upgrade path: any reverse-proxied 0.7.3 deployment already had `CSRF_TRUSTED_ORIGINS=https://archive.example.com` set (required for admin login to work). On upgrade, `BASE_URL` is the new knob — but it defaults to empty, and falling through to `BIND_ADDR` produces an unreachable URL like `http://0.0.0.0:5797`. If the user has exactly one CSRF origin we treat it as the implicit BASE_URL so links/redirects keep pointing at the public hostname they already configured.

Returns `""` when the inference is ambiguous (multiple origins) or impossible (none set) so callers fall through to their next strategy.

archivebox.core.routes_util.**get_listen_host**(*config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**_with_port**(*host: str*, *port: str | None*) → str

archivebox.core.routes_util.**strip_role_subdomain**(*host: str*) → str

Strip leading `admin.` / `web.` / `api.` / `snap-*.` labels from a host (preserving the port). Strips repeatedly so an already-compounded host like `snap-X.snap-X.<base>` reduces all the way down to `<base>`.

Used when we want to recover the canonical base host from a request that arrived on a role subdomain — otherwise builders that prepend their own role label (e.g. `snap-X.`) compound onto the existing prefix and you get `snap-X.snap-X.snap-X.<base>` on every click.

archivebox.core.routes_util.**_is_local_bind_host**(*host: str*) → bool

archivebox.core.routes_util.**canonical_base_host_for_request**(*request_host: str*) → str

Strip role subdomains and remap loopback hostnames to `archivebox.localhost`.

Used by the banner suggestion and the in-browser pin endpoint: when the user is hitting the server on raw `localhost:5797` or `127.0.0.1:5797` we want to suggest the wildcard-friendly `archivebox.localhost` family instead, so the eventual pinned `BASE_URL` plays nicely with subdomain routing without forcing the user to add a /etc/hosts entry.

archivebox.core.routes_util.**_root_host_from_listen**(*config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_base_url**(*request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_base_host**(*request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**_build_base_host**(*subdomain: str | None*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_admin_host**(*config: dict[str, Any] | None = None*, *request=None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_web_host**(*config: dict[str, Any] | None = None*, *request=None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_api_host**(*config: dict[str, Any] | None = None*, *request=None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_snapshot_subdomain**(*snapshot_id: str*) → str

archivebox.core.routes_util.**get_snapshot_host**(*snapshot_id: str*, *config: dict[str, Any] | None = None*, *request=None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_original_host**(*domain: str*, *config: dict[str, Any] | None = None*, *request=None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**is_snapshot_subdomain**(*subdomain: str*) → bool

archivebox.core.routes_util.**get_snapshot_lookup_key**(*snapshot_ref: str*) → str

archivebox.core.routes_util.**get_listen_subdomain**(*request_host: str*, *config: dict[str, Any] | None = None*, *request=None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**host_matches**(*request_host: str*, *target_host: str*) → bool

archivebox.core.routes_util.**is_allowed_archivebox_redirect_url**(*url: str | None*, *request=None*, *config: dict[str, Any] | None = None*) → bool

Allow login redirects to this host or to the explicit ArchiveBox BASE_URL family.

Django already rejects arbitrary absolute `next=` URLs for the current host. ArchiveBox also has legitimate cross-subdomain handoffs (admin/web/api/snap-*), but only when BASE_URL is pinned. Without an explicit BASE_URL we do not widen trust based on a request Host header that may be user supplied.

archivebox.core.routes_util.**_scheme_from_request**(*request=None*, *config: dict[str, Any] | None = None*) → str

archivebox.core.routes_util.**_build_base_url_for_host**(*host: str*, *request=None*, *config: dict[str, Any] | None = None*) → str

archivebox.core.routes_util.**get_admin_base_url**(*request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_web_base_url**(*request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_api_base_url**(*request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**get_snapshot_base_url**(*snapshot_id: str*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**build_admin_url**(*path: str = ''*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**build_web_url**(*path: str = ''*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**build_snapshot_url**(*snapshot_id: str*, *path: str = ''*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**build_snapshot_detail_path**(*archive_path: str*, *output_path: str = ''*, *prefix: str = '/'*) → str

Build the live Django snapshot-detail path and optional output anchor.

archivebox.core.routes_util.**build_snapshot_index_path**(*archive_path: str*, *output_path: str = ''*, *prefix: str = '/'*) → str

Build a path to the literal exported `index.html` snapshot file.

archivebox.core.routes_util.**get_snapshot_output_anchor**(*plugin: str*, *output_path: str*) → str

Map root-relative legacy outputs to the plugin card that displays them.

archivebox.core.routes_util.**build_snapshot_detail_url**(*archive_path: str*, *output_path: str = ''*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

Build a request-aware URL for trusted snapshot UI on the web origin.

archivebox.core.routes_util.**build_snapshot_role_output_url**(*snapshot*, *role: str*, *request=None*, *config: dict[str, Any] | None = None*, *archive_results=None*, *fallback_to_default: bool = False*, ***config_kwargs: Any*) → str

Build the first available output URL for a plugin-owned snapshot role.

archivebox.core.routes_util.**build_snapshot_files_url**(*snapshot_id: str*, *path: str = ''*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

Build a request-aware directory-browser URL on the isolated replay origin.

archivebox.core.routes_util.**build_snapshot_zip_url**(*snapshot_id: str*, *path: str = ''*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

Build a request-aware ZIP download URL for a snapshot directory.

archivebox.core.routes_util.**build_original_url**(*domain: str*, *path: str = ''*, *request=None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

archivebox.core.routes_util.**_build_url**(*base_url: str*, *path: str*) → str

archivebox.core.routes_util.**_replace_url_query_params**(*url: str*, ***params: str*) → str

archivebox.core.models

Module Contents

Classes

UngroupedSubquery	Scalar subquery that should not be copied into the outer GROUP BY.
Tag	
SnapshotTag	
SnapshotQuerySet	Custom QuerySet for Snapshot model with export methods that persist through .filter() etc.
SnapshotManager	Manager for Snapshot model - uses SnapshotQuerySet for chainable methods
Snapshot	
ArchiveResult	

API

exception archivebox.core.models.**SnapshotMigrationError**

Bases: RuntimeError

Raised when a snapshot filesystem migration fails validation.

Initialization

Initialize self. See help(type(self)) for accurate signature.

class archivebox.core.models.**UngroupedSubquery**

Bases: django.db.models.Subquery

Scalar subquery that should not be copied into the outer GROUP BY.

get_group_by_cols()

class archivebox.core.models.**Tag**

Bases: *archivebox.base_models.models.ModelWithUUID*

id

'AutoField(...)'

created_by

'ForeignKey(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

name

'CharField(...)'

classmethod get_or_create_by_name(*name: str*, *, *defaults: collections.abc.Mapping[str, Any] | None = None*) → tuple[*archivebox.core.models.Tag*, bool]

snapshot_set: django.db.models.Manager[Snapshot]

None

class Meta

Bases: *archivebox.base_models.models.ModelWithUUID.Meta*

app_label

'core'

verbose_name

'Tag'

verbose_name_plural

'Tags'

__str__()

save(**args*, ***kwargs*)

property slug: str

ASCII-safe slugified form of the tag name (derived, not stored).

property api_url: str

to_json() → dict

Convert Tag model instance to a JSON-serializable dict.

static from_json(*record: dict[str, Any]*, *overrides: dict[str, Any] | None = None*)

Create/update Tag from JSON dict.

Args: record: JSON dict with 'name' field overrides: Optional dict with 'snapshot' to auto-attach tag

Returns: Tag instance or None

class archivebox.core.models.**SnapshotTag**

Bases: django.db.models.Model

id

'AutoField(...)'

snapshot

'ForeignKey(...)'

tag

'ForeignKey(...)'

class Meta

app_label

'core'

db_table

'core_snapshot_tags'

unique_together: ClassVar[list[tuple[str, str]]]

[('snapshot', 'tag')]

class archivebox.core.models.**SnapshotQuerySet**

Bases: *archivebox.base_models.models.PagedQuerySet*

Custom QuerySet for Snapshot model with export methods that persist through .filter() etc.

request_delete() → int

delete()

bulk_create(*objs*, **args*, ***kwargs*)

FILTER_TYPES: ClassVar[dict[str, Any]]

None

FILTER_TYPE_CHOICES

'tuple(...)'

FILTER_ARG_KEYS

('after', 'before', 'filter_type', 'filter_patterns', 'status', 'url__icontains', 'url__istartswith'...

SPECIAL_FILTER_ARG_KEYS

'frozenset(...)'

filter_by_patterns(*patterns: list[str]*, *filter_type: str = 'exact'*) → *archivebox.core.models.SnapshotQuerySet*

Filter snapshots by URL patterns using specified filter type

search(***kwargs*) → *archivebox.core.models.SnapshotQuerySet*

to_json(*with_headers: bool = False*) → str

Generate JSON index from snapshots

to_csv(*cols: list[str] | None = None*, *header: bool = True*, *separator: str = ','*, *ljust: int = 0*) → str

Generate CSV output from snapshots

to_html(*with_headers: bool = True*) → str

Generate main index HTML from snapshots

class archivebox.core.models.**SnapshotManager**

Bases: models.Manager.from_queryset(*SnapshotQuerySet*)

Manager for Snapshot model - uses SnapshotQuerySet for chainable methods

filter(**args*, ***kwargs*)

get_queryset()

remove(*atomic: bool = False*) → tuple

Remove snapshots from the database

class archivebox.core.models.**Snapshot**

Bases: *archivebox.base_models.models.ModelWithDeleteAfter*, *archivebox.base_models.models.ModelWithOutputDir*, *archivebox.base_models.models.ModelWithConfig*, *archivebox.base_models.models.ModelWithNotes*, *archivebox.base_models.models.ModelWithHealthStats*, *archivebox.workers.models.ModelWithQueue*

BROWSER_EXTENSION_UPLOAD_HOOK_NAME

'on_Snapshot__archivebox_browser_extension_upload'

DELETING_STATE

'deleting'

id

'CompactUUIDField(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

url

'TextField(...)'

timestamp

'CharField(...)'

bookmarked_at

'DateTimeField(...)'

crawl: *archivebox.crawls.models.Crawl*

'ForeignKey(...)'

parent_snapshot

'ForeignKey(...)'

title

'CharField(...)'

downloaded_at

'DateTimeField(...)'

depth

'PositiveSmallIntegerField(...)'

fs_version

'CharField(...)'

retry_at
'RetryAtField(...)'

status
'StatusField(...)'

config
'JSONField(...)'

permissions
'GeneratedField(...)'

output_size
'BigIntegerField(...)'

notes
'TextField(...)'

tags
'ManyToManyField(...)'

state_field_name
'status'

retry_at_field_name
'retry_at'

StatusChoices
None

INITIAL_STATE
None

ACTIVE_STATE
None

FINAL_STATES
()

FINAL_OR_ACTIVE_STATES
()

active_state
None

delete_after_final_statuses
()

RUNNABLE_STATES
()

OPEN_STATES
()

crawl_id: uuid.UUID
None

parent_snapshot_id: uuid.UUID | None
None

_prefetched_objects_cache: dict[str, Any]

None

objects

'SnapshotManager(...)'

archiveresult_set: django.db.models.Manager[*ArchiveResult*]

None

add_tag_ids(*tag_ids: collections.abc.Iterable[int | str]*) → None

remove_tag_ids(*tag_ids: collections.abc.Iterable[int | str]*) → int

class Meta

Bases: *archivebox.base_models.models.ModelWithDeleteAfter.Meta*, *archivebox.base_models.models.ModelWithOutputDir.Meta*, *archivebox.base_models.models.ModelWithConfig.Meta*, *archivebox.base_models.models.ModelWithNotes.Meta*, *archivebox.base_models.models.ModelWithHealthStats.Meta*, *archivebox.workers.models.ModelWithQueue.Meta*

app_label

'core'

verbose_name

'Snapshot'

verbose_name_plural

'Snapshots'

indexes: ClassVar[list[django.db.models.Index]]

None

constraints: ClassVar[list[django.db.models.BaseConstraint]]

None

__str__()

output_paths_for_delete() → tuple[pathlib.Path, ...]

delete(*using=None*, *keep_parents=False*)

classmethod delete_requested(*, *batch_size=100*)

classmethod crawl_count_subquery(*, *status: str | None = None*, *outer_ref: str = 'pk'*) → django.db.models.QuerySet

Return a scalar subquery counting Snapshots for one outer Crawl.

classmethod crawl_count_expr(*, *status: str | None = None*, *outer_ref: str = 'pk'*)

classmethod crawl_total_and_status_counts(*crawl_ids: collections.abc.Iterable[Any]*, *, *status: str*) → dict[str, dict[str, int]]

Return total and status-filtered Snapshot counts keyed by Crawl ID.

update_and_requeue(***kwargs*) → bool

Update this Snapshot through the shared retry_at ownership path.

Any non-final Snapshot work means the parent Crawl must also be visible to the runner. Keep that invariant here so CLI/admin callers do not hand-edit the parent Crawl state every time they retry a hook.

queue_for_extraction(*, *when=None*) → bool

Queue this Snapshot for the runner using the normal state path.

schedule_plugin_run(*plugins: collections.abc.Iterable[str]*, *, *when=None*) → bool

Persist one snapshot-scoped plugin request until abx-dl completes it.

pause(*, *save: bool = True*) → bool

resume(*, *when: datetime.datetime | None = None*, *save: bool = True*) → bool

restore_paused_scheduler_marker() → None

Restore the indefinite scheduler marker owned by the PAUSED lifecycle.

reconcile_parent_lifecycle(*, *lock_seconds: int = 60*) → bool | None

Follow parent Crawl pause/seal state before any Snapshot work runs.

Crawl.pause()/cancel() only wake child rows. The runner claims each due Snapshot and lets this method perform the actual child transition, so cancellation stays fast and Snapshot cleanup still runs from the normal lifecycle owner.

finalize_completed_upload_results() → int

start_processing() → bool

Atomically move a claimed queued Snapshot into its active lease.

seal() → bool

Atomically finalize this Snapshot and reconcile its output metadata.

advance_lifecycle() → bool

Advance one explicit lifecycle step after the runner claims this row.

cancel() → None

get_delete_after_config_value()

classmethod missing_delete_at_candidates()

classmethod is_archivebox_internal_url(*url: str*, *, *config: collections.abc.Mapping[str, Any] | Any | None = None*) → bool

property created_by

Convenience property to access the user who created this snapshot via its crawl.

property process_set

Get all Process objects related to this snapshot's ArchiveResults.

property binary_set

Get all Binary objects used by processes related to this snapshot.

ensure_permissions_config(*crawl_permissions: str | None = None*) → bool

validate_url_for_archiving(*, *config: collections.abc.Mapping[str, Any] | Any | None = None*) → None

save(**args*, ***kwargs*)

static _fs_current_version() → str

Get current ArchiveBox filesystem layout version.

_FS_VERSION_MIGRATION_PATHS: ClassVar[dict[str, str]]

None

property fs_migration_needed: bool

Check if snapshot needs filesystem migration

_fs_next_version(*version: str*) → str

Get the next declared version in the filesystem migration chain.

static is_legacy_archive_dir(*path: pathlib.Path*) → bool

Return True for old-style archive/{timestamp} snapshot directories.

migrate_filesystem_to_current_version(*source_dir: pathlib.Path | None = None*, *config:* ArchiveBoxBaseConfig | *None = None*) → None

Copy legacy snapshot output into the current layout and safely remove the old tree.

The ordering is intentionally crash-safe:

1. Copy from the legacy directory into the new directory idempotently.
2. Verify the new directory has every old file.
3. Convert metadata in the new directory.
4. Remove the verified legacy source.
5. Persist fs_version last, so an interruption remains selected by the indexed stale-version query and resumes naturally.

Re-running this method also reconciles a legacy timestamp directory left behind when a machine stopped after the database commit but before the on-commit cleanup callback ran.

_fs_migrate_from_0_7_0_to_0_9_0(*source_dir: pathlib.Path | None = None*, *config:* ArchiveBoxBaseConfig | *None = None*)

_fs_migrate_from_0_8_0_to_0_9_0(*source_dir: pathlib.Path | None = None*, *config:* ArchiveBoxBaseConfig | *None = None*)

_fs_migrate_from_0_9_0_to_0_9_4(*source_dir: pathlib.Path | None = None*, *config:* ArchiveBoxBaseConfig | *None = None*)

hydrate_archiveresult_output_metadata(*snapshot_dir: pathlib.Path | None = None*) → int

Populate missing ArchiveResult file metadata from existing outputs.

_fs_migrate_legacy_to_0_9_0(*source_dir: pathlib.Path | None = None*, *target_dir: pathlib.Path | None = None*, *config:* ArchiveBoxBaseConfig | *None = None*)

Migrate from flat to nested structure.

0.8.x: archive/{timestamp}/ 0.9.x: archive/users/{user}/snapshots/YYYYMMDD/{domain}/{uuid}/

static _migration_trees_match(*old_dir: pathlib.Path*, *new_dir: pathlib.Path*) → bool

Verify every legacy entry exists unchanged at the destination.

_cleanup_old_migration_dir(*old_dir: pathlib.Path*, *new_dir: pathlib.Path*) → bool

Delete the old directory after its contents are verified at the new path.

static extract_domain_from_url(*url: str*) → str

Extract domain from URL for 0.9.x path structure. Uses full hostname with sanitized special chars.

Examples: https://example.com:5797 → example.com_5797 https://sub.example.com → sub.example.com file:///path → localhost data:text/html → data

get_storage_path_for_version(*version: str*) → pathlib.Path

Calculate storage path for specific filesystem version. Centralizes path logic so it's reusable.

0.7.x/0.8.x: archive/{timestamp} 0.9.x: archive/users/{username}/snapshots/YYYYMMDD/{domain}/{uuid}/

classmethod load_from_directory(*snapshot_dir: pathlib.Path*) → Optional[*archivebox.core.models.Snapshot*]

Load existing Snapshot from DB by reading index.jsonl or index.json.

Reads index file, extracts url+timestamp, queries DB. Returns existing Snapshot or None if not found/invalid. Does NOT create new snapshots.

ONLY used by: archivebox update (for orphan detection)

classmethod create_from_directory(*snapshot_dir: pathlib.Path*, *, *crawl=None*, *preserve_identity: bool = False*) → Optional[*archivebox.core.models.Snapshot*]

Create new Snapshot from orphaned directory.

Validates timestamp, ensures uniqueness. Returns new UNSAVED Snapshot or None if invalid.

ONLY used by: archivebox update (for orphan import)

static _select_best_timestamp(*index_timestamp: object | None*, *folder_name: str*) → str | None

Select best timestamp from index.json vs folder name.

Validates range (1995-2035). When a valid legacy folder name is available it is the stable filesystem identity, so preserve it over normalized variants like "1508259732.0" found in old index files.

classmethod _ensure_unique_timestamp(*url: str*, *timestamp: str*) → str

Ensure timestamp is globally unique. If there is a collision, add a tiny fractional suffix until unique.

static _detect_fs_version_from_index(*data: dict*) → str

Detect fs_version from index.json structure.

- Has fs_version field: use it
- Has history dict: 0.7.0
- Has archive_results list: 0.8.0
- Default: 0.7.0

reconcile_with_index(*output_dir: pathlib.Path | None = None*, *update_existing_archive_results: bool = True*)

Merge index.json/index.jsonl with DB. DB is source of truth.

- Title: longest non-URL
- Tags: union
- ArchiveResults: keep both (by plugin+start_ts)

Converts index.json to index.jsonl if needed, then writes back in JSONL format.

Used by: archivebox update (to sync index with DB)

reconcile_with_index_json(*output_dir: pathlib.Path | None = None*, *update_existing_archive_results: bool = True*)

Deprecated: use reconcile_with_index() instead.

_merge_title_from_index(*index_data: dict*)

Merge title - prefer longest non-URL title.

_merge_tags_from_index(*index_data: dict*)

Merge tags - union of both sources.

_merge_archive_results_from_index(*index_data: dict*, *update_existing: bool = True*)

Merge ArchiveResults one row per hook; retries update the existing row.

_create_archive_result_if_missing(*result_data: dict*, *existing: dict*, *update_existing: bool = True*, *preserve_identity: bool = False*)

Create ArchiveResult if not already in DB.

write_index_json()

Write index.json in 0.9.x format (deprecated, use write_index_jsonl).

write_index_jsonl(*output_dir: pathlib.Path | None = None*)

Write index.jsonl in flat JSONL format.

Each line is a JSON record with a 'type' field:

- Snapshot: snapshot metadata (crawl_id, url, tags, etc.)
- ArchiveResult: extractor results (plugin, status, output, etc.)
- Binary: binary info used for the extraction
- Process: process execution details (cmd, exit_code, timing, etc.)

read_index_jsonl(*output_dir: pathlib.Path | None = None*) → dict

Read index.jsonl and return parsed records grouped by type.

Returns dict with keys: 'snapshot', 'archive_results', 'binaries', 'processes'

convert_index_json_to_jsonl(*output_dir: pathlib.Path | None = None*) → bool

Convert index.json to index.jsonl format.

Reads existing index.json and creates index.jsonl while preserving the original JSON byte-for-byte for unknown legacy metadata. Returns True if conversion was performed, False if no conversion needed.

static move_directory_to_invalid(*snapshot_dir: pathlib.Path*)

Move invalid directory to data/invalid/YYYYMMDD/.

Used by: archivebox update (when encountering invalid directories)

classmethod find_and_merge_duplicates() → int

Find and merge snapshots with same url:timestamp. Returns count of duplicate sets merged.

Used by: archivebox update (Phase 3: deduplication)

classmethod _merge_snapshots(*snapshots: collections.abc.Sequence[*archivebox.core.models.Snapshot*]*)

Merge exact duplicates. Keep oldest, union files + ArchiveResults.

property output_dir_parent: str

property output_dir_name: str

archive(*overwrite=False*, *methods=None*)

tags_str() → str | None

plugin_output_path(*plugin: str*, *archive_results=None*) → str | None

Return the first declared successful output path for one plugin.

icons(*path: str | None = None*, *prefix: str = '/'*, *quote_paths: bool = False*) → str

Generate HTML icons showing which extractor plugins have succeeded for this snapshot

property api_url: str

get_absolute_url()

domain() → str

property title_stripped: str

static _normalize_title_candidate(*candidate: str | None*, *, *snapshot_url: str*) → str

property resolved_title: str

hashes_index() → dict[str, dict[str, Any]]

property output_dir: pathlib.Path

The filesystem path to the snapshot's output directory.

ensure_crawl_symlink(*, *crawl_dir: pathlib.Path | None = None*, *snapshot_dir: pathlib.Path | None = None*) → None

Ensure snapshot is symlinked under its crawl output directory.

remove_legacy_archive_symlink() → None

Remove a stale archive/<timestamp> compatibility projection.

reconcile_filesystem_links() → None

Repair filesystem projections after a save or migration.

legacy_archive_path() → str

classmethod archive_path_for_values(*, *fs_version: str*, *timestamp: str*, *snapshot_id*, *url: str*, *username: str = 'web'*, *bookmarked_at=None*, *created_at=None*) → str

Build the public detail path without consulting the filesystem.

archive_path_from_db() → str

Best-effort public URL path derived from DB fields only.

url_path() → str

URL path matching the current snapshot output_dir layout.

archive_path()

archive_size()

save_tags(*tags: collections.abc.Iterable[str] = ()*, *, *created_by: Any = None*) → None

finalize_output_metadata() → None

Clean up background ArchiveResult hooks and empty results.

Called after entering the sealed state. Reconcile late background outputs and hydrate result metadata.

to_json() → dict

Convert Snapshot model instance to a JSON-serializable dict. Includes all fields needed to fully reconstruct/identify this snapshot.

static from_json(*record: dict[str, Any]*, *overrides: dict[str, Any] | None = None*, *queue_for_extraction: bool = True*)

Create/update Snapshot from JSON dict.

Unified method that handles:

- ID-based patching: {"id": "...", "title": "new title"}
- URL-based create/update: {"url": "...", "title": "...", "tags": "..."}
- Auto-creates Crawl if not provided
- Optionally queues for extraction

Args: record: Dict with 'url' (for create) or 'id' (for patch), plus other fields overrides: Dict with 'crawl', 'snapshot' (parent), 'created_by_id' queue_for_extraction: If True, sets status=QUEUED and retry_at (default: True)

Returns: Snapshot instance or None

get_progress_stats() → dict

Get progress statistics for this snapshot's archiving process.

Returns dict with: - total: Total number of archive results - succeeded: Number of succeeded results - failed: Number of failed results - running: Number of currently running results - pending: Number of pending/queued results - percent: Completion percentage (0-100) - output_size: Total output size in bytes - is_sealed: Whether the snapshot is in a final state

retry_failed_archiveresults() → int

Queue the parent Snapshot to rerun plugins with failed facts.

url_hash() → str

scheme() → str

path() → str

basename() → str

extension() → str

base_url() → str

is_static() → bool

is_archived() → bool

bookmarked_date() → str | None

downloaded_datestr() → str | None

archive_dates() → list[datetime.datetime]

oldest_archive_date() → datetime.datetime | None

newest_archive_date() → datetime.datetime | None

num_outputs() → int

num_failures() → int

latest_outputs(*status: str | None = None*) → dict[str, Any]

Get the latest output that each plugin produced

discover_outputs(*include_filesystem_fallback: bool = True*, *archive_results: list[archivebox.core.models.Snapshot.discover_outputs.ArchiveResult] | None = None*, *filesystem_index: dict | None = None*) → list[dict]

Discover output paths; file availability is not result success.

Live cards must take succeeded/failed state from ArchiveResult.status. A failed hook can leave useful partial files, and hooks/retries share plugin directories, so an embeddable path or nonzero size is not proof of success. Filesystem discovery also supports portable static exports; it must not overwrite or replace the live database's lifecycle facts.

This includes partial files for browsing. get_html_details_context separately admits successful DB results to the output cards.

property static_archive_path: str

Snapshot output path relative to the data root, for portable exports.

to_dict(*extended: bool = False*, *static_export: bool = False*) → dict[str, Any]

Convert Snapshot to a dictionary (replacement for Link._asdict())

to_json_str(*indent: int = 4*) → str

Convert to JSON string (legacy method, use to_json() for dict)

to_csv(*cols: list[str] | None = None*, *separator: str = ','*, *ljust: int = 0*) → str

Convert to CSV string

write_json_details(*out_dir: pathlib.Path | str | None = None*) → None

Write JSON index file for this snapshot to its output directory

get_html_details_context(*request=None*, ***, *static_export_dir: pathlib.Path | None = None*, *discover_files: bool = False*) → dict[str, Any]

Build the one context used by both served and on-disk snapshot pages.

write_html_details(*out_dir: pathlib.Path | str | None = None*) → None

Write the unified snapshot detail page with portable filesystem URLs.

get_detail_page_auxiliary_items(*outputs: list[dict] | None = None*, *archive_results: list[archivebox.core.models.Snapshot.get_detail_page_auxiliary_items.ArchiveResult] | None = None*, *hashes_index: dict[str, dict[str, Any]] | None = None*) → tuple[list[dict[str, object]], list[dict[str, object]]]

static _ts_to_date_str(*dt: datetime.datetime | None*) → str | None

class archivebox.core.models.**ArchiveResult**

Bases: *archivebox.base_models.models.ModelWithDeleteAfter*, *archivebox.base_models.models.ModelWithOutputDir*, *archivebox.base_models.models.ModelWithNotes*

objects

'as_manager(...)'

class StatusChoices

Bases: django.db.models.TextChoices

QUEUED

('queued', 'Queued')

STARTED

('started', 'Started')

PAUSED

('paused', 'Paused')

BACKOFF

('backoff', 'Waiting to retry')

SUCCEEDED

('succeeded', 'Succeeded')

FAILED

('failed', 'Failed')

SKIPPED

('skipped', 'Skipped')

NORESULTS

('noresults', 'No Results')

INITIAL_STATE

None

ACTIVE_STATE

None

FINAL_STATES

()

FINAL_OR_ACTIVE_STATES

()

delete_after_final_statuses

None

classmethod normalize_status(*status: str | None*) → str

classmethod get_or_create_by_hook(*snapshot:* archivebox.core.models.Snapshot, *plugin: str*, *hook_name: str*, *, *defaults: collections.abc.Mapping[str, Any] | None = None*) → tuple[*archivebox.core.models.ArchiveResult*, bool]

static output_files_upload_complete(*output_files: dict[str, dict[str, Any]]*) → bool

classmethod get_plugin_choices()

Get plugin choices from discovered hooks (for forms/admin).

classmethod snapshot_count_subquery(*, *status: str | None = None*, *outer_ref: str = 'pk'*) → django.db.models.QuerySet

Return a scalar subquery counting ArchiveResults for one outer Snapshot.

Use this instead of filtered join aggregates for per-row Snapshot counts: the scalar form lets SQLite probe the covering `(snapshot_id, status)` or `(status, snapshot_id)` indexes once per visible Snapshot row, instead of joining and grouping the whole candidate Snapshot queryset.

classmethod snapshot_half_count_subquery(*, *outer_ref: str = 'snapshot_id'*) → django.db.models.QuerySet

classmethod snapshot_count_expr(*, *status: str | None = None*, *outer_ref: str = 'pk'*)

classmethod status_counts(*queryset: django.db.models.QuerySet | None = None*, *statuses: collections.abc.Iterable[str] | None = None*) → dict[str, int]

Count requested statuses with separate indexed COUNT probes.

classmethod snapshot_ids_with_majority_status(*status: str | collections.abc.Iterable[str]*) → django.db.models.QuerySet

Return Snapshot IDs where more than half of ArchiveResults have `status`.

Start from ArchiveResult.status for every majority-status filter. The `(status, snapshot_id)` index keeps the plan predictable even when a user's collection has an unusual status distribution.

id

'CompactUUIDField(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

snapshot: *archivebox.core.models.Snapshot*

'ForeignKey(...)'

plugin

'CharField(...)'

hook_name

'CharField(...)'

process

'OneToOneField(...)'

output_str

'TextField(...)'

output_json

'JSONField(...)'

output_files

'JSONField(...)'

output_size

'BigIntegerField(...)'

output_mimetypes

'CharField(...)'

start_ts

'DateTimeField(...)'

end_ts

'DateTimeField(...)'

status

'CharField(...)'

retry_at

'DateTimeField(...)'

notes

'TextField(...)'

snapshot_id: uuid.UUID

None

process_id: uuid.UUID | None

None

class Meta

Bases: *archivebox.base_models.models.ModelWithDeleteAfter.Meta*, *archivebox.base_models.models.ModelWithOutputDir.Meta*, *archivebox.base_models.models.ModelWithNotes.Meta*

app_label

'core'

verbose_name

'Archive Result'

verbose_name_plural

'Archive Results'

indexes: ClassVar[list[django.db.models.Index]]

None

constraints: ClassVar[list[django.db.models.BaseConstraint]]

None

__str__()

static _format_output_line_for_display(*line: str*) → str

output_str_for_display() → str

get_delete_after_config_value()

classmethod missing_delete_at_candidates()

property created_by

Convenience property to access the user who created this archive result via its snapshot's crawl.

to_json(*, *snapshot_output_dir: pathlib.Path | None = None*) → dict

Convert ArchiveResult model instance to a JSON-serializable dict.

static from_json(*record: dict[str, Any]*, *overrides: dict[str, Any] | None = None*)

Create/update ArchiveResult from JSON dict.

Args: record: JSON dict with 'snapshot_id', 'plugin', etc. overrides: Optional dict of field overrides

Returns: ArchiveResult instance or None

save(**args*, ***kwargs*)

safe_update(*update_fields: collections.abc.Mapping[str, Any]*, *, *refresh: bool = True*) → bool

Compare-and-swap one loaded ArchiveResult without opening a transaction.

schedule_delete_cleanup(*, *using: str | None = None*) → None

Remove shared plugin output and refresh persisted Snapshot metadata after commit.

static refresh_snapshot_output_sizes(*snapshot_ids*)

snapshot_dir()

url()

property api_url: str

get_absolute_url()

property is_paused: bool

static _normalize_output_files(*raw_output_files: Any*) → dict[str, dict[str, Any]]

static _coerce_output_file_size(*value: Any*) → int

output_file_map() → dict[str, dict[str, Any]]

output_file_paths() → list[str]

output_file_path(*path: str*, *output_file_map: dict[str, dict[str, Any]] | None = None*) → str | None

Resolve an output_files key to its snapshot-relative path.

update_output_metadata_from_filesystem(*snapshot_dir: pathlib.Path | None = None*, *save: bool = True*, *full_scan: bool = False*) → bool

static _looks_like_output_path(*raw_output: str | None*, *plugin_name: str | None = None*) → bool

_existing_output_path(*raw_output: str | None*) → str | None

static _fallback_output_file_path(*output_file_paths: collections.abc.Sequence[str]*, *plugin_name: str | None = None*, *output_file_map: dict[str, dict[str, Any]] | None = None*) → str | None

static _scan_output_file_map(*dir_path: pathlib.Path*, *, *max_scan: int | None = None*) → dict[str, dict[str, Any]]

Collect read-only metadata for the existing output preview selector.

static _find_best_output_file(*dir_path: pathlib.Path*, *plugin_name: str | None = None*) → pathlib.Path | None

embed_path_db(*output_file_map: dict[str, dict[str, Any]] | None = None*) → str | None

embed_path() → str | None

Get the relative path to the embeddable output file for this result.

This is intentionally DB-backed only so snapshot/admin rendering stays fast and predictable without filesystem probes.

output_folder_path() → str

Return the physical folder for this result's files, using saved metadata.

property output_dir_name: str

property output_dir_parent: str

property process_record

property pwd: str

Working directory metadata without probing archive storage.

property cmd: list

Command array (from Process).

property cmd_version: str

Command version (from Process.binary).

property binary

Binary FK (from Process).

property iface

Network interface FK (from Process).

property machine

Machine FK (from Process).

property timeout: int

Timeout in seconds (from Process).

_url_passes_filters(*url: str*) → bool

Check if URL passes URL_ALLOWLIST and URL_DENYLIST config filters.

Uses the centralized config resolver so frozen crawl/snapshot values and live Machine/Persona execution values apply in their scoped order.

property output_dir: pathlib.Path

Get the output directory for this plugin's results.

archivebox.core.middleware

Module Contents

Classes

ReverseProxyAuthMiddleware

Functions

_admin_login_hint_cookie_domain	Resolve the parent domain to scope the cross-subdomain login hint.
detect_timezone	
TimezoneMiddleware	
AdminCookieIsolationMiddleware	
CacheControlMiddleware	
ServerSecurityModeMiddleware	
HostRoutingMiddleware	

Data

ADMIN_LOGIN_HINT_COOKIE

API

archivebox.core.middleware.**ADMIN_LOGIN_HINT_COOKIE**

'archivebox_admin_logged_in'

archivebox.core.middleware.**_admin_login_hint_cookie_domain**(*config*) → str | None

Resolve the parent domain to scope the cross-subdomain login hint.

NOTE: this cookie carries only the single bit "a superuser is logged in on admin somewhere"; it MUST NOT be confused with the session cookie, which stays admin-host-scoped (see core/settings.py SESSION_COOKIE_DOMAIN comment — admin/web is a security boundary).

Returns the hostname portion of `get_base_host` (which respects `BASE_URL` and falls back to the local-bind mapping). Strips the port — cookie `Domain=` attributes don't include ports. Returns `None` when subdomain routing is off, the base host is empty, or the base host is an IP / bare `localhost` (browsers reject cross-host cookies for those).

archivebox.core.middleware.**detect_timezone**(*request*, *activate: bool = True*)

archivebox.core.middleware.**TimezoneMiddleware**(*get_response*)

archivebox.core.middleware.**AdminCookieIsolationMiddleware**(*get_response*)

archivebox.core.middleware.**CacheControlMiddleware**(*get_response*)

archivebox.core.middleware.**ServerSecurityModeMiddleware**(*get_response*)

archivebox.core.middleware.**HostRoutingMiddleware**(*get_response*)

class archivebox.core.middleware.**ReverseProxyAuthMiddleware**

Bases: `django.contrib.auth.middleware.RemoteUserMiddleware`

header

'HTTP_REMOTE_USER'

process_request(*request*)

archivebox.core.tag_util

Module Contents

Functions

normalize_tag_name
normalize_tag_sort
normalize_has_snapshots_filter
normalize_created_by_filter
normalize_created_year_filter
get_matching_tags
add_snapshot_counts

continues on next page

Table 181 – continued from previous page

get_tag_creator_choices
get_tag_year_choices
get_tag_by_ref
get_or_create_tag
rename_tag
delete_tag
export_tag_urls
export_tag_snapshots_jsonl
_display_snapshot_title
_build_snapshot_preview
_build_snapshot_preview_map
build_tag_card
build_tag_cards

Data

TAG_SNAPSHOT_PREVIEW_LIMIT
TAG_SORT_CHOICES
TAG_HAS_SNAPSHOTS_CHOICES

API

archivebox.core.tag_util.**TAG_SNAPSHOT_PREVIEW_LIMIT**

10

archivebox.core.tag_util.**TAG_SORT_CHOICES**

(('name_asc', 'Name A-Z'), ('name_desc', 'Name Z-A'), ('created_desc', 'Created newest'), ('created_…

archivebox.core.tag_util.**TAG_HAS_SNAPSHOTS_CHOICES**

(('all', 'All'), ('yes', 'Has snapshots'), ('no', 'No snapshots'))

archivebox.core.tag_util.**normalize_tag_name**(*name: str*) → str

archivebox.core.tag_util.**normalize_tag_sort**(*sort: str = 'created_desc'*) → str

archivebox.core.tag_util.**normalize_has_snapshots_filter**(*value: str = 'all'*) → str

archivebox.core.tag_util.**normalize_created_by_filter**(*created_by: str = ''*) → str

archivebox.core.tag_util.**normalize_created_year_filter**(*year: str = ''*) → str

archivebox.core.tag_util.**get_matching_tags**(*query: str = ''*, *sort: str = 'created_desc'*, *created_by: str = ''*, *year: str = ''*, *has_snapshots: str = 'all'*) → django.db.models.QuerySet[*archivebox.core.models.Tag*]

archivebox.core.tag_util.**add_snapshot_counts**(*tags: list[*archivebox.core.models.Tag*]*, *snapshot_queryset: django.db.models.QuerySet[*archivebox.core.models.Snapshot*] | None = None*) → None

archivebox.core.tag_util.**get_tag_creator_choices**() → list[tuple[str, str]]

archivebox.core.tag_util.**get_tag_year_choices**() → list[str]

archivebox.core.tag_util.**get_tag_by_ref**(*tag_ref: str | int*) → *archivebox.core.models.Tag*

archivebox.core.tag_util.**get_or_create_tag**(*name: str*, *created_by: django.contrib.auth.models.User | None = None*) → tuple[*archivebox.core.models.Tag*, bool]

archivebox.core.tag_util.**rename_tag**(*tag:* archivebox.core.models.Tag, *name: str*) → *archivebox.core.models.Tag*

archivebox.core.tag_util.**delete_tag**(*tag:* archivebox.core.models.Tag) → tuple[int, dict[str, int]]

archivebox.core.tag_util.**export_tag_urls**(*tag:* archivebox.core.models.Tag) → str

archivebox.core.tag_util.**export_tag_snapshots_jsonl**(*tag:* archivebox.core.models.Tag) → str

archivebox.core.tag_util.**_display_snapshot_title**(*snapshot:* archivebox.core.models.Snapshot) → str

archivebox.core.tag_util.**_build_snapshot_preview**(*snapshot:* archivebox.core.models.Snapshot, *request: django.http.HttpRequest | None = None*, *config: Any | None = None*) → dict[str, Any]

archivebox.core.tag_util.**_build_snapshot_preview_map**(*tags: list[*archivebox.core.models.Tag*]*, *request: django.http.HttpRequest | None = None*, *preview_limit: int = TAG_SNAPSHOT_PREVIEW_LIMIT*) → dict[int, list[dict[str, Any]]]

archivebox.core.tag_util.**build_tag_card**(*tag:* archivebox.core.models.Tag, *snapshot_previews: list[dict[str, Any]] | None = None*) → dict[str, Any]

archivebox.core.tag_util.**build_tag_cards**(*query: str = ''*, *request: django.http.HttpRequest | None = None*, *limit: int | None = None*, *preview_limit: int = TAG_SNAPSHOT_PREVIEW_LIMIT*, *sort: str = 'created_desc'*, *created_by: str = ''*, *year: str = ''*, *has_snapshots: str = 'all'*) → list[dict[str, Any]]

archivebox.core.admin_snapshots

Module Contents

Classes

SnapshotActionForm	
TagNameListFilter	
SnapshotPermissionsListFilter	
SnapshotStatusListFilter	
SnapshotArchiveStateListFilter	
SnapshotSizeListFilter	
SnapshotResultHealthListFilter	
SnapshotChangeList	
SnapshotAdminForm	Custom form for Snapshot admin with tag editor widget.
SnapshotAdmin	

Functions

_format_size_column

Data

GLOBAL_CONTEXT
SNAPSHOT_PERMISSION_META

API

archivebox.core.admin_snapshots.**GLOBAL_CONTEXT**

None

archivebox.core.admin_snapshots.**SNAPSHOT_PERMISSION_META**

None

archivebox.core.admin_snapshots.**_format_size_column**(*num_bytes: int*) → str

class archivebox.core.admin_snapshots.**SnapshotActionForm**(**args*, ***kwargs*)

Bases: django.contrib.admin.helpers.ActionForm

Initialization

clean_tags()

Parse comma-separated tag names without touching the DB.

class archivebox.core.admin_snapshots.**TagNameListFilter**

Bases: django.contrib.admin.SimpleListFilter

title

'By tag name'

parameter_name

'tag'

lookups(*request*, *model_admin*)

queryset(*request*, *queryset*)

class archivebox.core.admin_snapshots.**SnapshotPermissionsListFilter**

Bases: django.contrib.admin.SimpleListFilter

title

'permission'

parameter_name

'permissions'

lookups(*request*, *model_admin*)

queryset(*request*, *queryset*)

class archivebox.core.admin_snapshots.SnapshotStatusListFilter

Bases: django.contrib.admin.SimpleListFilter

title

'snapshot status'

parameter_name

'snapshot_status'

lookups(*request*, *model_admin*)

queryset(*request*, *queryset*)

class archivebox.core.admin_snapshots.SnapshotArchiveStateListFilter

Bases: django.contrib.admin.SimpleListFilter

title

'archive state'

parameter_name

'archive_state'

lookups(*request*, *model_admin*)

queryset(*request*, *queryset*)

class archivebox.core.admin_snapshots.SnapshotSizeListFilter

Bases: django.contrib.admin.SimpleListFilter

title

'size'

parameter_name

'size'

lookups(*request*, *model_admin*)

queryset(*request*, *queryset*)

class archivebox.core.admin_snapshots.SnapshotResultHealthListFilter

Bases: django.contrib.admin.SimpleListFilter

title

'ArchiveResult status'

parameter_name

'archiveresult_status'

SNAPSHOT_FIRST_VALUES

None

lookups(*request*, *model_admin*)

queryset(*request*, *queryset*)

class archivebox.core.admin_snapshots.SnapshotChangeList(*request*, **args*, ***kwargs*)

Bases: *archivebox.search.admin.SearchResultsChangeList*

_attach_archiveresult_summaries()

get_results(*request*)

class archivebox.core.admin_snapshots.**SnapshotAdminForm**(**args*, ***kwargs*)

Bases: django.forms.ModelForm

Custom form for Snapshot admin with tag editor widget.

Initialization

url

'CharField(…)'

tags_editor

'CharField(…)'

permissions_config

'ChoiceField(…)'

class Meta

model

None

fields

'**all**'

save(*commit=True*)

class archivebox.core.admin_snapshots.**SnapshotAdmin**

Bases: *archivebox.search.admin.SearchResultsAdminMixin*, *archivebox.base_models.admin.ConfigEditorMixin*, *archivebox.base_models.admin.BaseModelAdmin*

form

None

raw_id_fields

('crawl', 'parent_snapshot')

list_select_related

()

save_related(*request*, *form*, *formsets*, *change*)

list_display

('permissions_badge', 'created_at_display', 'preview_icon', 'title_str', 'tags_inline', 'status_with…

list_display_links

('created_at_display',)

sort_fields

('title_str', 'created_at', 'status', 'crawl')

readonly_fields

('admin_actions', 'snapshot_summary', 'url_favicon', 'tags_badges', 'imported_timestamp', 'created_a…

search_fields

('id', 'url', 'timestamp', 'title', 'tags__name')

list_filter

()

fieldsets

((‘Actions’,), (‘Snapshot’,), (‘URL’,), (‘Tags’,), (‘Status’,), (‘Timestamps’,), (‘Relations’,), (‘C…

ordering

[‘-created_at’]

actions

[‘add_tags’, ‘remove_tags’, ‘resnapshot_snapshot’, ‘update_snapshots’, ‘overwrite_snapshots’, ‘set_s…

inlines

[]

list_per_page

50

created_at_display(*obj*)

action_form

None

paginator

None

save_on_top

True

show_full_result_count

True

get_changelist(*request*, ***kwargs*)

get_ordering(*request*)

render_change_form(*request*, *context*, *add=False*, *change=False*, *form_url=''*, *obj=None*)

changelist_view(*request*, *extra_context=None*)

get_actions(*request*)

lookup_allowed(*lookup*, *value*, *request=None*)

get_snapshot_view_url(*obj:* archivebox.core.models.Snapshot) → str

get_snapshot_files_url(*obj:* archivebox.core.models.Snapshot) → str

get_snapshot_zip_url(*obj:* archivebox.core.models.Snapshot) → str

get_urls()

preview_view(*request*, *object_id*, *plugin*, *filename*)

Serve a saved image output from the authenticated admin origin.

search_stream_view(*request*)

set_permissions_view(*request*, *object_id*)

set_snapshot_permissions(*request*, *queryset*)

update_snapshot_permissions(*queryset*, *permissions*)

redo_failed_view(*request*, *object_id*)

get_queryset(*request*)

permissions_badge(*obj*)

imported_timestamp(*obj*)

admin_actions(*obj*)

status_info(*obj*)

archiveresults_list(*obj*)

title_str(*obj*)

tags_inline(*obj*)

tags_badges(*obj*)

_get_favicon_data(*obj*)

_render_thumbnail(*obj*, *width='100px'*, *height='100px'*)

url_favicon(*obj*)

preview_icon(*obj*)

snapshot_summary(*obj*)

files(*obj*)

size(*obj*)

status_with_progress(*obj*)
Show status with progress bar for in-progress snapshots.

size_with_stats(*obj*)
Show archive size with output size from archive results.

_get_progress_stats(*obj*)

_get_prefetched_results(*obj*)

_get_expected_hook_total(*obj*) → int

_get_prefetched_tags(*obj*)

_get_ordering_fields(*request*)

url_str(*obj*)

health_display(*obj*)

grid_view(*request*, *extra_context=None*)

update_snapshots(*request*, *queryset*)

resnapshot_snapshot(*request*, *queryset*)

overwrite_snapshots(*request*, *queryset*)

delete_snapshots(*request*, *queryset*)
Keep the rows until the runner removes and verifies their files.

add_tags(*request*, *queryset*)

remove_tags(*request*, *queryset*)

archivebox.core.wsgi

WSGI config for archivebox project.

It exposes the WSGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/2.1/howto/deployment/wsgi/

Module Contents

Data

application

API

archivebox.core.wsgi.**application**
'get_wsgi_application(...)'

archivebox.core.admin_users

Module Contents

Classes

CustomUserAdmin

Functions

register_admin

API

class archivebox.core.admin_users.**CustomUserAdmin**
Bases: `django.contrib.auth.admin.UserAdmin`

sort_fields
['id', 'email', 'username', 'is_superuser', 'last_login', 'date_joined']

list_display
['username', 'id', 'email', 'is_superuser', 'last_login', 'date_joined']

readonly_fields

('snapshot_set', 'archiveresult_set', 'tag_set', 'apitoken_set', 'outboundwebhook_set')

change_form_template

'admin/auth/user/change_form.html'

add_fieldsets

None

fieldsets

None

get_queryset(*request*)

snapshot_rss_badge(*obj*, *api_token: str = ''*)

snapshot_count_badge(*obj*)

snapshot_count_column(*obj*)

get_list_display(*request*)

snapshot_set(*obj*)

archiveresult_set(*obj*)

tag_set(*obj*)

apitoken_set(*obj*)

outboundwebhook_set(*obj*)

archivebox.core.admin_users.**register_admin**(*admin_site*)

archivebox.core.asgi

ASGI config for archivebox project.

It exposes the ASGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/stable/howto/deployment/asgi/

Module Contents

Classes

ArchiveBoxASGIHandler

Functions

application

Data

django_application

API

class archivebox.core.asgi.**ArchiveBoxASGIHandler**

Bases: django.core.handlers.asgi.ASGIHandler

create_request(*scope*, *body_file*)

archivebox.core.asgi.**django_application**

'ArchiveBoxASGIHandler(...)'

async archivebox.core.asgi.**application**(*scope*, *receive*, *send*)

archivebox.core.admin

Module Contents

Functions

register_admin

API

archivebox.core.admin.**register_admin**(*admin_site*)

Package Contents

Functions

register_admin	Register the core.models views (Snapshot, ArchiveResult, Tag, etc.) with the admin site

Data

__order__

API

archivebox.core.**__order__**

100

archivebox.core.**register_admin**(*admin_site*)

Register the core.models views (Snapshot, ArchiveResult, Tag, etc.) with the admin site

archivebox.search

Submodules

archivebox.search.sonic_daemon

Module Contents

Functions

register_sonic_daemon_event_handler	

API

archivebox.search.sonic_daemon.**register_sonic_daemon_event_handler**(*bus*) → None

archivebox.search.apps

Module Contents

Classes

SearchConfig	Register search templates and admin integration with Django.

API

class archivebox.search.apps.**SearchConfig**

Bases: django.apps.AppConfig

Register search templates and admin integration with Django.

default_auto_field

'django.db.models.BigAutoField'

name

'archivebox.search'

verbose_name

'Search'

archivebox.search.backends

Module Contents

Functions

search_backend_command_env	Serialize resolved application config for a standalone plugin command.
normalize_search_backend_name	Normalize a backend name for config and plugin lookup.

continues on next page

Table 197 – continued from previous page

get_available_backends	Discover search-capable plugins and cache their catalog entries.
get_backend	Resolve the configured search-capable plugin.

Data

_search_backends_cache

API

archivebox.search.backends.**_search_backends_cache: dict | None**

None

archivebox.search.backends.**search_backend_command_env**(*config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → dict[str, str]

Serialize resolved application config for a standalone plugin command.

archivebox.search.backends.**normalize_search_backend_name**(*backend_name: str | None*) → str

Normalize a backend name for config and plugin lookup.

archivebox.search.backends.**get_available_backends**() → dict

Discover search-capable plugins and cache their catalog entries.

archivebox.search.backends.**get_backend**(*config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → Any

Resolve the configured search-capable plugin.

archivebox.search.views

Module Contents

Functions

get_admin_search_cache_key	Build the cache key for one user and changelist URL.
get_public_search_cache_key	Build the cache key for one public search URL.
get_cached_admin_search_ids	Return streamed admin search IDs from Django cache.
get_cached_public_search_ids	Return streamed public search IDs from Django cache.
get_cached_public_search_state	Return streamed public search state from Django cache.
iter_url_search_prefixes	Yield URL prefixes that can use indexed startswith scans for common search input.
url_prefix_upper_bound	Return the exclusive upper bound for an indexed URL prefix range.
iter_url_prefix_search_ids	Yield IDs for one URL prefix using the URL index, then apply caller filters.
iter_meta_search_ids	Yield metadata search matches from a filtered Snapshot queryset.
normalize_search_result_id	Return a compact Snapshot ID string from a search provider result.
iter_filtered_search_result_ids	Yield provider IDs that still match the filtered queryset.

continues on next page

Table 199 – continued from previous page

iter_search_result_ids	Yield filtered Snapshot IDs from the selected search provider.
snapshot_search_stream_response	Stream Snapshot search progress and cache matching IDs for a list view.
admin_snapshot_search_stream_view	Stream admin Snapshot search progress and cache matching IDs.
public_snapshot_search_stream_view	Stream public Snapshot search progress and cache matching IDs.

Data

SEARCH_RESULT_CACHE_TTL
URL_PREFIX_SEARCH_LIMIT

API

archivebox.search.views.**SEARCH_RESULT_CACHE_TTL**

60

archivebox.search.views.**URL_PREFIX_SEARCH_LIMIT**

500

archivebox.search.views.**get_admin_search_cache_key**(*request*, *url: str | None = None*) → str

Build the cache key for one user and changelist URL.

archivebox.search.views.**get_public_search_cache_key**(*request*, *url: str | None = None*) → str

Build the cache key for one public search URL.

archivebox.search.views.**get_cached_admin_search_ids**(*request*) → list[str] | None

Return streamed admin search IDs from Django cache.

archivebox.search.views.**get_cached_public_search_ids**(*request*) → list[str] | None

Return streamed public search IDs from Django cache.

archivebox.search.views.**get_cached_public_search_state**(*request*) → dict | None

Return streamed public search state from Django cache.

archivebox.search.views.**iter_url_search_prefixes**(*query: str*)

Yield URL prefixes that can use indexed startswith scans for common search input.

archivebox.search.views.**url_prefix_upper_bound**(*prefix: str*) → str

Return the exclusive upper bound for an indexed URL prefix range.

archivebox.search.views.**iter_url_prefix_search_ids**(*prefix: str*, *queryset*)

Yield IDs for one URL prefix using the URL index, then apply caller filters.

archivebox.search.views.**iter_meta_search_ids**(*query*, *queryset*)

Yield metadata search matches from a filtered Snapshot queryset.

archivebox.search.views.**normalize_search_result_id**(*snapshot_id*) → str | None

Return a compact Snapshot ID string from a search provider result.

archivebox.search.views.**iter_filtered_search_result_ids**(*iterator*, *queryset*, ***, *flush_max_delay=0.05*)

Yield provider IDs that still match the filtered queryset.

This is the single intersection/dedupe path used for metadata and every search backend. It flushes by elapsed time so sparse providers stream rows as soon as IDs are found instead of waiting for a fixed batch size.

archivebox.search.views.**iter_search_result_ids**(*query*, *base_queryset*, ***, *search_mode*, *config*, *stop_event=None*)

Yield filtered Snapshot IDs from the selected search provider.

archivebox.search.views.**snapshot_search_stream_response**(*query*, *base_queryset*, ***, *search_mode*, *config*, *cache_key*, *thread_name*)

Stream Snapshot search progress and cache matching IDs for a list view.

archivebox.search.views.**admin_snapshot_search_stream_view**(*model_admin*, *request*)

Stream admin Snapshot search progress and cache matching IDs.

archivebox.search.views.**public_snapshot_search_stream_view**(*request*)

Stream public Snapshot search progress and cache matching IDs.

archivebox.search.query

Module Contents

Functions

escape_like_query	Escape a string for SQL LIKE matching (used with ESCAPE ‘’).
crawl_config_values_search_wave	Build a Snapshot Q predicate matching values inside Crawl.config.
snapshot_metadata_search_waves	Build ordered metadata predicates for Snapshot search.
prioritize_metadata_matches	Rank metadata hits before backend full-text hits.
apply_snapshot_search	Apply shared CLI/API/public/admin Snapshot search semantics.
query_search_index	Return a Snapshot queryset from backend search IDs.
iter_query_search_ids	Yield snapshot IDs from configured search backend modules.
flush_search_index	Remove Snapshot IDs from the configured search backend index.

Data

MAX_SEARCH_RANK_IDS

API

archivebox.search.query.**MAX_SEARCH_RANK_IDS**

500

archivebox.search.query.**escape_like_query**(*query: str*) → str

Escape a string for SQL LIKE matching (used with ESCAPE ‘’).

archivebox.search.query.**crawl_config_values_search_wave**(*query: str*) → django.db.models.Q | None

Build a Snapshot Q predicate matching values inside Crawl.config.

archivebox.search.query.**snapshot_metadata_search_waves**(*query: str*, *, *include_id_matches: bool = False*) → list[django.db.models.Q]

Build ordered metadata predicates for Snapshot search.

archivebox.search.query.**prioritize_metadata_matches**(*base_queryset: django.db.models.QuerySet*, *metadata_queryset: django.db.models.QuerySet*, *fulltext_queryset: django.db.models.QuerySet*, *, *deep_queryset: django.db.models.QuerySet | None = None*, *ordering: list[str] | tuple[str, ...] | None = None*) → django.db.models.QuerySet

Rank metadata hits before backend full-text hits.

archivebox.search.query.**apply_snapshot_search**(*base_queryset: django.db.models.QuerySet*, *query: str*, *, *search_mode: str | None = None*, *config: dict[str, Any] | None = None*, *ordering: list[str] | tuple[str, ...] | None = None*, *max_results: int | None = None*, *skip_backend_when_metadata_satisfies_limit: bool = False*, *include_metadata_for_forced_backend: bool = False*, *include_id_matches: bool = False*) → django.db.models.QuerySet

Apply shared CLI/API/public/admin Snapshot search semantics.

archivebox.search.query.**query_search_index**(*query: str*, *search_mode: str | None = None*, *config: dict[str, Any] | None = None*, *max_results: int | None = None*, ***config_kwargs: Any*) → django.db.models.QuerySet

Return a Snapshot queryset from backend search IDs.

archivebox.search.query.**iter_query_search_ids**(*query: str*, *search_mode: str | None = None*, *config: dict[str, Any] | None = None*, *max_results: int | None = None*, *stop_event=None*, ***config_kwargs: Any*)

Yield snapshot IDs from configured search backend modules.

archivebox.search.query.**flush_search_index**(*snapshots: django.db.models.QuerySet*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → None

Remove Snapshot IDs from the configured search backend index.

archivebox.search.config

Module Contents

Functions

get_default_search_mode	Choose the default search mode from config and discovered backends.
get_search_mode	Normalize a user-supplied search mode or fall back to the default.
get_search_mode_base	Return the mode portion of a normalized search mode.
get_search_mode_backend	Return the backend portion of a backend-qualified search mode.

continues on next page

Table 203 – continued from previous page

get_search_mode_options	Build search mode choices for admin and public selectors.

Data

SEARCH_MODES

API

archivebox.search.config.**SEARCH_MODES**

('meta', 'contents', 'deep')

archivebox.search.config.**get_default_search_mode**(*config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

Choose the default search mode from config and discovered backends.

archivebox.search.config.**get_search_mode**(*search_mode: str | None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

Normalize a user-supplied search mode or fall back to the default.

archivebox.search.config.**get_search_mode_base**(*search_mode: str | None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str

Return the mode portion of a normalized search mode.

archivebox.search.config.**get_search_mode_backend**(*search_mode: str | None*, *config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → str | None

Return the backend portion of a backend-qualified search mode.

archivebox.search.config.**get_search_mode_options**(*config: dict[str, Any] | None = None*, ***config_kwargs: Any*) → list[dict[str, str]]

Build search mode choices for admin and public selectors.

archivebox.search.admin

Module Contents

Classes

SearchResultsChangeList	Django admin ChangeList with ArchiveBox search mode state.
SearchResultsAdminMixin	Mixin that routes admin searches through ArchiveBox search modes.

API

class archivebox.search.admin.**SearchResultsChangeList**(*request*, **args*, ***kwargs*)

Bases: `django.contrib.admin.views.main.ChangeList`

Django admin ChangeList with ArchiveBox search mode state.

Initialization

Capture normalized search mode before Django builds results.

get_results(*request*)

Populate normal admin results plus search-index hint state.

get_filters_params(*params=None*)

Remove UI-only search params before admin filter processing.

class archivebox.search.admin.**SearchResultsAdminMixin**

Bases: `django.contrib.admin.ModelAdmin`

Mixin that routes admin searches through ArchiveBox search modes.

show_search_mode_selector

True

get_changelist(*request*, ***kwargs*)

Return the ArchiveBox search-aware ChangeList class.

get_default_search_mode()

Return the default search mode for the current request config.

get_search_mode_options()

Return selector options for the current request config.

get_search_results(*request*, *queryset*, *search_term: str*)

Apply admin search semantics to a changelist queryset.

archivebox.mcp

Submodules

archivebox.mcp.apps

Module Contents

Classes

MCPConfig

API

class archivebox.mcp.apps.**MCPConfig**

Bases: `django.apps.AppConfig`

name

'mcp'

verbose_name

'Model Context Protocol Server'

default_auto_field

'django.db.models.BigAutoField'

archivebox.mcp.server

Model Context Protocol server for ArchiveBox.

The server exposes five agent-friendly workflow tools backed by ArchiveBox's existing Click CLI and newline-delimited JSON-RPC.

Module Contents

Classes

MCPJSONEncoder	JSON encoder for UUIDs, paths, Click sentinels, and other CLI values.
MCPTool	A discovered leaf Click command and its ArchiveBox command path.
MCPServer	ArchiveBox MCP server using JSON-RPC 2.0 over stdio.

Functions

click_type_to_json_schema_type	Convert a Click parameter type to JSON Schema.
command_accepts_stdin	Return whether a command documents a stdin/JSONL input path.
tool_annotations	Describe a command's side effects to MCP clients.
click_command_to_mcp_tool	Convert a leaf Click command to an MCP tool definition.
tool_output_schema	Return the shared envelope schema for all CLI-backed tools.
click_group_to_mcp_tool	Expose a Click command group as one MCP tool with an action selector.
shell_to_mcp_tool	Expose `archivebox shell -c` as one explicit Python escape hatch.
_option_args	Serialize one MCP argument using the option's real Click spellings.
arguments_to_cli	Convert MCP JSON arguments into Click argv and optional stdin.
parse_structured_records	Parse a CLI JSON array or JSONL stream without guessing at human output.
execute_click_command	Execute one discovered CLI command and return MCP structured content.
_tool_result	
run_mcp_server	Start the ArchiveBox MCP stdio server.

Data

PROTOCOL_VERSION
PUBLIC_TOOLS
ACTION_TOOLS
READ_ONLY_ACTIONS
DESTRUCTIVE_ACTIONS

API

archivebox.mcp.server.**PROTOCOL_VERSION**

'2025-11-25'

archivebox.mcp.server.**PUBLIC_TOOLS**

('add', 'search', 'crawl', 'snapshot', 'archiveresult', 'shell')

archivebox.mcp.server.**ACTION_TOOLS**

None

archivebox.mcp.server.**READ_ONLY_ACTIONS**

None

archivebox.mcp.server.**DESTRUCTIVE_ACTIONS**

None

class archivebox.mcp.server.**MCPJSONEncoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: `json.JSONEncoder`

JSON encoder for UUIDs, paths, Click sentinels, and other CLI values.

Initialization

Constructor for JSONEncoder, with sensible defaults.

If skipkeys is false, then it is a TypeError to attempt encoding of keys that are not str, int, float, bool or None. If skipkeys is True, such items are simply skipped.

If ensure_ascii is true, the output is guaranteed to be str objects with all incoming non-ASCII characters escaped. If ensure_ascii is false, the output can contain non-ASCII characters.

If check_circular is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an RecursionError). Otherwise, no such check takes place.

If allow_nan is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a ValueError to encode such floats.

If sort_keys is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If indent is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, separators should be an (item_separator, key_separator) tuple. The default is (', ', ': ') if *indent* is `None` and (',', ': ') otherwise. To get the most compact JSON representation, you should specify (',', ':') to eliminate whitespace.

If specified, default is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a `TypeError`.

default(*value*)

class archivebox.mcp.server.**MCPTool**

A discovered leaf Click command and its ArchiveBox command path.

name: str

None

command_path: tuple[str, ...]

None

command: click.Command

None

archivebox.mcp.server.**click_type_to_json_schema_type**(*click_type: click.ParamType*) → dict[str, Any]

Convert a Click parameter type to JSON Schema.

archivebox.mcp.server.**command_accepts_stdin**(*click_command: click.Command*) → bool

Return whether a command documents a stdin/JSONL input path.

archivebox.mcp.server.**tool_annotations**(*command_path: tuple[str, ...]*) → dict[str, Any]

Describe a command's side effects to MCP clients.

archivebox.mcp.server.**click_command_to_mcp_tool**(*tool:* archivebox.mcp.server.MCPTool) → dict[str, Any]

Convert a leaf Click command to an MCP tool definition.

archivebox.mcp.server.**tool_output_schema**() → dict[str, Any]

Return the shared envelope schema for all CLI-backed tools.

archivebox.mcp.server.**click_group_to_mcp_tool**(*group_name: str*, *actions: list[*archivebox.mcp.server.MCPTool*]*) → dict[str, Any]

Expose a Click command group as one MCP tool with an action selector.

archivebox.mcp.server.**shell_to_mcp_tool**() → dict[str, Any]

Expose `archivebox shell -c` as one explicit Python escape hatch.

archivebox.mcp.server.**_option_args**(*param: click.Option*, *value: Any*) → list[str]

Serialize one MCP argument using the option's real Click spellings.

archivebox.mcp.server.**arguments_to_cli**(*click_command: click.Command*, *arguments: dict[str, Any]*) → tuple[list[str], str | None]

Convert MCP JSON arguments into Click argv and optional stdin.

archivebox.mcp.server.**parse_structured_records**(*stdout: str*) → list[Any]

Parse a CLI JSON array or JSONL stream without guessing at human output.

archivebox.mcp.server.**execute_click_command**(*tool:* archivebox.mcp.server.MCPTool, *arguments: dict[str, Any]*) → dict[str, Any]

Execute one discovered CLI command and return MCP structured content.

archivebox.mcp.server.**_tool_result**(*tool:* archivebox.mcp.server.MCPTool, *, exit_code: int*, *stdout: str = ''*, *stderr: str = ''*, *is_error: bool*) → dict[str, Any]

class archivebox.mcp.server.**MCPServer**

ArchiveBox MCP server using JSON-RPC 2.0 over stdio.

Initialization

_discover_command(*command: click.Command*, *command_path: tuple[str, ...]*, *tools: dict[str,* archivebox.mcp.server.MCPTool*]*) → None

get_tools() → dict[str, *archivebox.mcp.server.MCPTool*]

Discover and cache the CLI commands backing the five public tools.

get_public_tool_definitions() → list[dict[str, Any]]

Build the small agent-facing surface from the existing Click tree.

handle_initialize(*params: dict*) → dict

handle_tools_list(*params: dict*) → dict

handle_tools_call(*params: dict*) → dict

handle_request(*request: dict*) → dict | None

run_stdio_server() → None

Read and write one UTF-8 JSON-RPC message per line.

archivebox.mcp.server.**run_mcp_server**() → None

Start the ArchiveBox MCP stdio server.

archivebox.personas

Submodules

archivebox.personas.importers

Shared persona browser discovery/import helpers.

These helpers are used by both the CLI and the Django admin so Persona import behavior stays consistent regardless of where it is triggered from.

Module Contents

Classes

PersonaImportSource	
PersonaImportResult	

Functions

_path_exists	
_path_is_dir	
_iter_children	
_is_copyable_profile_entry	
profile_copy_ignore	
get_chrome_user_data_dir	Get the default Chrome user data directory for the current platform.
get_chromium_user_data_dir	Select Chromium's own profile even when Google Chrome is installed.
get_brave_user_data_dir	Get the default Brave user data directory for the current platform.
get_edge_user_data_dir	Get the default Edge user data directory for the current platform.

continues on next page

Table 211 – continued from previous page

validate_persona_name	Validate persona name to prevent path traversal.
discover_local_browser_profiles	
discover_persona_template_profiles	
resolve_browser_import_source	
resolve_browser_profile_source	
resolve_custom_import_source	
pick_default_profile_dir	
resolve_source_browser_binary	Use the originating browser so OS-protected cookies use the correct keychain.
import_persona_from_source	
export_profile_cookies	Decode with the host keychain, never by starting the user's browser.
copy_browser_user_data_dir	
export_browser_state	
_list_profile_names	
_looks_like_profile_dir	
_looks_like_cdp_url	
_parse_netscape_cookies	
_write_netscape_cookies	
_merge_netscape_cookies	
_merge_auth_storage	
_load_auth_storage	
_merge_cookie_dicts	
_apply_imported_user_agent	

Data

BROWSER_LABELS
BROWSER_PROFILE_DIR_NAMES
VOLATILE_PROFILE_COPY_PATTERNS
PERSONA_PROFILE_DIR_CANDIDATES
BROWSER_PROFILE_FINDERS
CHROMIUM_BROWSERS
NETSCAPE_COOKIE_HEADER

API

archivebox.personas.importers.**BROWSER_LABELS**

None

archivebox.personas.importers.**BROWSER_PROFILE_DIR_NAMES**

('Default', 'Profile ', 'Guest Profile')

archivebox.personas.importers.**VOLATILE_PROFILE_COPY_PATTERNS**

('.*.??????', 'Cache', 'Code Cache', 'GPUCache', 'ShaderCache', 'Crashpad', 'BrowserMetrics', 'Brows…

archivebox.personas.importers.**PERSONA_PROFILE_DIR_CANDIDATES**

('chrome_profile', 'chrome_user_data')

archivebox.personas.importers.**_path_exists**(*path: pathlib.Path*) → bool

archivebox.personas.importers.**_path_is_dir**(*path: pathlib.Path*) → bool

archivebox.personas.importers.**_iter_children**(*path: pathlib.Path*) → list[pathlib.Path]

archivebox.personas.importers.**_is_copyable_profile_entry**(*path: pathlib.Path*) → bool

archivebox.personas.importers.**profile_copy_ignore**(*src: str*, *names: list[str]*) → set[str]

class archivebox.personas.importers.**PersonaImportSource**

kind: str
None

browser: str
'custom'

source_name: str | None
None

user_data_dir: pathlib.Path | None
None

profile_dir: str | None
None

browser_binary: str | None
None

cdp_url: str | None
None

property browser_label: str

property profile_path: pathlib.Path | None

property display_label: str

property choice_value: str

as_choice_label() → django.utils.safestring.SafeString

classmethod from_choice_value(*value: str*) → *archivebox.personas.importers.PersonaImportSource*

class archivebox.personas.importers.**PersonaImportResult**

source: *archivebox.personas.importers.PersonaImportSource*
None

profile_copied: bool
False

cookies_imported: bool
False

cookie_count: int
0

storage_captured: bool
False

user_agent_imported: bool

False

warnings: list[str]

'field(...)'

property did_work: bool

archivebox.personas.importers.**get_chrome_user_data_dir**(*, *beta: bool = False*) → pathlib.Path | None

Get the default Chrome user data directory for the current platform.

archivebox.personas.importers.**get_chromium_user_data_dir**() → pathlib.Path | None

Select Chromium's own profile even when Google Chrome is installed.

archivebox.personas.importers.**get_brave_user_data_dir**() → pathlib.Path | None

Get the default Brave user data directory for the current platform.

archivebox.personas.importers.**get_edge_user_data_dir**() → pathlib.Path | None

Get the default Edge user data directory for the current platform.

archivebox.personas.importers.**BROWSER_PROFILE_FINDERS**

None

archivebox.personas.importers.**CHROMIUM_BROWSERS**

'tuple(...)'

archivebox.personas.importers.**NETSCAPE_COOKIE_HEADER**

['# Netscape HTTP Cookie File', '# https://curl.se/docs/http-cookies.html', '# This file was generat...

archivebox.personas.importers.**validate_persona_name**(*name: str*) → tuple[bool, str]

Validate persona name to prevent path traversal.

archivebox.personas.importers.**discover_local_browser_profiles**() → list[*archivebox.personas.importers.PersonaImportSou*

archivebox.personas.importers.**discover_persona_template_profiles**(*personas_dir: pathlib.Path | None = None*) → list[*archivebox.personas.importers.PersonaImport*

archivebox.personas.importers.**resolve_browser_import_source**(*browser: str*, *profile_dir: str | None = None*) → *archivebox.personas.importers.PersonaImportSource*

archivebox.personas.importers.**resolve_browser_profile_source**(*browser: str*, *user_data_dir: pathlib.Path*, *profile_dir: str*, *source_name: str | None = None*, *browser_binary: str | None = None*) → *archivebox.personas.importers.PersonaImportSource*

archivebox.personas.importers.**resolve_custom_import_source**(*raw_value: str*, *profile_dir: str | None = None*) → *archivebox.personas.importers.PersonaImportSource*

archivebox.personas.importers.**pick_default_profile_dir**(*user_data_dir: pathlib.Path*) → str | None

archivebox.personas.importers.**resolve_source_browser_binary**(*source:* archivebox.personas.importers.PersonaImportSource) → str

Use the originating browser so OS-protected cookies use the correct keychain.

archivebox.personas.importers.**import_persona_from_source**(*persona:* archivebox.personas.models.Persona, *source:* archivebox.personas.importers.PersonaImportSource, *, *copy_profile: bool = True*, *import_cookies: bool = True*, *capture_storage: bool = False*) → *archivebox.personas.importers.PersonaImportResult*

archivebox.personas.importers.**export_profile_cookies**(*source:* archivebox.personas.importers.PersonaImportSource, *staged_profile: pathlib.Path*, *output: pathlib.Path*) → dict

Decode with the host keychain, never by starting the user's browser.

archivebox.personas.importers.**copy_browser_user_data_dir**(*source_dir: pathlib.Path*, *destination_dir: pathlib.Path*) → None

archivebox.personas.importers.**export_browser_state**(*, *user_data_dir: pathlib.Path | None = None*, *cdp_url: str | None = None*, *profile_dir: str | None = None*, *chrome_binary: str | None = None*, *cookies_output_file: pathlib.Path | None = None*, *auth_output_file: pathlib.Path | None = None*) → tuple[bool, dict | None, str]

archivebox.personas.importers.**_list_profile_names**(*user_data_dir: pathlib.Path*) → list[str]

archivebox.personas.importers.**_looks_like_profile_dir**(*path: pathlib.Path*) → bool

archivebox.personas.importers.**_looks_like_cdp_url**(*value: str*) → bool

archivebox.personas.importers.**_parse_netscape_cookies**(*path: pathlib.Path*) → dict[tuple[str, str, str], tuple[str, str, str, str, str, str, str]]

archivebox.personas.importers.**_write_netscape_cookies**(*path: pathlib.Path*, *cookies: dict[tuple[str, str, str], tuple[str, str, str, str, str, str, str]]*) → None

archivebox.personas.importers.**_merge_netscape_cookies**(*existing_file: pathlib.Path*, *new_file: pathlib.Path*) → None

archivebox.personas.importers.**_merge_auth_storage**(*existing_file: pathlib.Path*, *new_file: pathlib.Path*) → None

archivebox.personas.importers.**_load_auth_storage**(*path: pathlib.Path*) → dict

archivebox.personas.importers.**_merge_cookie_dicts**(*existing: list[dict]*, *new: list[dict]*) → list[dict]

archivebox.personas.importers.**_apply_imported_user_agent**(*persona:* archivebox.personas.models.Persona, *auth_payload: dict | None*) → bool

archivebox.personas.forms

Module Contents

Classes

PersonaAdminForm

Functions

_mode_label

API

archivebox.personas.forms.**_mode_label**(*title: str*, *description: str*) → str

class archivebox.personas.forms.**PersonaAdminForm**(**args: Any*, ***kwargs: Any*)

Bases: *archivebox.plugins.forms.PluginConfigFormMixin*, django.forms.ModelForm

Initialization

permissions

'ChoiceField(…)'

import_mode

'ChoiceField(…)'

import_discovered_profile

'ChoiceField(…)'

import_source

'CharField(…)'

import_profile_name

'CharField(…)'

import_copy_profile

'BooleanField(…)'

import_extract_cookies

'BooleanField(…)'

import_capture_storage

'BooleanField(…)'

class Meta

model

None

fields

('name', 'created_by', 'config')

clean_name() → str

clean() → dict[str, Any]

apply_import(*persona:* archivebox.personas.models.Persona) → *archivebox.personas.importers.PersonaImportResult* | None

archivebox.personas.apps

Module Contents

Classes

PersonasConfig	

API

class archivebox.personas.apps.**PersonasConfig**

Bases: django.apps.AppConfig

default_auto_field

'django.db.models.BigAutoField'

name

'archivebox.personas'

label

'personas'

archivebox.personas.models

Persona management for ArchiveBox.

A Persona represents a browser profile/identity used for archiving. Each persona has its own:

- Chrome user data directory (for cookies, localStorage, extensions, etc.)
- Cookies file
- Config overrides

Module Contents

Classes

Persona	Browser persona/profile for archiving sessions.

Functions

derive_persona_config

Data

_fcntl
VOLATILE_PROFILE_DIR_NAMES
VOLATILE_PROFILE_FILE_NAMES

API

archivebox.personas.models.**_fcntl: Any | None**

None

archivebox.personas.models.**VOLATILE_PROFILE_DIR_NAMES**

None

archivebox.personas.models.**VOLATILE_PROFILE_FILE_NAMES**

None

archivebox.personas.models.**derive_persona_config**(*, *name: str*, *config: collections.abc.Mapping[str, Any] | None*, *persona_dir: pathlib.Path*) → dict[str, Any]

class archivebox.personas.models.**Persona**

Bases: *archivebox.base_models.models.ModelWithConfig*

Browser persona/profile for archiving sessions.

Each persona provides:

- CHROME_USER_DATA_DIR: Chrome profile directory
- CHROME_DOWNLOADS_DIR: Chrome downloads directory
- COOKIES_FILE: Cookies file for wget/curl
- config: JSON field with persona-specific config overrides

Usage: # Get persona and its derived config config = get_config(persona=crawl.persona, crawl=crawl, snapshot=snapshot) chrome_dir = config['CHROME_USER_DATA_DIR']

```
# Or access directly from persona
persona = Persona.objects.get(name='Default')
persona.CHROME_USER_DATA_DIR  # -> Path to chrome_profile
```

id

'CompactUUIDField(...)'

name

'CharField(...)'

created_at

'DateTimeField(...)'

created_by

'ForeignKey(...)'

permissions

'GeneratedField(...)'

class Meta

Bases: *archivebox.base_models.models.ModelWithConfig.Meta*

app_label

'personas'

save(**args*, ***kwargs*)

__str__() → str

property path: pathlib.Path

Reuse the directory's spelling without renaming a Chrome profile.

static directory_name(*name: str*) → str

classmethod find_named(*name: str*) → *Persona* | None

validate_name(*name: str*) → str

property CHROME_USER_DATA_DIR: str

Derived path to Chrome user data directory for this persona.

property CHROME_DOWNLOADS_DIR: str

Derived path to Chrome downloads directory for this persona.

property COOKIES_FILE: str

Derived path to cookies.txt file for this persona (if exists).

property AUTH_STORAGE_FILE: str

Derived path to a nonempty auth.json export for this persona.

get_derived_config() → dict

Get config dict with derived paths filled in.

Returns dict with:

- All values from self.config JSONField
- PERSONAS_DIR (derived from DATA_DIR/personas)
- COOKIES_FILE (derived from persona path, if file exists)
- AUTH_STORAGE_FILE (derived from persona path, if file exists)
- ACTIVE_PERSONA (set to this persona's name)

ensure_dirs() → None

Create persona directories if they don't exist.

cleanup_chrome_profile(*profile_dir: pathlib.Path*) → bool

Remove volatile Chrome state that should never be reused across launches.

cleanup_chrome() → bool

Clean up volatile Chrome state for this persona's base profile.

lock_runtime_for_crawl()

classmethod get_or_create_named(*name: str*, *, *defaults: dict | None = None*) → *archivebox.personas.models.Persona*

runtime_root_for_crawl(*crawl*) → pathlib.Path

runtime_profile_dir_for_crawl(*crawl*) → pathlib.Path

runtime_downloads_dir_for_crawl(*crawl*) → pathlib.Path

runtime_root_for_snapshot(*snapshot*) → pathlib.Path

runtime_profile_dir_for_snapshot(*snapshot*) → pathlib.Path

runtime_downloads_dir_for_snapshot(*snapshot*) → pathlib.Path

copy_chrome_profile(*source_dir: pathlib.Path*, *destination_dir: pathlib.Path*) → None

prepare_runtime_for_crawl(*crawl*, *chrome_binary: str = ''*) → dict[str, str]

prepare_runtime_for_snapshot(*snapshot*, *chrome_binary: str = ''*) → dict[str, str]

cleanup_runtime_for_crawl(*crawl*) → None

classmethod get_or_create_default() → *archivebox.personas.models.Persona*

Get or create the Default persona.

classmethod cleanup_chrome_all() → int

Clean up Chrome state files for all personas.

archivebox.personas.admin

Module Contents

Classes

PersonaAdmin

Functions

register_admin

API

class archivebox.personas.admin.**PersonaAdmin**

Bases: *archivebox.base_models.admin.ConfigEditorMixin*, *archivebox.base_models.admin.BaseModelAdmin*

form

None

change_form_template

'admin/personas/persona/change_form.html'

list_display

('name', 'created_by', 'created_at', 'chrome_profile_state', 'cookies_state', 'auth_state')

search_fields

('name', 'created_by__username')

list_filter
('created_by',)

ordering
['name']

list_per_page
100

readonly_fields
('id', 'created_at', 'persona_paths', 'import_artifact_status')

add_fieldsets
(('Persona',), ('Browser Import',), ('Advanced',))

change_fieldsets
()

chrome_profile_state(*obj:* archivebox.personas.models.Persona) → str

cookies_state(*obj:* archivebox.personas.models.Persona) → str

auth_state(*obj:* archivebox.personas.models.Persona) → str

persona_paths(*obj:* archivebox.personas.models.Persona) → str

import_artifact_status(*obj:* archivebox.personas.models.Persona) → str

get_fieldsets(*request*, *obj=None*)

get_form(*request*, *obj=None*, *change=False*, ***kwargs*)

render_change_form(*request*, *context*, *add=False*, *change=False*, *form_url=''*, *obj=None*)

save_model(*request*, *obj*, *form*, *change*)

archivebox.personas.admin.**register_admin**(*admin_site: django.contrib.admin.AdminSite*) → None

archivebox.crawls

Submodules

archivebox.crawls.apps

Module Contents

Classes

CrawlsConfig

API

class archivebox.crawls.apps.**CrawlsConfig**

Bases: django.apps.AppConfig

default_auto_field
'django.db.models.BigAutoField'

name

'archivebox.crawls'

label

'crawls'

archivebox.crawls.schedule_util

Module Contents

Functions

normalize_schedule
validate_schedule
next_run_for_schedule

Data

SCHEDULE_ALIASES

API

archivebox.crawls.schedule_util.**SCHEDULE_ALIASES: dict[str, str]**

None

archivebox.crawls.schedule_util.**normalize_schedule**(*schedule: str*) → str

archivebox.crawls.schedule_util.**validate_schedule**(*schedule: str*) → str

archivebox.crawls.schedule_util.**next_run_for_schedule**(*schedule: str*, *after: datetime.datetime*) → datetime.datetime

archivebox.crawls.models

Module Contents

Classes

CrawlSchedule
Crawl

API

class archivebox.crawls.models.**CrawlSchedule**

Bases: *archivebox.base_models.models.ModelWithUUID*, *archivebox.base_models.models.ModelWithNotes*

id

'CompactUUIDField(...)'

created_at

'DateTimeField(...)'

created_by

'ForeignKey(...)'

modified_at

'DateTimeField(...)'

template: *Crawl*

'ForeignKey(...)'

schedule

'CharField(...)'

is_enabled

'BooleanField(...)'

config

'JSONField(...)'

label

'CharField(...)'

notes

'TextField(...)'

crawl_set: **django.db.models.Manager[*Crawl*]**

None

class Meta

Bases: *archivebox.base_models.models.ModelWithUUID.Meta*, *archivebox.base_models.models.ModelWithNotes.Meta*

app_label

'crawls'

verbose_name

'Scheduled Crawl'

verbose_name_plural

'Scheduled Crawls'

__str__() → str

property api_url: str

save(**args*, ***kwargs*)

property last_run_at

property next_run_at

is_due(*now=None*) → bool

property kind: str

dispatch(*queued_at=None*) → *Crawl* | None

Run maintenance directly or enqueue one ordinary Crawl.

enqueue(*queued_at=None*) → *archivebox.crawls.models.Crawl*

class archivebox.crawls.models.**Crawl**

Bases: *archivebox.base_models.models.ModelWithDeleteAfter*, *archivebox.base_models.models.ModelWithOutputDir*, *archivebox.base_models.models.ModelWithConfig*, *archivebox.base_models.models.ModelWithHealthStats*, *archivebox.workers.models.ModelWithQueue*

objects

'as_manager(...)'

id

'CompactUUIDField(...)'

created_at

'DateTimeField(...)'

created_by

'ForeignKey(...)'

modified_at

'DateTimeField(...)'

urls

'TextField(...)'

config

'JSONField(...)'

permissions

'GeneratedField(...)'

max_depth

'PositiveSmallIntegerField(...)'

tags_str

'CharField(...)'

persona

'ForeignKey(...)'

label

'CharField(...)'

notes

'TextField(...)'

schedule

'ForeignKey(...)'

status

'StatusField(...)'

retry_at

'RetryAtField(...)'

retry_at_field_name

'retry_at'

state_field_name

'status'

StatusChoices

None

INITIAL_STATE

None

ACTIVE_STATE

None

FINAL_STATES

()

FINAL_OR_ACTIVE_STATES

()

active_state

None

delete_after_final_statuses

()

RUNNABLE_STATES

()

INACTIVE_STATES

()

schedule_id: uuid.UUID | None

None

snapshot_set: django.db.models.Manager[*archivebox.core.models.Snapshot*]

None

class Meta

Bases: *archivebox.base_models.models.ModelWithDeleteAfter.Meta*, *archivebox.base_models.models.ModelWithOutputDir.Meta*, *archivebox.base_models.models.ModelWithConfig.Meta*, *archivebox.base_models.models.ModelWithHealthStats.Meta*, *archivebox.workers.models.ModelWithQueue.Meta*

app_label

'crawls'

verbose_name

'Crawl'

verbose_name_plural

'Crawls'

indexes

None

__str__()

get_delete_after_config_value()

pause(*, *save: bool = True*) → bool

resume(*, *when=None*, *save: bool = True*) → bool

cancel() → None

schedule_child_snapshots_for_sealing() → int

schedule_child_snapshots_for_pause() → int

classmethod missing_delete_at_candidates()

save(**args*, ***kwargs*)

update_child_snapshot_permissions(*old_permissions: str | None*, *new_permissions: str | None*) → int

property api_url: str

static parse_tag_names(*tags: collections.abc.Iterable[str] | str*, *, *pattern: str = ','*) → list[str]

current_tag_names() → list[str]

apply_snapshot_tag_diff(*, *added_tag_names: collections.abc.Iterable[str]*, *removed_tag_names: collections.abc.Iterable[str]*) → None

to_json() → dict

Convert Crawl model instance to a JSON-serializable dict.

static from_json(*record: dict*, *overrides: dict | None = None*)

Create or get a Crawl from a JSON dict.

Args: record: Dict with 'urls' (required), optional 'max_depth', 'tags_str', 'label' overrides: Dict of field overrides (e.g., created_by_id)

Returns: Crawl instance or None if invalid

property output_dir: pathlib.Path

get_urls_list() → list[str]

Get list of URLs from urls field, filtering out comments and empty lines.

static normalize_domain(*value: str*) → str

static split_filter_patterns(*value*) → list[str]

classmethod _pattern_matches_url(*url: str*, *pattern: str*) → bool

get_current_config(*, *refresh: bool = False*) → dict[str, Any]

get_url_allowlist(*, *use_effective_config: bool = False*, *snapshot=None*) → list[str]

get_url_denylist(*, *use_effective_config: bool = False*, *snapshot=None*) → list[str]

url_passes_filters(*url: str*, *, *snapshot=None*, *use_effective_config: bool = True*) → bool

url_passes_compiled_filters(*url: str*, *, *allowlist: list[str]*, *denylist: list[str]*) → bool

set_url_filters(*allowlist*, *denylist*) → None

apply_crawl_config_filters() → dict[str, int]

_iter_url_lines() → list[tuple[str, str]]

count_urls_for_limit() → int

Count unique URLs already queued or snapshotted for this crawl.

max_urls is a crawl-wide cap on snapshots, so direct URL entries and recursively discovered snapshots both have to consume the same budget.

remaining_url_capacity() → int | None

has_remaining_url_capacity() → bool

remaining_snapshot_capacity() → int | None

has_remaining_snapshot_capacity() → bool

prune_urls(*predicate*) → list[str]

prune_url(*url: str*) → int

exclude_domain(*domain: str*) → dict[str, int | str | bool]

resolve_persona()

static _config_value(*config: collections.abc.Mapping[str, Any] | Any*, *key: str*, *default: Any = None*) → Any

classmethod create_scheduler_row(**kwargs*) → *archivebox.crawls.models.Crawl*

limit_stop_reason(*, *config: collections.abc.Mapping[str, Any] | Any | None = None*, *output_dir: pathlib.Path | None = None*, *num_snapshots: int | None = None*) → str

lifecycle_stop_reason(*, *num_snapshots: int | None = None*, *num_sealed_snapshots: int | None = None*) → str

stop_reason(*, *config: collections.abc.Mapping[str, Any] | Any | None = None*, *output_dir: pathlib.Path | None = None*, *num_snapshots: int | None = None*, *num_sealed_snapshots: int | None = None*) → str

add_url(*entry: dict*) → bool

Add a URL to the crawl queue if not already present.

Args: entry: dict with 'url', optional 'depth', 'title', 'timestamp', 'tags', 'via_snapshot', 'plugin'

Returns: True if URL was added, False if skipped (duplicate or depth exceeded)

create_snapshots_from_urls() → list[*archivebox.core.models.Snapshot*]

Create Snapshot objects for each URL in self.urls that doesn't already exist.

Returns: List of newly created Snapshot objects

create_discovered_snapshot(*parent_snapshot*, *, *url: str*, *depth: int*, *title: str = ''*, *tags: str = ''*, *created_by_id: int | None = None*)

Create one child snapshot if it passes crawl filters and limits.

create_discovered_snapshots(*parent_snapshot*, *records: collections.abc.Iterable[collections.abc.Mapping[str, Any]]*, *, *depth: int*, *created_by_id: int | None = None*) → list[*archivebox.core.models.Snapshot*]

Create child snapshots from discovered URL records after filtering and deduping once.

is_finished() → bool

Check if crawl is finished (all snapshots sealed or no snapshots exist).

can_start() → bool

has_finished_snapshots() → bool

mark_started() → bool

seal() → bool

Finalize a runner-owned Crawl without dispatching hooks directly.

advance_lifecycle() → bool

Advance one explicit lifecycle step after the runner claims this row.

cleanup_runtime() → None

Remove runner-owned runtime artifacts after abx-dl cleanup hooks finish.

archivebox.crawls.locks

Module Contents

Classes

_LifecycleLockState

Functions

crawl_lifecycle_lock_path
_lifecycle_lock
crawl_lifecycle_lock
binary_lifecycle_lock

Data

_registry_lock
_registry_pid
_registry

API

class archivebox.crawls.locks.**_LifecycleLockState**

Initialization

archivebox.crawls.locks.**_registry_lock**

'Lock(...)'

archivebox.crawls.locks.**_registry_pid**

'getpid(...)'

archivebox.crawls.locks._registry: dict[str, archivebox.crawls.locks._LifecycleLockState]

None

archivebox.crawls.locks.**crawl_lifecycle_lock_path**(*crawl_id: str*) → pathlib.Path

archivebox.crawls.locks._**lifecycle_lock**(*key: str*, *lock_path: pathlib.Path*) → collections.abc.Iterator[None]

archivebox.crawls.locks.**crawl_lifecycle_lock**(*crawl_id: str*) → collections.abc.Iterator[None]

archivebox.crawls.locks.**binary_lifecycle_lock**(*binary_id: str*) → collections.abc.Iterator[None]

archivebox.crawls.admin

Module Contents

Classes

MaxDepthListFilter	
URLFiltersField	
CrawlAdminForm	Custom form for Crawl admin to render urls field as textarea.
CrawlAdmin	
CrawlScheduleAdmin	

Functions

render_snapshots_list	Render a nice inline list view of snapshots with status, title, URL, and progress.
register_admin	

API

class archivebox.crawls.admin.**MaxDepthListFilter**

Bases: django.contrib.admin.SimpleListFilter

title

'max depth'

parameter_name

'max_depth'

lookups(*request*, *model_admin*)

queryset(*request*, *queryset*)

archivebox.crawls.admin.**render_snapshots_list**(*snapshots_qs*, *request=None*, *crawl=None*, *page_size=50*, *prefix='snapshots'*)

Render a nice inline list view of snapshots with status, title, URL, and progress.

class archivebox.crawls.admin.**URLFiltersField**

Bases: django.forms.Field

widget

'URLFiltersWidget(…)'

to_python(*value*)

class archivebox.crawls.admin.**CrawlAdminForm**(**args*, ***kwargs*)

Bases: `django.forms.ModelForm`

Custom form for Crawl admin to render urls field as textarea.

Initialization

tags_editor

'CharField(…)'

url_filters

'URLFiltersField(…)'

class Meta

model

None

fields

'**all**'

widgets: ClassVar[dict[str, django.forms.Widget]]

None

static extract_url_line(*line*)

static regex_escape(*text*)

classmethod generated_host_allowlist(*urls*)

static subpath_prefix(*pathname*)

static parsed_host_and_port(*parsed*)

classmethod generated_subpath_allowlist(*urls*)

classmethod derive_filter_toggles(*urls*, *allowlist*)

static effective_only_new(*crawl=None*)

static inherited_only_new(*crawl*)

clean_tags_editor()

clean_url_filters()

save(*commit=True*)

class archivebox.crawls.admin.**CrawlAdmin**(*model*, *admin_site*)

Bases: *archivebox.base_models.admin.ConfigEditorMixin*, *archivebox.base_models.admin.BaseModelAdmin*

form

None

change_form_template

'admin/crawls/crawl/change_form.html'

list_select_related

()

paginator

None

show_full_result_count

False

list_display

('short_id', 'permissions_badge', 'created_at', 'owner', 'depth', 'status_with_stop_reason', 'pause_…

sort_fields

('id', 'created_at', 'created_by', 'max_depth', 'label', 'notes', 'schedule_str', 'status', 'retry_a…

search_fields

('id', 'created_by__username', 'max_depth', 'label', 'notes', 'schedule_id', 'status', 'urls')

readonly_fields

('created_at', 'modified_at', 'stop_reason_display')

fieldsets

(('URLs',), ('Overview',), ('Config',))

add_fieldsets

(('URLs',), ('Overview',), ('Config',))

list_filter

()

ordering

('-created_at', '-retry_at')

list_per_page

50

actions

('pause_selected_crawls', 'resume_selected_crawls', 'seal_selected_crawls', 'delete_selected_batched…

change_actions

('recrawl',)

class Media

css: ClassVar[dict[str, tuple[str, ...]]]

None

js: ClassVar[tuple[str, ...]]

('admin/crawls/crawl_admin.js',)

changelist_view(*request*, *extra_context=None*)

should_annotate_snapshot_counts(*request*)

hydrate_visible_snapshot_counts(*crawls*)

get_queryset(*request*)

Keep joins page-local while computing per-row snapshot counts in the page query.

change_view(*request*, *object_id*, *form_url=''*, *extra_context=None*)

add_view(*request*, *form_url=''*, *extra_context=None*)

get_fieldsets(*request*, *obj=None*)

get_urls()

get_actions(*request*)

delete_selected_batched(*request*, *queryset*)

Delete crawls in a single transaction to avoid SQLite concurrency issues.

pause_selected_crawls(*request*, *queryset*)

resume_selected_crawls(*request*, *queryset*)

seal_selected_crawls(*request*, *queryset*)

set_crawl_permissions(*request*, *queryset*)

update_crawl_permissions(*queryset*, *permissions*)

recrawl(*request*, *obj*)

Duplicate this crawl as a new crawl with the same URLs and settings.

stop_reason_display(*obj*)

stop_reason_for_crawl(*obj*)

limit_config_for_crawl(*obj*, *output_dir*)

status_with_stop_reason(*obj*)

short_id(*obj*)

owner(*obj*)

depth(*obj*)

permissions_badge(*obj*)

pause_resume_control(*obj*)

num_archived_snapshots(*obj*)

num_total_snapshots(*obj*)

snapshots_changelist(*obj*)

delete_snapshot_view(*request: django.http.HttpRequest*, *object_id: str*, *snapshot_id: str*)

exclude_domain_view(*request: django.http.HttpRequest*, *object_id: str*, *snapshot_id: str*)

set_permissions_view(*request: django.http.HttpRequest*, *object_id: str*)

schedule_str(*obj*)

urls_preview(*obj*)

urls_editor(*obj*)

Editor for crawl URLs.

class archivebox.crawls.admin.**CrawlScheduleAdmin**

Bases: *archivebox.base_models.admin.BaseModelAdmin*

list_display

('id', 'created_at', 'created_by', 'label', 'notes', 'template_str', 'crawls', 'num_crawls', 'num_sn…

sort_fields

('id', 'created_at', 'created_by', 'label', 'notes', 'template_str')

search_fields

('id', 'created_by__username', 'label', 'notes', 'schedule_id', 'template_id', 'template__urls')

readonly_fields

('created_at', 'modified_at', 'crawls', 'snapshots')

autocomplete_fields

('template', 'created_by')

fieldsets

(('Schedule Info',), ('Configuration',), ('Metadata',), ('Crawls',), ('Snapshots',))

list_filter

('created_by',)

ordering

('-created_at',)

list_per_page

100

actions

('delete_selected',)

get_queryset(*request*)

change_view(*request*, *object_id*, *form_url=''*, *extra_context=None*)

add_view(*request*, *form_url=''*, *extra_context=None*)

get_fieldsets(*request*, *obj=None*)

save_model(*request*, *obj*, *form*, *change*)

template_str(*obj*)

num_crawls(*obj*)

num_snapshots(*obj*)

crawls(*obj*)

snapshots(*obj*)

archivebox.crawls.admin.**register_admin**(*admin_site*)

Package Contents

Functions

register_admin

Data

__order__

API

archivebox.crawls.**__order__**

100

archivebox.crawls.**register_admin**(*admin_site*)

archivebox.config

Minimal import-time config exports.

Submodules

archivebox.config.collection

Module Contents

Functions

_coerce_to_str_dict	Project an arbitrary config payload to flat `{UPPER_KEY: str}` form.
_load_file_config_dict	Return `(flat_dict, mtime)` for `ArchiveBox.conf` (`({}, None)` if missing).
_resolve_section_for_key	
_render_config_file_content	Render a flat config dict to INI text, grouped by inferred section.
_write_file_if_changed	Atomic-write `ArchiveBox.conf` only when contents actually differ.
mirror_machine_config_to_file	Rewrite `ArchiveBox.conf` so it mirrors `Machine.config` exactly.
_coerce_from_str_dict	Inverse of `_coerce_to_str_dict`: decode complex INI values to native.
_mirror_file_to_machine_config	Copy `ArchiveBox.conf` contents into `Machine.config`.
sync_machine_and_file	One-time-per-process reconciliation between the two stores.
write_config_file	Merge `config` into `ArchiveBox.conf`, validate, then mirror to Machine.config.

Data

CONFIG_FILE_HEADER
_MIRROR_IN_PROGRESS
_INITIAL_SYNC_DONE

API

archivebox.config.collection.**CONFIG_FILE_HEADER = <Multiline-String>**

archivebox.config.collection.**_MIRROR_IN_PROGRESS: bool**

False

archivebox.config.collection.**_INITIAL_SYNC_DONE: bool**

False

archivebox.config.collection.**_coerce_to_str_dict**(*config: Any*) → dict[str, str]

Project an arbitrary config payload to flat `{UPPER_KEY: str}` form.

INI files only round-trip strings, so we normalize Machine.config values to strings on the way out and accept the same shape on the way back. Pydantic re-coerces types at read time inside `get_config`.

Composite values (`dict` / `list` / `tuple`) are JSON-encoded so they round-trip back through pydantic-settings — `str(some_dict)` would produce Python's repr (`{'k': 'v'}` with single quotes), and pydantic-settings refuses to parse that as a `dict` field, causing e.g. `ABX_INSTALL_CACHE` to crash with `ValidationError: Input should be a valid dictionary`.

archivebox.config.collection.**_load_file_config_dict**() → tuple[dict[str, str], float | None]

Return `(flat_dict, mtime)` for `ArchiveBox.conf` (`({}, None)` if missing).

archivebox.config.collection.**_resolve_section_for_key**(*key: str*, *config_sections*, *plugin_configs*) → str

archivebox.config.collection.**_render_config_file_content**(*config: dict[str, str]*) → str

Render a flat config dict to INI text, grouped by inferred section.

archivebox.config.collection.**_write_file_if_changed**(*content: str*) → bool

Atomic-write `ArchiveBox.conf` only when contents actually differ.

Skipping unchanged writes is the difference between every Machine.save in a hot autodetection loop costing one disk write vs. zero.

archivebox.config.collection.**mirror_machine_config_to_file**(*config: Any*) → None

Rewrite `ArchiveBox.conf` so it mirrors `Machine.config` exactly.

Called from `Machine.save` after the row is committed. Recursion-guarded so the matching write_config_file -> Machine.save bounce doesn't loop.

archivebox.config.collection.**_coerce_from_str_dict**(*file_config: dict[str, str]*) → dict[str, Any]

Inverse of `_coerce_to_str_dict`: decode complex INI values to native.

`mirror_machine_config_to_file` JSON-encodes `dict` / `list` values so they round-trip through INI's string-only storage. When reading the file back into `Machine.config` (a JSONField that holds native types) those strings have to be decoded — otherwise downstream consumers like `MachineEvent` → abx-dl see a JSON string where they expect a dict and raise `TypeError`. Declared fields go through pydantic-settings' own `field_is_complex` / `prepare_field_value` so they're decoded per annotation. Undeclared keys (e.g. `ABX_INSTALL_CACHE`, written dynamically by abx-dl) are JSON-decoded when their string starts with `{` or `[` — the same shape `_coerce_to_str_dict` writes them as.

archivebox.config.collection.**_mirror_file_to_machine_config**(*file_config: dict[str, str]*) → None

Copy `ArchiveBox.conf` contents into `Machine.config`.

Internal helper used by `write_config_file` and the startup sync — callers must hold the `_MIRROR_IN_PROGRESS` guard around it.

archivebox.config.collection.**sync_machine_and_file**(*machine: Any = None*) → None

One-time-per-process reconciliation between the two stores.

Cheap on the common case where they already agree (single `stat` + dict compare 1ms). When the two sides diverge we merge them: each side's unique keys are preserved, and for keys present on both we let the newer side win (file mtime vs. `Machine.modified_at`). After the merge both stores hold the union, so every subsequent write keeps them in lockstep via the full-replace mirror functions.

Pass `machine` when the caller already has a current `Machine` instance (e.g. from `Machine.current()`) to skip the 10–15ms `get_host_guid()` round-trip on the cold path.

archivebox.config.collection.**write_config_file**(*config: dict[str, str]*) → *archivebox.misc.logging.AttrDict*

Merge `config` into `ArchiveBox.conf`, validate, then mirror to Machine.config.

Backwards-compatible signature: callers (CLI `archivebox config --set` and the init flow) pass a partial dict of keys to upsert.

archivebox.config.permissions

Module Contents

Functions

is_root_identity	
select_archivebox_user	
get_or_create_archivebox_account	
root_data_dir_handoff_paths	Return bounded top-level paths safe to hand from root to archivebox.
root_should_handoff_data_dir	Allow bounded handoff only for root or archivebox-owned collection roots.
root_parent_can_grant_group_traversal	Return whether adding archivebox traversal preserves existing parent access.
grant_archivebox_group_traversal	Let the archivebox account traverse private root-owned parents.
handoff_root_owned_data_dir	
drop_privileges	If running as root, drop privileges to the data dir owner or archivebox user.
SudoPermission	Attempt to run code with sudo permissions for a given user (or root)

Data

DATA_DIR
DEFAULT_UID
DEFAULT_GID
RUNNING_AS_UID

continues on next page

Table 235 – continued from previous page

RUNNING_AS_GID
EUID
EGID
SUDO_UID
SUDO_GID
USER
HOSTNAME
IS_ROOT
IN_DOCKER
FALLBACK_UID
FALLBACK_GID
ARCHIVEBOX_ACCOUNT
ARCHIVEBOX_USER_EXISTS
ROOT_HANDOFF_NAMES

API

archivebox.config.permissions.**is_root_identity**(*running_uid: int*, *effective_uid: int*) → bool

archivebox.config.permissions.**select_archivebox_user**(*, *running_uid: int*, *running_gid: int*, *effective_uid: int*, *effective_gid: int*, *sudo_uid: int*, *sudo_gid: int*, *data_dir_uid: int*, *data_dir_gid: int*, *account_uid: int | None*, *account_gid: int | None*, *data_dir_owner_gid: int | None = None*, *data_dir_owner_exists: bool = True*) → tuple[int, int]

archivebox.config.permissions.**DATA_DIR**
'Path(...)'

archivebox.config.permissions.**DEFAULT_UID**
911

archivebox.config.permissions.**DEFAULT_GID**
911

archivebox.config.permissions.**RUNNING_AS_UID**
'getuid(...)'

archivebox.config.permissions.**RUNNING_AS_GID**
'getgid(...)'

archivebox.config.permissions.**EUID**
'geteuid(...)'

archivebox.config.permissions.**EGID**
'getegid(...)'

archivebox.config.permissions.**SUDO_UID**
'int(...)'

archivebox.config.permissions.**SUDO_GID**
'int(...)'

archivebox.config.permissions.**USER: str**

None

archivebox.config.permissions.**HOSTNAME: str**

'cast(…)'

archivebox.config.permissions.**IS_ROOT**

'is_root_identity(…)'

archivebox.config.permissions.**IN_DOCKER**

None

archivebox.config.permissions.**FALLBACK_UID**

None

archivebox.config.permissions.**FALLBACK_GID**

None

archivebox.config.permissions.**get_or_create_archivebox_account**()

archivebox.config.permissions.**ARCHIVEBOX_ACCOUNT**

'get_or_create_archivebox_account(…)'

archivebox.config.permissions.**ARCHIVEBOX_USER_EXISTS**

False

archivebox.config.permissions.**ROOT_HANDOFF_NAMES**

('.archivebox_id', 'ArchiveBox.conf', 'index.sqlite3', 'index.sqlite3-shm', 'index.sqlite3-wal', 'ar…

archivebox.config.permissions.**root_data_dir_handoff_paths**(*data_dir: pathlib.Path*, *argv: list[str]*) → tuple[pathlib.Path, ...]

Return bounded top-level paths safe to hand from root to archivebox.

archivebox.config.permissions.**root_should_handoff_data_dir**(*, *is_root: bool*, *data_dir_uid: int*, *account_uid: int | None*, *data_dir_owner_exists: bool = True*) → bool

Allow bounded handoff only for root or archivebox-owned collection roots.

archivebox.config.permissions.**root_parent_can_grant_group_traversal**(*, *parent_uid: int*, *parent_gid: int*, *parent_mode: int*, *account_gid: int*) → bool

Return whether adding archivebox traversal preserves existing parent access.

archivebox.config.permissions.**grant_archivebox_group_traversal**(*path: pathlib.Path*) → None

Let the archivebox account traverse private root-owned parents.

archivebox.config.permissions.**handoff_root_owned_data_dir**() → None

archivebox.config.permissions.**drop_privileges**()

If running as root, drop privileges to the data dir owner or archivebox user.

archivebox.config.permissions.**SudoPermission**(*uid=0*, *fallback=False*)

Attempt to run code with sudo permissions for a given user (or root)

archivebox.config.constants

Constants are for things that never change at runtime. (but they can change from run-to-run or machine-to-machine)

DATA_DIR will never change at runtime, but you can run archivebox from inside a different DATA_DIR on the same machine.

This is loaded very early in the archivebox startup flow, so nothing in this file or imported from this file should import anything from archivebox.config.common, django, other INSTALLED_APPS, or anything else that is not in a standard library.

Module Contents

Classes

ConstantsDict

Data

CONSTANTS
CONSTANTS_CONFIG

API

class archivebox.config.constants.**ConstantsDict**

PACKAGE_DIR: pathlib.Path

None

DATA_DIR: pathlib.Path

None

ARCHIVE_DIR: pathlib.Path

None

USERS_DIR: pathlib.Path

None

MACHINE_TYPE: str

'get_machine_type(...)'

MACHINE_ID: str

'get_machine_id(...)'

COLLECTION_ID: str

'get_collection_id(...)'

VERSION: str

'detect_installed_version(...)'

IN_DOCKER: bool

None

IS_ROOT: bool

None

ARCHIVEBOX_USER: int

None

ARCHIVEBOX_GROUP: int

None

RUNNING_AS_UID: int

None

RUNNING_AS_GID: int

None

DEFAULT_UID: int

None

DEFAULT_GID: int

None

IS_INSIDE_VENV: bool

None

PACKAGE_DIR_NAME: str

None

TEMPLATES_DIR_NAME: str

'templates'

TEMPLATES_DIR: pathlib.Path

None

STATIC_DIR_NAME: str

'static'

STATIC_DIR: pathlib.Path

None

ARCHIVE_DIR_NAME: str

'archive'

USERS_DIR_NAME: str

'users'

SNAPSHOTS_DIR_NAME: str

'snapshots'

CRAWLS_DIR_NAME: str

'crawls'

SOURCES_DIR_NAME: str

'sources'

PERSONAS_DIR_NAME: str

'personas'

CACHE_DIR_NAME: str

'cache'

LOGS_DIR_NAME: str
'logs'

CUSTOM_PLUGINS_DIR_NAME: str
'custom_plugins'

CUSTOM_TEMPLATES_DIR_NAME: str
'custom_templates'

SOURCES_DIR: pathlib.Path
None

PERSONAS_DIR: pathlib.Path
None

LOGS_DIR: pathlib.Path
None

CACHE_DIR: pathlib.Path
None

CUSTOM_TEMPLATES_DIR: pathlib.Path
None

USER_PLUGINS_DIR: pathlib.Path
None

CONFIG_FILENAME: str
'ArchiveBox.conf'

SQL_INDEX_FILENAME: str
'index.sqlite3'

CONFIG_FILE: pathlib.Path
None

DATABASE_FILE: pathlib.Path
None

JSON_INDEX_FILENAME: str
'index.json'

JSONL_INDEX_FILENAME: str
'index.jsonl'

HTML_INDEX_FILENAME: str
'index.html'

ROBOTS_TXT_FILENAME: str
'robots.txt'

FAVICON_FILENAME: str
'favicon.ico'

TMP_DIR_NAME: str
'tmp'

DEFAULT_TMP_DIR: pathlib.Path
None

DEFAULT_ABXPKG_LIB_DIR: pathlib.Path

'_env_path(...)'

RESERVED_ARCHIVE_DIR_NAMES: frozenset[str]

'frozenset(...)'

TIMEZONE: str

'UTC'

DEFAULT_CLI_COLORS: dict[str, str]

None

DISABLED_CLI_COLORS: dict[str, str]

'AttrDict(...)'

MAX_HOOK_RUNTIME_SECONDS: int

None

MAX_SNAPSHOT_RUNTIME_SECONDS: int

None

ALLOWDENYLIST_REGEX_FLAGS: int

None

STATICFILE_EXTENSIONS: frozenset[str]

'frozenset(...)'

PIP_RELATED_NAMES: frozenset[str]

'frozenset(...)'

NPM_RELATED_NAMES: frozenset[str]

'frozenset(...)'

ALLOWED_IN_DATA_DIR: frozenset[str]

'frozenset(...)'

classmethod __getitem__(*key: str*)

archivebox.config.constants.**CONSTANTS**

None

archivebox.config.constants.**CONSTANTS_CONFIG**

'AttrDict(...)'

archivebox.config.views

Module Contents

Functions

_read_text_tail
is_superuser
format_parsed_datetime
get_environment_binary_url
get_installed_binary_change_url
render_binary_detail_description

continues on next page

Table 238 – continued from previous page

obj_to_yaml	
_binary_sort_key	
get_db_binaries_by_name	Group Binary rows by a URL-safe canonical name.
binaries_list_view	
binary_detail_view	
worker_list_view	
worker_detail_view	
log_list_view	
log_detail_view	

Data

ENVIRONMENT_BINARIES_BASE_URL
INSTALLED_BINARIES_BASE_URL
LOG_DETAIL_TAIL_BYTES

API

archivebox.config.views.**ENVIRONMENT_BINARIES_BASE_URL**

'/admin/environment/binaries/'

archivebox.config.views.**INSTALLED_BINARIES_BASE_URL**

'/admin/machine/binary/'

archivebox.config.views.**LOG_DETAIL_TAIL_BYTES**

100000

archivebox.config.views.**_read_text_tail**(*path: pathlib.Path*, *max_bytes: int = LOG_DETAIL_TAIL_BYTES*) → str

archivebox.config.views.**is_superuser**(*request: django.http.HttpRequest*) → bool

archivebox.config.views.**format_parsed_datetime**(*value: object*) → str

archivebox.config.views.**get_environment_binary_url**(*name: str*) → str

archivebox.config.views.**get_installed_binary_change_url**(*name: str*, *binary: archivebox.machine.models.Binary | None*) → str | None

archivebox.config.views.**render_binary_detail_description**(*name: str*, *merged: dict[str, Any]*, *db_binary: Any*) → str

archivebox.config.views.**obj_to_yaml**(*obj: Any*, *indent: int = 0*) → str

archivebox.config.views.**_binary_sort_key**(*binary: archivebox.machine.models.Binary*) → tuple[int, int, int, Any]

archivebox.config.views.**get_db_binaries_by_name**() → dict[str, *archivebox.machine.models.Binary*]

Group Binary rows by a URL-safe canonical name.

Hooks occasionally emit `BinaryEvent.name` carrying an abspath rather than a short binary name (see `services/binary_service.py`). That used to leak `name='/Users/.../bin/foo'` rows into the DB, which

then broke /admin/environment/binaries because the admin URL regex is (?P<key>[^/]+). Canonicalize at the keying step so duplicates fold into the real binary and the admin link key stays slash-free regardless of legacy DB state.

archivebox.config.views.**binaries_list_view**(*request: django.http.HttpRequest*, ***kwargs*) → admin_data_views.typing.TableContext

archivebox.config.views.**binary_detail_view**(*request: django.http.HttpRequest*, *key: str*, ***kwargs*) → admin_data_views.typing.ItemContext

archivebox.config.views.**worker_list_view**(*request: django.http.HttpRequest*, ***kwargs*) → admin_data_views.typing.TableContext

archivebox.config.views.**worker_detail_view**(*request: django.http.HttpRequest*, *key: str*, ***kwargs*) → admin_data_views.typing.ItemContext

archivebox.config.views.**log_list_view**(*request: django.http.HttpRequest*, ***kwargs*) → admin_data_views.typing.TableContext

archivebox.config.views.**log_detail_view**(*request: django.http.HttpRequest*, *key: str*, ***kwargs*) → admin_data_views.typing.ItemContext

archivebox.config.configset

Pydantic-backed config loading for ArchiveBox.

Module Contents

Classes

CaseConfigParser	
IniConfigSettingsSource	Custom settings source that reads from ArchiveBox.conf (INI format). Flattens all sections into a single namespace.
BaseConfigSet	Pure typed runtime model for config sections.

Functions

read_ini_config	Read and flatten an ArchiveBox INI config file.
decode_config_inputs	Decode string-only config values once at the boundary to typed config.

Data

COMPUTED_CONFIG_KEYS

API

archivebox.config.configset.**COMPUTED_CONFIG_KEYS**

('TERM_WIDTH', 'COMMIT_HASH', 'BUILD_TIME', 'USES_SUBDOMAIN_ROUTING', 'ENABLES_FULL_JS_REPLAY', 'CON…

class archivebox.config.configset.**CaseConfigParser**(*defaults=None*, *dict_type=_default_dict*, *allow_no_value=False*, ***, *delimiters=('=', ':')*, *comment_prefixes=('#', ';')*, *inline_comment_prefixes=None*, *strict=True*, *empty_lines_in_values=True*, *default_section=DEFAULTSECT*, *interpolation=_UNSET*, *converters=_UNSET*)

Bases: configparser.ConfigParser

optionxform(*optionstr: str*) → str

archivebox.config.configset.**read_ini_config**(*config_path: str | pathlib.Path*) → dict[str, Any]

Read and flatten an ArchiveBox INI config file.

class archivebox.config.configset.**IniConfigSettingsSource**

Bases: pydantic_settings.PydanticBaseSettingsSource

Custom settings source that reads from ArchiveBox.conf (INI format). Flattens all sections into a single namespace.

get_field_value(*field: Any*, *field_name: str*) → tuple[Any, str, bool]

__call__() → dict[str, Any]

_load_config_file() → dict[str, Any]

archivebox.config.configset.**decode_config_inputs**(*settings_cls: type[pydantic_settings.BaseSettings]*, *config: collections.abc.Mapping[str, Any]*, ***, *decode_unknown_json: bool = False*) → dict[str, Any]

Decode string-only config values once at the boundary to typed config.

class archivebox.config.configset.**BaseConfigSet**

Bases: pydantic.BaseModel

Pure typed runtime model for config sections.

Source-loading subclasses combine this model with BaseSettings and load:

1. Environment variables
2. ArchiveBox.conf file (INI format, flattened)
3. Default values

Subclasses define fields with defaults and types:

```
class ShellConfig(BaseConfigSet):
    DEBUG: bool = Field(default=False)
    USE_COLOR: bool = Field(default=True)
```

model_config

'SettingsConfigDict(...)'

computed_config_keys: ClassVar[tuple[str, ...]]

()

classmethod settings_customise_sources(*settings_cls: type[pydantic_settings.BaseSettings]*, *init_settings: pydantic_settings.PydanticBaseSettingsSource*, *env_settings: pydantic_settings.PydanticBaseSettingsSource*, *dotenv_settings: pydantic_settings.PydanticBaseSettingsSource*, *file_secret_settings: pydantic_settings.PydanticBaseSettingsSource*) → tuple[pydantic_settings.PydanticBaseSettingsSource, ...]

Define the order of settings sources (first = highest priority).

classmethod load_from_file(*config_path: pathlib.Path*) → dict[str, str]

Load config values from INI file.

__getitem__(*key: str*) → Any

__setitem__(*key: str*, *value: Any*) → None

update(**args*, ***kwargs*) → None

__contains__(*key: str*) → bool

get(*key: str*, *default: Any = None*) → Any

as_dict() → dict[str, Any]

items()

keys()

values()

archivebox.config.paths

Module Contents

Functions

_env_path	
_get_collection_id	
get_collection_id	Get a short, stable, unique ID for the current collection (e.g. abc45678)
get_machine_id	Get a machine ID, using the network identity when no OS ID is available.
get_machine_type	Get a short, stable, unique type identifier for the current machine (e.g. linux-x86_64-docker)
dir_is_writable	Check if a given directory is writable by a specific user and group (fallback=try as current user is unable to check with provided uid)
assert_dir_can_contain_unix_sockets	Check if a given directory can contain unix sockets (e.g. /tmp/supervisord.sock)
create_and_chown_dir	Create a required runtime dir and fix only that dir's ownership when needed.

continues on next page

Table 243 – continued from previous page

tmp_dir_socket_path_is_short_enough
tmp_dir_candidates
get_or_create_working_tmp_dir
get_or_create_working_lib_dir
_sql_index_location
get_data_locations
get_code_locations

Data

PACKAGE_DIR
DATA_DIR
MAX_TMP_SOCKET_URL_LENGTH
SUPERVISORD_SOCKET_FILENAME
ARCHIVE_DIR
USERS_DIR
IN_DOCKER
DATABASE_FILE

API

archivebox.config.paths.**PACKAGE_DIR: pathlib.Path**

None

archivebox.config.paths.**DATA_DIR: pathlib.Path**

'resolve(...)'

archivebox.config.paths.**MAX_TMP_SOCKET_URL_LENGTH**

90

archivebox.config.paths.**SUPERVISORD_SOCKET_FILENAME**

'supervisord.sock'

archivebox.config.paths.**_env_path**(*key: str*, *default: pathlib.Path*) → pathlib.Path

archivebox.config.paths.**ARCHIVE_DIR: pathlib.Path**

None

archivebox.config.paths.**USERS_DIR: pathlib.Path**

None

archivebox.config.paths.**IN_DOCKER**

None

archivebox.config.paths.**DATABASE_FILE**

None

archivebox.config.paths.**_get_collection_id**(*DATA_DIR=DATA_DIR*, *force_create=False*) → str

archivebox.config.paths.**get_collection_id**(*DATA_DIR=DATA_DIR*) → str

Get a short, stable, unique ID for the current collection (e.g. abc45678)

archivebox.config.paths.**get_machine_id**(*length: int = 8*) → str

Get a machine ID, using the network identity when no OS ID is available.

archivebox.config.paths.**get_machine_type**() → str

Get a short, stable, unique type identifier for the current machine (e.g. linux-x86_64-docker)

archivebox.config.paths.**dir_is_writable**(*dir_path: pathlib.Path*, *uid: int | None = None*, *gid: int | None = None*, *fallback=True*, *chown=True*) → bool

Check if a given directory is writable by a specific user and group (fallback=try as current user is unable to check with provided uid)

archivebox.config.paths.**assert_dir_can_contain_unix_sockets**(*dir_path: pathlib.Path*) → bool

Check if a given directory can contain unix sockets (e.g. /tmp/supervisord.sock)

archivebox.config.paths.**create_and_chown_dir**(*dir_path: pathlib.Path*) → None

Create a required runtime dir and fix only that dir's ownership when needed.

archivebox.config.paths.**tmp_dir_socket_path_is_short_enough**(*dir_path: pathlib.Path*) → bool

archivebox.config.paths.**tmp_dir_candidates**(*config:* archivebox.config.common.ArchiveBoxConfig) → list[pathlib.Path]

archivebox.config.paths.**get_or_create_working_tmp_dir**(*autofix=True*, *quiet=True*, *config:* ArchiveBoxConfig | *None = None*, ***config_kwargs*)

archivebox.config.paths.**get_or_create_working_lib_dir**(*autofix=True*, *quiet=False*, *config:* ArchiveBoxConfig | *None = None*, ***config_kwargs*)

archivebox.config.paths.**_sql_index_location**(*config:* archivebox.config.common.ArchiveBoxConfig) → dict

archivebox.config.paths.**get_data_locations**(*config:* ArchiveBoxConfig | *None = None*, ***config_kwargs*)

archivebox.config.paths.**get_code_locations**(*config:* ArchiveBoxConfig | *None = None*, ***config_kwargs*)

archivebox.config.django

Module Contents

Functions

setup_django

Data

CONFIG
STDERR
DJANGO_SET_UP

API

archivebox.config.django.**CONFIG**

'get_config(...)'

archivebox.config.django.**STDERR**

'Console(...)'

archivebox.config.django.**DJANGO_SET_UP**

False

archivebox.config.django.**setup_django**(*check_db=False*) → None

archivebox.config.common

Module Contents

Classes

ShellConfig	
StorageConfig	
GeneralConfig	
ServerConfig	
DatabaseConfig	
ArchivingConfig	
SearchBackendConfig	
ArchiveBoxBaseConfig	Merged, typed ArchiveBox config.
ArchiveBoxSourceSettings	Environment and INI source loader for the pure runtime config model.

Functions

_is_cookie_file_config_key	
_resolve_cookie_file_paths	Keep collection-relative auth paths valid after hooks change cwd.
_legacy_bool	
permissions_from_legacy_public_flags	
base_url_from_legacy_server_config	Derive the current canonical URL from the pre-0.9 server settings.
resolve_delete_after_config_value	
_plugin_sensitive_config_keys	
is_sensitive_config_key	True if a config key names a credential and must be write-only in the UI.
redact_sensitive_config	Return a copy of `config` with credential values replaced by ********.
normalize_runtime_config	Return config filtered for runtime/frozen usage, optionally JSON-safe.
build_crawl_config_snapshot	Build the frozen crawl config stored on Crawl.config at creation time.
rprint	
_default_sqlite_journal_mode	

continues on next page

Table 248 – continued from previous page

parse_delete_after	
_plugin_user_config_value	
_plugin_user_config	
_plugin_input_config	Return raw config inputs understood by core or plugin schemas.
_explicit_plugin_enabled_keys	Return canonical *_ENABLED keys explicitly provided by the user.
_discover_plugin_config_schemas	
_plugin_config_properties	
_plugin_config_model	
_archivebox_config_input_names	
_build_archivebox_config_model	
_normalize_plugins_config_value	
_plugin_enabled_config_keys	
get_live_config_url	
config_field_metadata	Return one centralized metadata map for core and plugin config fields.
find_config_section	
find_config_default	
config_field_type	
find_config_type	
find_config_source	Determine where a config value comes from.
get_request_config	Return the per-request ArchiveBox config, upgrading to plugin resolution if needed.
get_config	Get merged config from all sources.
get_all_configs	Get all config section objects as a dictionary.

Data

ConfigOverrides
ConfigPayload
PluginSchemaDocuments
LIVE_CONFIG_BASE_URL
_STDOUT_CONSOLE
_STDERR_CONSOLE
_WARNED_ARCHIVING_CONFIGS
_SENSITIVE_CONFIG_KEY_NEEDLES
SENSITIVE_CONFIG_VALUE_REDACTED
_SCOPE_CRAWL_FROZEN
_SCOPE_CRAWL_EXECUTION
_SCOPE_SERVER
PLUGIN_CONFIG_SCHEMAS
ArchiveBoxConfig

API

archivebox.config.common.**ConfigOverrides**

None

archivebox.config.common.**ConfigPayload**

None

archivebox.config.common.**PluginSchemaDocuments**

None

archivebox.config.common.**LIVE_CONFIG_BASE_URL**

'/admin/environment/config/'

archivebox.config.common.**_is_cookie_file_config_key**(*key: str*) → bool

archivebox.config.common.**_resolve_cookie_file_paths**(*config: dict[str, Any]*) → dict[str, Any]

Keep collection-relative auth paths valid after hooks change cwd.

archivebox.config.common.**_STDOUT_CONSOLE**

'Console(...)'

archivebox.config.common.**_STDERR_CONSOLE**

'Console(...)'

archivebox.config.common.**_WARNED_ARCHIVING_CONFIGS: set[tuple[int, bool]]**

'set(...)'

archivebox.config.common.**_legacy_bool**(*value: object*) → bool | None

archivebox.config.common.**permissions_from_legacy_public_flags**(*raw_config: collections.abc.Mapping[str, object]*) → str | None

archivebox.config.common.**base_url_from_legacy_server_config**(*raw_config: collections.abc.Mapping[str, object]*) → str

Derive the current canonical URL from the pre-0.9 server settings.

archivebox.config.common.**resolve_delete_after_config_value**(**configs: collections.abc.Mapping[str, Any] | None*) → str

archivebox.config.common.**_SENSITIVE_CONFIG_KEY_NEEDLES**

('TOKEN', 'SECRET', 'API_KEY', 'APIKEY', 'PASSWORD')

archivebox.config.common.**SENSITIVE_CONFIG_VALUE_REDACTED**

'********'

archivebox.config.common.**_SCOPE_CRAWL_FROZEN**

'crawl_frozen'

archivebox.config.common.**_SCOPE_CRAWL_EXECUTION**

'crawl_execution'

archivebox.config.common.**_SCOPE_SERVER**

'server'

archivebox.config.common.**_plugin_sensitive_config_keys**() → frozenset[str]

archivebox.config.common.**is_sensitive_config_key**(*key: str*) → bool

True if a config key names a credential and must be write-only in the UI.

Matches any key whose uppercase form contains `TOKEN`, `SECRET`, `API_KEY`, `APIKEY`, or `PASSWORD` — covers `SECRET_KEY`, `OPENAI_API_KEY`, `TWOCAPTCHA_APIKEY`, `GITHUB_TOKEN`, `ADMIN_PASSWORD`, etc. Centralized here so the KeyValueWidget (Machine/Crawl/Snapshot/Persona admin forms), the plugin config grid, REST API responses, and any future surface that round-trips raw config values all agree on which keys to redact.

archivebox.config.common.**redact_sensitive_config**(*config: collections.abc.Mapping[str, Any] | None*) → dict[str, Any]

Return a copy of `config` with credential values replaced by `********`.

Used wherever a config dict crosses an API/export/debug-dump boundary. The widget-side write-only treatment handles the form-render path; this helper handles every JSON-response path (REST schemas, `to_json` exports, admin debug views, etc.). Empty values are passed through unchanged so callers can still tell "unset" from "set-but-hidden."

archivebox.config.common.**normalize_runtime_config**(*config:* archivebox.config.configset.BaseConfigSet *| collections.abc.Mapping[str, Any] | str | None*, ***, *only_crawl_execution: bool = False*, *exclude_runtime_derived: bool = False*, *exclude_crawl_execution: bool = False*, *json_safe: bool = True*) → dict[str, Any]

Return config filtered for runtime/frozen usage, optionally JSON-safe.

archivebox.config.common.**build_crawl_config_snapshot**(***, *persona: Any = None*, *overrides: collections.abc.Mapping[str, Any] | None = None*, *base_config:* ArchiveBoxBaseConfig *| collections.abc.Mapping[str, object] | None = None*) → dict[str, Any]

Build the frozen crawl config stored on Crawl.config at creation time.

archivebox.config.common.**rprint**(**args*, *file=None*, ***kwargs*)

class archivebox.config.common.**ShellConfig**

Bases: *archivebox.config.configset.BaseConfigSet*

toml_section_header: str

'SHELL_CONFIG'

_scope: str

'PrivateAttr(…)'

DEBUG: bool

'Field(…)'

IS_TTY: bool

'Field(…)'

USE_COLOR: bool

'Field(…)'

SHOW_PROGRESS: bool

'Field(…)'

IN_DOCKER: bool

'Field(…)'

IN_QEMU: bool

'Field(…)'

ANSI: dict[str, str]

'Field(…)'

property TERM_WIDTH: int

property COMMIT_HASH: str | None

property BUILD_TIME: str

class archivebox.config.common.StorageConfig

Bases: archivebox.config.configset.BaseConfigSet

toml_section_header: str

'STORAGE_CONFIG'

_scope: str

'PrivateAttr(...)'

TMP_DIR: pathlib.Path

'Field(...)'

ABXPKG_LIB_DIR: pathlib.Path

'Field(...)'

OUTPUT_PERMISSIONS: str

'Field(...)'

ENFORCE_ATOMIC_WRITES: bool

'Field(...)'

ALLOW_NO_UNIX_SOCKETS: bool

'Field(...)'

class archivebox.config.common.GeneralConfig

Bases: archivebox.config.configset.BaseConfigSet

toml_section_header: str

'GENERAL_CONFIG'

_scope: str

'PrivateAttr(...)'

TAG_SEPARATOR_PATTERN: str

'Field(...)'

class archivebox.config.common.ServerConfig

Bases: archivebox.config.configset.BaseConfigSet

toml_section_header: str

'SERVER_CONFIG'

_scope: str

'PrivateAttr(...)'

SERVER_SECURITY_MODES: ClassVar[tuple[str, ...]]

('auto', 'safe-subdomains-fullreplay', 'safe-onedomain-nojsreplay', 'unsafe-onedomain-noadmin', 'dan...

SECRET_KEY: str

'Field(...)'

BIND_ADDR: str

'Field(...)'

BASE_URL: str

'Field(...)'

ALLOWED_HOSTS: str

'Field(...)'

CSRF_TRUSTED_ORIGINS: str

'Field(...)'

SERVER_SECURITY_MODE: str

'Field(...)'

SNAPSHOTS_PER_PAGE: int

'Field(...)'

FOOTER_INFO: str

'Field(...)'

PUBLIC_INDEX: bool

'Field(...)'

PUBLIC_ADD_VIEW: bool

'Field(...)'

ADMIN_USERNAME: str | None

'Field(...)'

ADMIN_PASSWORD: str | None

'Field(...)'

REVERSE_PROXY_USER_HEADER: str

'Field(...)'

REVERSE_PROXY_WHITELIST: str

'Field(...)'

LOGOUT_REDIRECT_URL: str

'Field(...)'

validate_server_security_mode(*v: str*) → str

property USES_SUBDOMAIN_ROUTING: bool

property ENABLES_FULL_JS_REPLAY: bool

property CONTROL_PLANE_ENABLED: bool

property BLOCK_UNSAFE_METHODS: bool

property SHOULD_NEUTER_RISKY_REPLAY: bool

property IS_UNSAFE_MODE: bool

property IS_DANGEROUS_MODE: bool

property IS_LOWER_SECURITY_MODE: bool

archivebox.config.common.**_default_sqlite_journal_mode**(*database_name: str*) → str

class archivebox.config.common.**DatabaseConfig**

Bases: *archivebox.config.configset.BaseConfigSet*

toml_section_header: str

'DATABASE_CONFIG'

_scope: str

'PrivateAttr(...)'

DATABASE_ENGINE: str

'Field(...)'

DATABASE_NAME: str

'Field(...)'

DATABASE_HOST: str

'Field(...)'

DATABASE_PORT: int

'Field(...)'

DATABASE_USER: str

'Field(...)'

DATABASE_PASSWORD: str

'Field(...)'

SQLITE_JOURNAL_MODE: str

'Field(...)'

SQLITE_MMAP_SIZE: int

'Field(...)'

SQLITE_BUSY_TIMEOUT: int

'Field(...)'

SQLITE_LOCK_RETRY_TIMEOUT: float

'Field(...)'

SQLITE_LOCK_RETRY_INTERVAL: float

'Field(...)'

reject_shared_filesystem_sqlite_wal()

class archivebox.config.common.**ArchivingConfig**

Bases: *archivebox.config.configset.BaseConfigSet*

toml_section_header: str

'ARCHIVING_CONFIG'

_scope: str

'PrivateAttr(...)'

PLUGINS: str

'Field(...)'

PARSER: str

'Field(...)'

ONLY_NEW: bool

'Field(...)'

INDEX_ONLY: bool

'Field(...)'

TIMEOUT: int

'Field(...)'

CRAWL_MAX_URLS: int

'Field(...)'

CRAWL_MAX_SIZE: int

'Field(...)'

CRAWL_TIMEOUT: int

'Field(...)'

CRAWL_MAX_CONCURRENT_SNAPSHOTS: int

'Field(...)'

SNAPSHOT_MAX_SIZE: int

'Field(...)'

RESOLUTION: str

'Field(...)'

CHECK_SSL_VALIDITY: bool

'Field(...)'

USER_AGENT: str

'Field(...)'

COOKIES_FILE: pathlib.Path | None

'Field(...)'

AUTH_STORAGE_FILE: pathlib.Path | None

'Field(...)'

classmethod empty_cookie_file_is_unset(*value*)

URL_DENYLIST: str

'Field(...)'

URL_ALLOWLIST: str | None

'Field(...)'

DEFAULT_PERSONA: str

'Field(...)'

PERMISSIONS: str

'Field(...)'

DELETE_AFTER: str

'Field(...)'

warn_if_invalid() → None

validate_check_ssl_validity(*v*)

SIDE EFFECT: disable "you really shouldnt disable ssl" warnings emitted by requests

classmethod validate_delete_after(*value*)

classmethod validate_permissions(*value*)

property URL_ALLOWLIST_PTN: re.Pattern | None

property URL_DENYLIST_PTN: re.Pattern

archivebox.config.common.**parse_delete_after**(*value*) → datetime.timedelta | None

class archivebox.config.common.**SearchBackendConfig**

Bases: *archivebox.config.configset.BaseConfigSet*

toml_section_header: str

'SEARCH_BACKEND_CONFIG'

_scope: str

'PrivateAttr(...)'

SEARCH_BACKEND_ENGINE: str

'Field(...)'

archivebox.config.common.**_plugin_user_config_value**(*value: Any*) → str

archivebox.config.common.**_plugin_user_config**(*config: collections.abc.Mapping[str, object]*) → dict[str, str]

archivebox.config.common.**_plugin_input_config**(*config: collections.abc.Mapping[str, object]*) → dict[str, object]

Return raw config inputs understood by core or plugin schemas.

archivebox.config.common.**_explicit_plugin_enabled_keys**(*config: collections.abc.Mapping[str, object]*) → set[str]

Return canonical *_ENABLED keys explicitly provided by the user.

PLUGINS is a selector, not an override for concrete plugin enable flags. If a user passes PLUGINS=wget and SAVE_WGET=False together, the alias-resolved WGET_ENABLED=False must survive the model validator that expands PLUGINS into concrete booleans.

archivebox.config.common.**_discover_plugin_config_schemas**() → *archivebox.config.common.PluginSchemaDocuments*

archivebox.config.common.**_plugin_config_properties**(*plugin_schemas:* archivebox.config.common.PluginSchemaDocuments) → dict[str, dict[str, Any]]

archivebox.config.common.**_plugin_config_model**(*plugin_schemas:* archivebox.config.common.PluginSchemaDocuments) → *type*[pydantic.BaseModel]

archivebox.config.common.**_archivebox_config_input_names**() → set[str]

class archivebox.config.common.**ArchiveBoxBaseConfig**

Bases: *archivebox.config.common.ShellConfig*, *archivebox.config.common.StorageConfig*, *archivebox.config.common.GeneralConfig*, *archivebox.config.common.ServerConfig*, *archivebox.config.common.DatabaseConfig*, *archivebox.config.common.ArchivingConfig*, *archivebox.config.common.SearchBackendConfig*, *archivebox.config.ldap.LDAPConfig*

Merged, typed ArchiveBox config.

Core ArchiveBox fields are declared above. Plugin-owned fields are added to the concrete ArchiveBoxConfig model from plugin JSONSchema below, so ArchiveBox does not hardcode any individual plugin config names.

model_config

'SettingsConfigDict(...)'

computed_config_keys: ClassVar[tuple[str, ...]]

None

classmethod _core_config_classes() → tuple[*type*[*archivebox.config.configset.BaseConfigSet*], ...]

classmethod _core_field_scope(*key: str*) → str | None

classmethod _plugin_field_scope(*key: str*) → str | None

classmethod scope_for_key(*key: str*) → str

classmethod _scope_by_key() → dict[str, str]

classmethod _crawl_frozen_keys() → frozenset[str]

classmethod _crawl_runtime_keys() → frozenset[str]

classmethod runtime_derived_config_keys() → frozenset[str]

_scoped_config(*, *include_execution: bool*) → dict[str, Any]

for_crawl() → dict[str, Any]

Config scoped to crawl execution, without runtime object overlays.

for_crawl_frozen(*, *persona: Any = None*) → dict[str, Any]

Config safe to persist permanently on Crawl.config.

for_crawl_runtime(*, *crawl: Any = None*, *snapshot: Any = None*, *persona: Any = None*, *runtime_overrides: collections.abc.Mapping[str, Any] | None = None*, *extra_context: collections.abc.Mapping[str, Any] | None = None*, *crawl_output_dir: Any = None*, *snapshot_output_dir: Any = None*) → dict[str, Any]

Config payload safe to pass to crawl/snapshot hook execution.

resolve_runtime_paths()

derive_plugin_enabled_config()

_derive_plugin_enabled_config(*, *respect_current_enabled: bool*) → None

archivebox.config.common.**_build_archivebox_config_model**(*plugin_schemas:* archivebox.config.common.PluginSchemaDocuments) → *type*[*archivebox.config.common.ArchiveBoxBaseConfig*]

archivebox.config.common.**PLUGIN_CONFIG_SCHEMAS**

'_discover_plugin_config_schemas(...)'

archivebox.config.common.**ArchiveBoxConfig**

'_build_archivebox_config_model(...)'

class archivebox.config.common.**ArchiveBoxSourceSettings**

Bases: *archivebox.config.common.ArchiveBoxConfig*, pydantic_settings.BaseSettings

Environment and INI source loader for the pure runtime config model.

model_config

'SettingsConfigDict(...)'

archivebox.config.common.**_normalize_plugins_config_value**(*value: Any*) → set[str]

archivebox.config.common.**_plugin_enabled_config_keys**() → dict[str, str]

archivebox.config.common.**get_live_config_url**(*key: str*) → str

archivebox.config.common.**config_field_metadata**() → dict[str, dict[str, Any]]

Return one centralized metadata map for core and plugin config fields.

archivebox.config.common.**find_config_section**(*key: str*) → str

archivebox.config.common.**find_config_default**(*key: str*) → str

archivebox.config.common.**config_field_type**(*key: str*) → str

archivebox.config.common.**find_config_type**(*key: str*) → str

archivebox.config.common.**find_config_source**(*key: str*, *merged_config: collections.abc.Mapping[str, Any]*) → str

Determine where a config value comes from.

archivebox.config.common.**get_request_config**(*request: Any*, ***, *resolve_plugins: bool = False*) → *archivebox.config.common.ArchiveBoxBaseConfig*

Return the per-request ArchiveBox config, upgrading to plugin resolution if needed.

archivebox.config.common.**get_config**(*defaults:* archivebox.config.common.ConfigOverrides | *None = None*, *overrides:* archivebox.config.common.ConfigOverrides | *None = None*, *base_config:* archivebox.config.common.ArchiveBoxBaseConfig | *collections.abc.Mapping[str, object] | None = None*, *persona: Any = None*, *crawl: Any = None*, *snapshot: Any = None*, *machine: Any = None*, *include_machine: bool = True*, *resolve_plugins: bool = True*, *redact_sensitive: bool = False*) → *archivebox.config.common.ArchiveBoxBaseConfig*

Get merged config from all sources.

Defaults are hydrated by pydantic from core/plugin defaults, ArchiveBox.conf, and environment variables. Persisted Machine/Persona values then apply for live crawl-execution scope, while Crawl/Snapshot rows apply their frozen crawl-scope values. Explicit overrides win last.

Crawl-execution config is not persisted on Crawl.config. It is rederived from current Machine/Persona state and hydrated process defaults each time.

archivebox.config.common.**get_all_configs**() → dict[str, *archivebox.config.configset.BaseConfigSet*]

Get all config section objects as a dictionary.

archivebox.config.ldap

Module Contents

Classes

LDAPConfig	LDAP authentication configuration.

API

class archivebox.config.ldap.**LDAPConfig**

Bases: *archivebox.config.configset.BaseConfigSet*

LDAP authentication configuration.

Only loads and validates if django-auth-ldap is installed. These settings integrate with Django's LDAP authentication backend.

toml_section_header: str

'LDAP_CONFIG'

_scope: str

'PrivateAttr(...)'

LDAP_ENABLED: bool

'Field(...)'

LDAP_SERVER_URI: str | None

'Field(...)'

LDAP_BIND_DN: str | None

'Field(...)'

LDAP_BIND_PASSWORD: str | None

'Field(...)'

LDAP_USER_BASE: str | None

'Field(...)'

LDAP_USER_FILTER: str

'Field(...)'

LDAP_USERNAME_ATTR: str

'Field(...)'

LDAP_FIRSTNAME_ATTR: str

'Field(...)'

LDAP_LASTNAME_ATTR: str

'Field(...)'

LDAP_EMAIL_ATTR: str

'Field(...)'

LDAP_CREATE_SUPERUSER: bool

'Field(...)'

validate_ldap_config() → tuple[bool, str]

Validate that all required LDAP settings are configured.

Returns: Tuple of (is_valid, error_message)

archivebox.config.version

Module Contents

Functions

detect_installed_version	Autodetect the installed archivebox version by using pip package metadata, pyproject.toml file, or package.json file
get_COMMIT_HASH	
get_BUILD_TIME	

Data

IN_DOCKER
PACKAGE_DIR
VERSION

API

archivebox.config.version.**IN_DOCKER**

None

archivebox.config.version.**PACKAGE_DIR: pathlib.Path**

None

archivebox.config.version.**detect_installed_version**(*PACKAGE_DIR: pathlib.Path = PACKAGE_DIR*)

Autodetect the installed archivebox version by using pip package metadata, pyproject.toml file, or package.json file

archivebox.config.version.**get_COMMIT_HASH**() → str | None

archivebox.config.version.**get_BUILD_TIME**() → str

archivebox.config.version.**VERSION: str**

'detect_installed_version(...)'

Package Contents

Functions

__getattr__

Data

__order__
__all__

API

archivebox.config.**__order__**

200

archivebox.config.**__getattr__**(*name: str*)

archivebox.config.**__all__**

('CONSTANTS', 'CONSTANTS_CONFIG', 'PACKAGE_DIR', 'DATA_DIR', 'VERSION')

archivebox.base_models

Submodules

archivebox.base_models.apps

archivebox.base_models.models

Base models using UUIDv7 for all id fields.

Module Contents

Classes

PagedQuerySet	
AutoDateTimeField	DateTimeField that automatically updates on save (legacy compatibility).
ModelWithUUID	
ModelWithNotes	Mixin for models with a notes field.
ModelWithHealthStats	Mixin for models with health tracking fields.
ModelWithConfig	Mixin for models with a JSON config field.
ModelWithDeleteAfter	
ModelWithOutputDir	

Functions

normalize_config_json_values
get_or_create_system_user_pk

API

class archivebox.base_models.models.**PagedQuerySet**

Bases: `django.db.models.QuerySet`

paged_iterator(*chunk_size: int = 500*)

Iterate rows using bounded keyset pages instead of one streaming cursor.

Django's iterator(chunk_size=...) still keeps a single SQLite SELECT cursor open until the full queryset is exhausted. That is fine for read-only exports, but update/migration code does filesystem work and writes while iterating; a long-lived read cursor there can stretch lock waits across thousands of rows. This respects the queryset's existing filters, order_by(), select_related(), and prefetch_related() state; if no ordering is defined, it falls back to primary-key order.

archivebox.base_models.models.**normalize_config_json_values**(*config: Any*) → Any

archivebox.base_models.models.**get_or_create_system_user_pk**(*username='system'*)

class archivebox.base_models.models.**AutoDateTimeField**

Bases: django.db.models.DateTimeField

DateTimeField that automatically updates on save (legacy compatibility).

pre_save(*model_instance*, *add*)

class archivebox.base_models.models.**ModelWithUUID**

Bases: django.db.models.Model

id

'CompactUUIDField(...)'

created_at

'DateTimeField(...)'

modified_at

'DateTimeField(...)'

created_by

'ForeignKey(...)'

class Meta

Bases: django_stubs_ext.db.models.TypedModelMeta

abstract

True

__str__() → str

property admin_change_url: str

property api_url: str

property api_docs_url: str

class archivebox.base_models.models.**ModelWithNotes**

Bases: django.db.models.Model

Mixin for models with a notes field.

notes

'TextField(...)'

class Meta

Bases: django_stubs_ext.db.models.TypedModelMeta

abstract

True

class archivebox.base_models.models.**ModelWithHealthStats**

Bases: django.db.models.Model

Mixin for models with health tracking fields.

num_uses_failed

'PositiveIntegerField(...)'

num_uses_succeeded

'PositiveIntegerField(...)'

class Meta

Bases: django_stubs_ext.db.models.TypedModelMeta

abstract

True

property admin_change_url: str

property health: int

increment_health_stats(*success: bool*)

Atomically increment success or failure counter using F() expression.

class archivebox.base_models.models.**ModelWithConfig**

Bases: django.db.models.Model

Mixin for models with a JSON config field.

config

'JSONField(...)'

class Meta

Bases: django_stubs_ext.db.models.TypedModelMeta

abstract

True

save(**args*, ***kwargs*)

class archivebox.base_models.models.**ModelWithDeleteAfter**

Bases: django.db.models.Model

delete_after_final_statuses: tuple[str, ...]

()

delete_at

'DateTimeField(...)'

class Meta

Bases: django_stubs_ext.db.models.TypedModelMeta

abstract

True

save(**args*, ***kwargs*)

get_delete_after_config_value()

set_delete_at_from_config(*config_value=None*) → bool

classmethod missing_delete_at_candidates()

classmethod delete_expired(*, *batch_size: int = 100*, *backfill_missing: bool = True*, *backfill_cursors: dict | None = None*) → int

class archivebox.base_models.models.**ModelWithOutputDir**

Bases: *archivebox.base_models.models.ModelWithUUID*

class Meta

Bases: *archivebox.base_models.models.ModelWithUUID*

abstract

True

_delete_signal_registered

False

property output_dir_parent: str

property output_dir_name: str

property output_dir_str: str

abstract property output_dir: pathlib.Path

output_paths_for_delete() → tuple[pathlib.Path, ...]

classmethod validate_output_paths_for_delete(*paths*) → tuple[pathlib.Path, ...]

classmethod delete_output_paths(*paths*) → None

schedule_delete_cleanup(*, *using: str | None = None*) → None

Capture output paths before DB deletion and remove them after commit.

classmethod register_delete_signal() → None

archivebox.base_models.admin

Base admin classes for models using UUIDv7.

Module Contents

Classes

HexUUIDConverter	URL path converter that canonicalizes UUIDs to their 32-char hex form.
ConfigOption	
KeyValueWidget	A widget that renders JSON dict as editable key-value input fields with + and - buttons to add/remove rows. Includes autocomplete for available config keys from the plugin system.
ConfigEditorMixin	Mixin for admin classes with a config JSON field.

continues on next page

Table 257 – continued from previous page

BaseModelAdmin

API

class archivebox.base_models.admin.**HexUUIDConverter**

URL path converter that canonicalizes UUIDs to their 32-char hex form.

Accepts both the hyphenated (`aaaaaaaa-bbbb-...`) and bare-hex (`aaaaaaaabbbb...`) UUID strings on the way in (Django's UUIDField parses either), but `to_url` always emits the bare-hex form. This is what makes `reverse("admin:app_model_change", args=[obj.pk])` produce `/admin/app/model/06a1a8facb0d.../change/` instead of the default hyphenated rendering — admin links throughout the app reverse through this converter once `BaseModelAdmin.get_urls` swaps in `<hexuuid:object_id>` below.

regex

'[0-9a-fA-F]{32}|[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}'

to_python(*value: str*) → str

to_url(*value*) → str

class archivebox.base_models.admin.**ConfigOption**

Bases: `typing.TypedDict`

plugin: str

None

type: str | list[str]

None

default: object

None

description: str

None

enum: NotRequired[list[object]]

None

pattern: NotRequired[str]

None

minimum: NotRequired[int | float]

None

maximum: NotRequired[int | float]

None

class archivebox.base_models.admin.**KeyValueWidget**

Bases: `django.forms.Widget`

A widget that renders JSON dict as editable key-value input fields with + and - buttons to add/remove rows. Includes autocomplete for available config keys from the plugin system.

template_name = <Multiline-String>

class Media

css: ClassVar[dict[str, list[str]]]

None

js: ClassVar[list[str]]

[]

_get_config_options() → dict[str, *archivebox.base_models.admin.ConfigOption*]

Get available config options from plugins.

_parse_value(*value: object*) → dict[str, object]

render(*name: str*, *value: object*, *attrs: collections.abc.Mapping[str, str] | None = None*, *renderer: django.forms.renderers.BaseRenderer | None = None*) → django.utils.safestring.SafeString

_render_row(*widget_id: str*, *key: str*, *value: str*) → str

_escape(*s: object*) → str

Escape HTML special chars in attribute values.

value_from_datadict(*data: django.http.QueryDict | collections.abc.Mapping[str, object]*, *files: object*, *name: str*) → str

class archivebox.base_models.admin.**ConfigEditorMixin**

Bases: django.contrib.admin.ModelAdmin

Mixin for admin classes with a config JSON field.

Provides a key-value editor widget with autocomplete for available config keys.

formfield_for_dbfield(*db_field: django.db.models.Field*, *request: django.http.HttpRequest*, ***kwargs: object*) → django.forms.Field | None

Use KeyValueWidget for the config JSON field.

save_model(*request: django.http.HttpRequest*, *obj*, *form*, *change*)

Preserve write-only redacted credentials on save.

The KeyValueWidget renders sensitive keys (*TOKEN*, *SECRET*, *API_KEY*, *APIKEY*) with an empty value + password input — the real value never leaves the server. On submit, an empty value for a sensitive key that was previously set means "leave untouched", not "clear it." We honor that here by re-merging the stored value before the row is written. Explicitly removing the row in the UI still clears it (the key is gone from the submitted JSON, so there's nothing to merge over).

class archivebox.base_models.admin.**BaseModelAdmin**

Bases: django_object_actions.DjangoObjectActions, django.contrib.admin.ModelAdmin

list_display

('id', 'created_at', 'created_by')

readonly_fields

('id', 'created_at', 'modified_at')

show_search_mode_selector

False

change_form_template

'admin/archivebox_change_form.html'

get_admin_toolbar_actions(*request*, *obj*)

Return extra action button dicts for the shared change-form toolbar.

See `templates/admin/includes/archivebox_toolbar.html` for the accepted keys. Default: no extras (toolbar renders Save/History/Delete plus any `django-object-actions` change_actions).

render_change_form(*request*, *context*, *add=False*, *change=False*, *form_url=''*, *obj=None*)

get_default_search_mode() → str

get_form(*request: django.http.HttpRequest*, *obj: django.db.models.Model | None = None*, *change: bool = False*, ***kwargs: object*)

get_urls()

Swap the per-object admin URLs from <path:object_id> to <hexuuid:object_id> so canonical change/delete/history URLs use the 32-char hex form. The hyphenated form still resolves because the converter's regex accepts both — Django reverses through `to_url` which always emits hex, so links in templates / changelists / inline formsets all canonicalize automatically.

Non-UUID PKs (an `IntegerField` PK on some legacy table, for example) won't match the converter's regex and fall back to the default <path:object_id> patterns we still include after our swap.

archivebox.services

Submodules

archivebox.services.process_service

Module Contents

Classes

ProcessService

Functions

parse_event_datetime
current_network_interface_with_machine
normalize_process_env

API

archivebox.services.process_service.**parse_event_datetime**(*value: str | None*)

archivebox.services.process_service.**current_network_interface_with_machine**()

archivebox.services.process_service.**normalize_process_env**(*env: dict*) → dict

class archivebox.services.process_service.**ProcessService**(*bus*)

Bases: `abx_dl.services.base.BaseService`

Initialization

LISTENS_TO: ClassVar[list[*type*[abxbus.BaseEvent]]]

None

EMITS: ClassVar[list[*type*[abxbus.BaseEvent]]]

[]

async current_iface()

async on_ProcessStartedEvent__save_to_db(*event: abx_dl.events.ProcessStartedEvent*) → None

async _completed_worker_loop() → None

_ensure_completed_worker() → None

async on_ProcessCompletedEvent__save_to_db(*event: abx_dl.events.ProcessCompletedEvent*) → None

async flush_completed() → None

async on_CrawlCleanupEvent__flush_completed(*event: abx_dl.events.CrawlCleanupEvent*) → None

async on_CrawlCompletedEvent__flush_completed(*event: abx_dl.events.CrawlCompletedEvent*) → None

async _save_completed_process_to_db(*event: abx_dl.events.ProcessCompletedEvent*) → None

archivebox.services.binary_service

Module Contents

Classes

ArchiveBoxBinaryService	Preserve ArchiveBox's legacy Binary Process rows around abxpkg requests.

Functions

project_abxpkg_derived_cache_to_db	Mirror abxpkg subprocess-resolved binaries into ArchiveBox's DB.
_provider_names	
_binproviders_to_str	
_persisted_overrides_for_request	

API

class archivebox.services.binary_service.**ArchiveBoxBinaryService**(*bus: abxbus.EventBus*)

Bases: abx_dl.services.base.BaseService

Preserve ArchiveBox's legacy Binary Process rows around abxpkg requests.

Initialization

LISTENS_TO

None

EMITS: list[*type*[abxbus.BaseEvent]]

[]

async on_BinaryRequestEvent__project_process(*request: abxpkg.binary_service.BinaryRequestEvent*) → None

async on_BinaryEvent__finalize_process(*event: abxpkg.binary_service.BinaryEvent*) → None

async _get_or_create_binary(*machine*, *binary_name: str*, *request: abxpkg.binary_service.BinaryRequestEvent*)

_process_cmd(*request: abxpkg.binary_service.BinaryRequestEvent*) → list[str]

_binary_event_json(*event: abxpkg.binary_service.BinaryEvent*, *binary*) → dict[str, Any]

async _finalize_missing_process(*request: abxpkg.binary_service.BinaryRequestEvent*) → None

async _finalize_request_when_done(*request: abxpkg.binary_service.BinaryRequestEvent*) → None

_schedule_missing_finalize(*request: abxpkg.binary_service.BinaryRequestEvent*) → None

async flush_missing_finalizers() → None

async on_BinaryRequestEvent__schedule_missing_finalize(*request: abxpkg.binary_service.BinaryRequestEvent*) → None

_process_output_dir(*binary*, *request: abxpkg.binary_service.BinaryRequestEvent*) → pathlib.Path

_write_binary_index(*binary*, *process*, *output_dir: pathlib.Path*) → None

archivebox.services.binary_service.**project_abxpkg_derived_cache_to_db**(*lib_dir: pathlib.Path | str | None*) → None

Mirror abxpkg subprocess-resolved binaries into ArchiveBox's DB.

Hook shebangs resolve binaries through the abxpkg CLI in subprocesses, so those resolutions cannot emit in-process BinaryRequestEvent/BinaryEvent objects on the ArchiveBox runner bus. abxpkg's generic cross-process projection point is `LIB_DIR/env/derived.env`; ArchiveBox imports those resolved records here after hook execution.

archivebox.services.binary_service.**_provider_names**(*binproviders: str | list[str] | None*) → list[str]

archivebox.services.binary_service.**_binproviders_to_str**(*binproviders: str | list[str] | None*) → str

archivebox.services.binary_service.**_persisted_overrides_for_request**(*request: abxpkg.binary_service.BinaryRequestEvent | None*) → dict[str, Any]

archivebox.services.archive_result_service

Module Contents

Classes

ArchiveResultService	Project abx-dl ArchiveResult facts into Django models.

Functions

_perf_trace	
_perf_span	
_manifest_metadata	
_resolve_output_metadata	
_normalize_status	
_normalize_snapshot_title	
_extract_snapshot_title	
_should_update_snapshot_title	
_save_archiveresult_event_to_db	Project one ArchiveResultEvent with a single thread-sensitive ORM hop.
mark_archiveresult_started	Project a running abx-dl hook after its OS process is persisted.

API

archivebox.services.archive_result_service.**_perf_trace**(*label*)

archivebox.services.archive_result_service.**_perf_span**(*label: str*)

archivebox.services.archive_result_service.**_manifest_metadata**(*manifest: abx_dl.output_files.OutputManifest*) → tuple[dict[str, dict], int, str]

archivebox.services.archive_result_service.**_resolve_output_metadata**(*raw_output_files: Any*, *plugin_dir: pathlib.Path*) → tuple[dict[str, dict], int, str]

archivebox.services.archive_result_service.**_normalize_status**(*status: str*) → str

archivebox.services.archive_result_service.**_normalize_snapshot_title**(*candidate: str*, ***, *snapshot_url: str*) → str

archivebox.services.archive_result_service.**_extract_snapshot_title**(*snapshot_output_dir: str*, *plugin: str*, *output_str: str*, ***, *snapshot_url: str*) → str

archivebox.services.archive_result_service.**_should_update_snapshot_title**(*current_title: str*, *next_title: str*, ***, *snapshot_url: str*) → bool

archivebox.services.archive_result_service.**_save_archiveresult_event_to_db**(*event: abx_dl.events.ArchiveResultEvent*, *process_started: abx_dl.events.ProcessStartedEvent | None*) → None

Project one ArchiveResultEvent with a single thread-sensitive ORM hop.

Django's async ORM still delegates each query to sync Django work. The hot search/index maintenance path was paying that handoff separately for Snapshot lookup, Process lookup, ArchiveResult lookup, update, and title checks. Keep the public ArchiveResultEvent path intact, but run the DB projection as one short synchronous block so SQLite sees the same indexed reads/writes without per-query asyncio/threadpool churn.

archivebox.services.archive_result_service.**mark_archiveresult_started**(*event: abx_dl.events.ProcessStartedEvent*, **, snapshot_id: str*, *process_id: str*) → None

Project a running abx-dl hook after its OS process is persisted.

This DB state records process execution; the scheduler separately waits for stdout readiness before advancing past a background hook. Neither boundary proves that an archive was produced. Final status comes from the owning hook's ArchiveResultEvent, including corrections after an abnormal exit. Rows are per snapshot/plugin/hook: a recorder's start hook cannot declare that its separate export hook succeeded.

class archivebox.services.archive_result_service.**ArchiveResultService**(*bus*)

Bases: abx_dl.services.base.BaseService

Project abx-dl ArchiveResult facts into Django models.

Initialization

LISTENS_TO

None

EMITS

[]

async on_ArchiveResultEvent__save_to_db(*event: abx_dl.events.ArchiveResultEvent*) → None

archivebox.services.resource_admission

Admission signals, not estimates of how much RAM an extractor will need.

Module Contents

Classes

ResourceAdmission	

Functions

zram_swap_identifiers	Return zram block names and major:minor IDs known to sysfs.

continues on next page

Table 265 – continued from previous page

swap_source_identifiers	Identify a swap source by its visible name and, for block devices, ID.
disk_backed_swap_free_bytes	Sum free active swap except zram, whose capacity is already backed by RAM.
host_swap_free_bytes	Use only independently backed swap as extra capacity on Linux.

Data

RESOURCE_RECHECK_SECONDS
MEMORY_FULL_STALL_PERCENT
ZRAM_DEVICE_NAME
resource_admission

API

archivebox.services.resource_admission.**RESOURCE_RECHECK_SECONDS**
2.0

archivebox.services.resource_admission.**MEMORY_FULL_STALL_PERCENT**
10.0

archivebox.services.resource_admission.**ZRAM_DEVICE_NAME**
'compile(...)'

archivebox.services.resource_admission.**zram_swap_identifiers**(*sys_block: pathlib.Path = Path('/sys/block')*) → set[str]
Return zram block names and major:minor IDs known to sysfs.

archivebox.services.resource_admission.**swap_source_identifiers**(*filename: str*) → set[str]
Identify a swap source by its visible name and, for block devices, ID.

archivebox.services.resource_admission.**disk_backed_swap_free_bytes**(*swaps: str*, *zram_identifiers: set[str]*) → int
Sum free active swap except zram, whose capacity is already backed by RAM.

archivebox.services.resource_admission.**host_swap_free_bytes**(*meminfo_swap_free: int*) → int
Use only independently backed swap as extra capacity on Linux.

class archivebox.services.resource_admission.**ResourceAdmission**

Initialization

static cgroup_paths() → list[pathlib.Path]

snapshot_limit(*configured: int*) → int

additional_snapshot_slots(*configured: int*, *active: int*, *observed_cost: int | None*) → int

static slots_for_headroom(*configured: int*, *active: int*, *observed_cost: int*, *available: int*) → int

memory_headroom() → tuple[int, int] | None
Return workload RAM+swap usage and effective available headroom.

observe_memory_stalls(*source: str*, *total: int*, *avg10: float*, *sampled_at: float*) → bool

memory_pressure() → str | None

archivebox.services.resource_admission.**resource_admission**

'ResourceAdmission(...)'

archivebox.services.runner

Module Contents

Classes

CrawlRunner

Functions

_bus_name	
_runner_short_id	
_runner_label	
_runner_console_line	
_count_selected_hooks	
_enable_requested_plugins	
_is_nonfatal_setup_hook	
_runner_task_context	
_is_external_task_cancelled	
_run_event_now	
ensure_background_runner	
run_crawl	
_run_crawl_locked	
_run_binary	
run_binary	
run_snapshot_maintenance	
defer_crawl_for_resources	Leave extraction durable; never delay cancellation or final cleanup.
run_due_crawl	
_run_due_crawl_locked	
run_due_snapshot	
_run_due_snapshot_locked	
run_due_binary	
_run_install	
run_install	
_first_due_id	
_run_due_crawl_status	
_run_due_snapshot_query	
_run_due_snapshot_id	
_run_due_binary	
run_pending_crawls	

API

archivebox.services.runner.**_bus_name**(*prefix: str*, *identifier: str*) → str

archivebox.services.runner.**_runner_short_id**(*identifier*) → str

archivebox.services.runner.**_runner_label**(*value: str*, *, *reserve: int*) → str

archivebox.services.runner.**_runner_console_line**(*, *crawl=None*, *crawl_id=None*, *snapshot=None*, *status: str = 'STARTED'*) → None

archivebox.services.runner.**_count_selected_hooks**(*catalog: abx_dl.catalog.PluginCatalog*, *selected_plugins: list[str] | None*) → int

archivebox.services.runner.**_enable_requested_plugins**(*config: dict[str, Any]*, *plugins: abx_dl.catalog.PluginCatalog*) → None

archivebox.services.runner.**_is_nonfatal_setup_hook**(*plugin_name: str*, *hook_name: str*) → bool

archivebox.services.runner.**_runner_task_context**() → contextvars.Context

archivebox.services.runner.**_is_external_task_cancelled**(*error: asyncio.CancelledError*) → bool

async archivebox.services.runner.**_run_event_now**(*event*, *timeout: float | None = None*)

archivebox.services.runner.**ensure_background_runner**() → bool

class archivebox.services.runner.**CrawlRunner**(*crawl*, *, *snapshot_ids: list[str] | None = None*, *selected_plugins: list[str] | None = None*, *process_discovered_snapshots_inline: bool = True*, *show_progress: bool = True*, *interactive_interrupts: bool = False*, *config_overrides: dict[str, Any] | None = None*)

Initialization

execution_interrupted() → bool

Owner loss preserves work; explicit user abort keeps its sealing policy.

async on_CrawlAbortEvent(*event: abx_dl.events.CrawlAbortEvent*) → None

_interrupt_hook() → None

async _prompt_in_foreground(*hook_name: str*) → Literal[abort, retry, skip]

_choose_interrupt_action(*choice: Literal[abort, retry, skip]*) → None

_request_abort_from_signal(*sig: signal.Signals*) → None

async crawl_is_cancelled() → bool

async crawl_is_paused() → bool

async watch_for_cancelled_crawl(*parent_event: abxbus.BaseEvent*, *, *poll_interval: float = 1.0*) → None

property allow_maintenance_on_inactive_crawl: bool

Run explicitly targeted plugin work on already-sealed snapshots.

async run() → None

async enqueue_snapshot(*snapshot_id: str*, *crawl_start_event: abx_dl.events.CrawlStartEvent | None = None*) → None

async stop_snapshot_tasks() → None

async wait_for_snapshot_tasks(*, *enqueue_projected: bool = True*) → None

_observe_snapshot_memory() → None

_finish_warmup_snapshot(*snapshot_id: str*) → None

async heartbeat_active_leases() → None

async drain_snapshot_tasks() → None

async enqueue_pending_snapshots_from_projection() → None

load_run_state() → list[str]

create_initial_snapshots() → list

finalize_run_state() → None

_create_live_ui() → abx_dl.cli.LiveBusUI | None

load_snapshot_payload(*snapshot_id: str*) → dict[str, Any]

async enqueue_discovered_snapshots_from_outputs(*snapshot_payload: dict[str, Any]*) → None

async run_crawl(*root_snapshot_id: str*, *snapshot_ids: list[str]*) → None

async run_snapshot(*snapshot_id: str*, *crawl_start_event: abx_dl.events.CrawlStartEvent | None = None*) → None

seal_snapshot_due_to_limit(*snapshot_id: str*) → None

archivebox.services.runner.**run_crawl**(*crawl_id: str*, *, *snapshot_ids: list[str] | None = None*, *selected_plugins: list[str] | None = None*, *process_discovered_snapshots_inline: bool = True*, *show_progress: bool = True*, *interactive_interrupts: bool = False*, *config_overrides: dict[str, Any] | None = None*) → None

archivebox.services.runner.**_run_crawl_locked**(*crawl_id: str*, *, *snapshot_ids: list[str] | None = None*, *selected_plugins: list[str] | None = None*, *process_discovered_snapshots_inline: bool = True*, *show_progress: bool = True*, *interactive_interrupts: bool = False*, *config_overrides: dict[str, Any] | None = None*) → None

async archivebox.services.runner.**_run_binary**(*binary_id: str*) → None

archivebox.services.runner.**run_binary**(*binary_id: str*) → None

archivebox.services.runner.**run_snapshot_maintenance**(*snapshot_id: str*, *, *output_dir: pathlib.Path | None = None*) → bool

archivebox.services.runner.**defer_crawl_for_resources**(*crawl*, *, *excluded_snapshot_ids=()*) → bool

Leave extraction durable; never delay cancellation or final cleanup.

archivebox.services.runner.**run_due_crawl**(*crawl*, *, *lock_seconds: int*, *interactive_interrupts: bool = False*) → bool

archivebox.services.runner.**_run_due_crawl_locked**(*crawl*, *, *lock_seconds: int*, *interactive_interrupts: bool = False*) → bool

archivebox.services.runner.**run_due_snapshot**(*snapshot*, *, *lock_seconds: int*, *interactive_interrupts: bool = False*, *runtime_config=None*) → bool

archivebox.services.runner.**_run_due_snapshot_locked**(*snapshot*, *, *lock_seconds: int*, *interactive_interrupts: bool = False*, *runtime_config=None*) → bool

archivebox.services.runner.**run_due_binary**(*binary*, *, *lock_seconds: int*) → bool

async archivebox.services.runner.**_run_install**(*plugin_names: list[str] | None = None*) → None

archivebox.services.runner.**run_install**(*, *plugin_names: list[str] | None = None*) → None

archivebox.services.runner.**_first_due_id**(*queryset*, *, *newest_first: bool = False*)

archivebox.services.runner.**_run_due_crawl_status**(*status: str*, *, *crawl_id: str | None*, *lock_seconds: int*, *interactive_interrupts: bool*, *newest_first: bool = False*) → bool

archivebox.services.runner.**_run_due_snapshot_query**(*queryset*, *, *lock_seconds: int*, *interactive_interrupts: bool*, *runtime_config*, *newest_first: bool = False*) → bool

archivebox.services.runner.**_run_due_snapshot_id**(*snapshot_id*, *, *lock_seconds: int*, *interactive_interrupts: bool*, *runtime_config*) → bool

archivebox.services.runner.**_run_due_binary**() → bool

archivebox.services.runner.**run_pending_crawls**(*, *daemon: bool = False*, *crawl_id: str | None = None*, *maintenance_only: bool = False*, *interactive_interrupts: bool = False*) → int

archivebox.services.snapshot_service

Module Contents

Classes

SnapshotService

Functions

project_discovered_snapshots	Persist parser output before the parent Snapshot can enter a final state.
finalize_completed_snapshot	
_crawl_limit_stop_reason	

API

archivebox.services.snapshot_service.**project_discovered_snapshots**(*snapshot_id: str*) → list

Persist parser output before the parent Snapshot can enter a final state.

archivebox.services.snapshot_service.**finalize_completed_snapshot**(*snapshot_id: str*, *, *owned_retry_at*, *was_sealed: bool*, *consumed_retry_plugins: list[str] | None = None*, *output_dir=None*, *crawl_limit_stop_reason: str | None = None*) → None

archivebox.services.snapshot_service.**_crawl_limit_stop_reason**(*crawl*) → str

class archivebox.services.snapshot_service.**SnapshotService**(*bus*, *, *crawl_id: str*, *interrupted: collections.abc.Callable[[], bool] | None = None*)

Bases: abx_dl.services.base.BaseService

Initialization

LISTENS_TO

None

EMITS

[]

async on_SnapshotEvent(*event: abx_dl.events.SnapshotEvent*) → None

async renew_lease(*snapshot_id: str*, *lease_until*) → bool | None

async on_SnapshotCompletedEvent(*event: abx_dl.events.SnapshotCompletedEvent*) → None

archivebox.services.tag_service

Module Contents

Classes

TagService

API

class archivebox.services.tag_service.**TagService**(*bus*)

Bases: abx_dl.services.base.BaseService

Initialization

LISTENS_TO

None

EMITS

[]

async on_TagEvent__save_to_db(*event: abx_dl.events.TagEvent*) → None

archivebox.services.machine_service

Module Contents

Classes

MachineService

Functions

_is_binary_event_key	MachineEvent projector only ever writes binary-related state.
_strip_to_binary_keys	

API

archivebox.services.machine_service.**_is_binary_event_key**(*key: str*) → bool

MachineEvent projector only ever writes binary-related state.

Machine.config mirrors ArchiveBox.conf so arbitrary user keys can legitimately live there — but they get there through the file ↔ DB sync, not through events. Letting events write arbitrary keys would let an untrusted plugin overwrite security-sensitive user config (the file ↔ DB mirror is a security boundary), so the projector strips anything that isn't a binary path or the binary install cache.

archivebox.services.machine_service.**_strip_to_binary_keys**(*config: dict[str, Any] | None*) → dict[str, Any]

class archivebox.services.machine_service.**MachineService**(*bus*)

Bases: abx_dl.services.base.BaseService

Initialization

LISTENS_TO

None

EMITS

[]

async on_MachineEvent__save_to_db(*event: abx_dl.events.MachineEvent*) → None

archivebox.services.crawl_service

Module Contents

Classes

CrawlService

API

class archivebox.services.crawl_service.**CrawlService**(*bus*, *, *crawl_id: str*)

Bases: abx_dl.services.base.BaseService

Initialization

LISTENS_TO

None

EMITS

[]

async on_CrawlSetupEvent__save_to_db(*event: abx_dl.events.CrawlSetupEvent*) → None

async on_CrawlStartEvent__save_to_db(*event: abx_dl.events.CrawlStartEvent*) → None

async on_CrawlCleanupEvent__save_to_db(*event: abx_dl.events.CrawlCleanupEvent*) → None

async on_CrawlCompletedEvent__save_to_db(*event: abx_dl.events.CrawlCompletedEvent*) → None

Package Contents

Data

__all__

API

archivebox.services.**__all__**

['ArchiveResultService', 'ArchiveBoxBinaryService', 'CrawlService', 'MachineService', 'ProcessServic…

archivebox.plugins

Submodules

archivebox.plugins.forms

Module Contents

Classes

PluginConfigFormMixin

Functions

get_plugin_choices	Get available extractor plugins from discovered hooks.
get_plugin_choice_label	
get_choice_field	
_plugin_config_input_name	
_schema_types	
_jsonish	
_same_config_value	
_coerce_plugin_config_value	
_resolve_required_binary_name	
_iter_required_binary_names	
_build_required_binary_url_lookup	Resolve admin URLs for every required binary across all plugin schemas in a single DB query.
_build_required_binary_links	
get_plugin_config_binary_urls	

Data

PLUGIN_CONFIG_FIELD_PREFIX
PLUGIN_GROUPS
TIMEOUT_INPUT_PATTERN
_BINARY_TEMPLATE_PATTERN

API

archivebox.plugins.forms.**PLUGIN_CONFIG_FIELD_PREFIX**

'plugin_config__'

archivebox.plugins.forms.**PLUGIN_GROUPS**

((‘main’, ‘main_plugins’, ‘Main’), (‘page_setup’, ‘page_setup_plugins’, ‘Page Setup’), (‘media’, ‘me…

archivebox.plugins.forms.**TIMEOUT_INPUT_PATTERN**

‘(0|[1-9][0-9]|*[0-9]+(?:.[0-9]+)?*\s(?:s|sec|secs|second|seconds|m|min|mins|minute|minutes|h|hr|hrs…’

archivebox.plugins.forms.**get_plugin_choices**()

Get available extractor plugins from discovered hooks.

archivebox.plugins.forms.**get_plugin_choice_label**(*plugin_name: str*, *plugin_configs: dict[str, dict]*) → str

archivebox.plugins.forms.**get_choice_field**(*form: django.forms.Form*, *name: str*) → django.forms.ChoiceField

archivebox.plugins.forms.**_plugin_config_input_name**(*plugin_name: str*, *config_key: str*) → str

archivebox.plugins.forms.**_schema_types**(*schema: collections.abc.Mapping[str, Any]*) → list[str]

archivebox.plugins.forms.**_jsonish**(*value: Any*) → str

archivebox.plugins.forms.**_same_config_value**(*left: Any*, *right: Any*) → bool

archivebox.plugins.forms.**_coerce_plugin_config_value**(*raw_value: Any*, *schema: collections.abc.Mapping[str, Any]*) → Any

class archivebox.plugins.forms.**PluginConfigFormMixin**

plugin_groups: list[dict[str, Any]]

None

allow_crawl_execution_config_fields

True

build_plugin_groups(*runtime_config: collections.abc.Mapping[str, Any] | None = None*) → None

_build_plugin_cards(*field_name: str*, *plugin_names: collections.abc.Iterable[str]*, *plugin_configs: dict[str, dict[str, Any]]*, *runtime_config: collections.abc.Mapping[str, Any]*, *binary_url_lookup: collections.abc.Mapping[str, str] | None = None*) → list[dict[str, Any]]

_build_plugin_config_field(*plugin_name: str*, *config_key: str*, *prop_schema: collections.abc.Mapping[str, Any]*, *runtime_config: collections.abc.Mapping[str, Any]*) → dict[str, Any]

clean_plugin_config_overrides(*effective_config: collections.abc.Mapping[str, Any] | None = None*) → dict[str, Any]

plugin_config_keys() → set[str]

archivebox.plugins.forms.**_BINARY_TEMPLATE_PATTERN**

'compile(...)'

archivebox.plugins.forms.**_resolve_required_binary_name**(*template_name: str*, *runtime_config: collections.abc.Mapping[str, Any]*) → str

archivebox.plugins.forms.**_iter_required_binary_names**(*required_binaries: collections.abc.Iterable[Any]*, *runtime_config: collections.abc.Mapping[str, Any]*) → collections.abc.Iterable[str]

archivebox.plugins.forms.**_build_required_binary_url_lookup**(*plugin_configs: collections.abc.Mapping[str, dict[str, Any]]*, *runtime_config: collections.abc.Mapping[str, Any]*) → dict[str, str]

Resolve admin URLs for every required binary across all plugin schemas in a single DB query.

archivebox.plugins.forms.**_build_required_binary_links**(*required_binaries: list[dict[str, Any]]*, *runtime_config: collections.abc.Mapping[str, Any]*, *binary_url_lookup: collections.abc.Mapping[str, str] | None = None*) → list[dict[str, str]]

archivebox.plugins.forms.**get_plugin_config_binary_urls**(*runtime_config: collections.abc.Mapping[str, Any]*) → dict[str, str]

archivebox.plugins.apps

Module Contents

Classes

PluginsConfig

API

class archivebox.plugins.apps.**PluginsConfig**

Bases: `django.apps.AppConfig`

default_auto_field

'django.db.models.BigAutoField'

name

'archivebox.plugins'

verbose_name

'Plugins'

archivebox.plugins.output_groups

Snapshot output taxonomy. Display preference is independent of hook order.

Module Contents

Functions

_plugin_presentation	
output_group_for_plugin	Return the snapshot-detail stack that contains a plugin output.
plugin_icon_is_hidden	
plugin_card_is_hidden	
display_plugin_name	
plugin_output_sizes	Count each plugin's files once, even when several hooks report them.
order_output_plugins	Use the detail-card ordering for icon piles, without synthetic cards.
snapshot_thumbnail_result	Select the first available card using plugin-owned thumbnail priority.
order_snapshot_outputs	Annotate and order available cards; unknown plugins stay visible in Other files.

Data

_GROUP_LABELS
PLUGIN_PRESENTATION
OUTPUT_GROUPS

continues on next page

Table 281 – continued from previous page

UNORDERED_OUTPUTS
HIDDEN_ICON_PLUGINS
HIDDEN_CARD_PLUGINS

API

archivebox.plugins.output_groups.**_GROUP_LABELS**

((‘html’, ‘HTML’), (‘raster’, ‘Raster’), (‘article_text’, ‘Article text’), (‘embedded_media’, ‘Embed…

archivebox.plugins.output_groups.**_plugin_presentation**() → dict[str, dict[str, Any]]

archivebox.plugins.output_groups.**PLUGIN_PRESENTATION**

‘_plugin_presentation(…)’

archivebox.plugins.output_groups.**OUTPUT_GROUPS**

‘tuple(…)’

archivebox.plugins.output_groups.**UNORDERED_OUTPUTS**

None

archivebox.plugins.output_groups.**HIDDEN_ICON_PLUGINS**

‘frozenset(…)’

archivebox.plugins.output_groups.**HIDDEN_CARD_PLUGINS**

‘frozenset(…)’

archivebox.plugins.output_groups.**output_group_for_plugin**(*plugin: str*) → str

Return the snapshot-detail stack that contains a plugin output.

archivebox.plugins.output_groups.**plugin_icon_is_hidden**(*plugin: str*) → bool

archivebox.plugins.output_groups.**plugin_card_is_hidden**(*plugin: str*) → bool

archivebox.plugins.output_groups.**display_plugin_name**(*plugin: str*) → str

archivebox.plugins.output_groups.**plugin_output_sizes**(*results*) → dict[str, int]

Count each plugin’s files once, even when several hooks report them.

archivebox.plugins.output_groups.**order_output_plugins**(*plugins*, *sizes: dict[str, int]*) → list[str]

Use the detail-card ordering for icon piles, without synthetic cards.

archivebox.plugins.output_groups.**snapshot_thumbnail_result**(*results*)

Select the first available card using plugin-owned thumbnail priority.

archivebox.plugins.output_groups.**order_snapshot_outputs**(*outputs: list[dict[str, Any]]*) → list[dict[str, Any]]

Annotate and order available cards; unknown plugins stay visible in Other files.

archivebox.plugins.discovery

Module Contents

Classes

ConfigLookup	
PluginSpecialConfig	

Functions

iter_plugin_dirs	Return the exact plugin directories exposed by the shared catalog.
get_plugin_catalog	
get_plugin_config_resolver	
get_plugins	Get list of available plugins by discovering plugin directories.
get_plugin_name	Get the base plugin name without numeric prefix.
get_archive_result_aliases	
get_archive_result_names	Return the canonical ArchiveResult name and its declared historic aliases.
plugin_card_is_interactive	
get_plugin_output_extension_preference	
get_plugin_default_output_path	
get_snapshot_thumbnail_card_order	Return plugin-owned ArchiveResult names eligible for snapshot thumbnails.
get_snapshot_role_names	Return ArchiveResult names for a plugin-owned snapshot presentation role.
get_snapshot_role_result	Select the first successful result declared for a presentation role.
get_plugin_presentation_modules	Load optional plugin-owned presentation hooks declared in manifests.
get_plugin_presentation_module	Load one plugin's optional presentation hooks module.
plugin_has_custom_full_response	Return whether a plugin renders its full template through a response hook.
get_extra_snapshot_output_cards	Collect additional cards from plugin-owned presentation hooks.
serve_plugin_replay_asset	Give plugin presentation hooks the first chance to serve replay assets.
render_plugin_full_response	Let plugin presentation hooks render a saved output's full template.
get_enabled_plugins	Get the list of enabled plugins based on config and available hooks.
get_search_backends	Return plugins that declare both standalone search commands.
discover_plugin_configs	Discover all plugin config.json schemas.
get_plugin_special_config	Extract special config keys for a plugin following naming conventions.
get_plugin_template	Get a plugin template by plugin name and template type.
get_plugin_icon	Get the icon for a plugin from its icon.html template.

Data

BUILTIN_PLUGINS_DIR	
USER_PLUGINS_DIR	

continues on next page

Table 284 – continued from previous page

DEFAULT_TEMPLATES

API

class archivebox.plugins.discovery.**ConfigLookup**

Bases: typing.Protocol

get(*key: str*, *default: Any = None*) → Any

items() → collections.abc.Iterable[tuple[str, Any]]

class archivebox.plugins.discovery.**PluginSpecialConfig**

Bases: typing.TypedDict

enabled: bool

None

timeout: int

None

binary: str

None

archivebox.plugins.discovery.**BUILTIN_PLUGINS_DIR**

'resolve(...)'

archivebox.plugins.discovery.**USER_PLUGINS_DIR**

None

archivebox.plugins.discovery.**iter_plugin_dirs**() → list[pathlib.Path]

Return the exact plugin directories exposed by the shared catalog.

archivebox.plugins.discovery.**get_plugin_catalog**() → abx_dl.catalog.PluginCatalog

archivebox.plugins.discovery.**get_plugin_config_resolver**() → abx_dl.catalog.PluginConfigResolver

archivebox.plugins.discovery.**get_plugins**() → list[str]

Get list of available plugins by discovering plugin directories.

Returns plugin directory names for any plugin that exposes hooks, config.json, or a standardized templates/icon.html asset. This includes non-extractor plugins such as binary providers and shared base plugins.

archivebox.plugins.discovery.**get_plugin_name**(*plugin: str*) → str

Get the base plugin name without numeric prefix.

Examples: '10_title' -> 'title' '26_readability' -> 'readability' '50_parse_html_urls' -> 'parse_html_urls'

archivebox.plugins.discovery.**get_archive_result_aliases**() → dict[str, str]

archivebox.plugins.discovery.**get_archive_result_names**(*plugin_name: str*) → tuple[str, ...]

Return the canonical ArchiveResult name and its declared historic aliases.

archivebox.plugins.discovery.**plugin_card_is_interactive**(*plugin_name: str*) → bool

archivebox.plugins.discovery.**get_plugin_output_extension_preference**(*plugin_name: str*) → tuple[tuple[str, ...], ...] | None

archivebox.plugins.discovery.**get_plugin_default_output_path**(*plugin_name: str*) → str | None

archivebox.plugins.discovery.**get_snapshot_thumbnail_card_order**() → dict[str, int]

Return plugin-owned ArchiveResult names eligible for snapshot thumbnails.

archivebox.plugins.discovery.**get_snapshot_role_names**(*role: str*) → tuple[str, ...]

Return ArchiveResult names for a plugin-owned snapshot presentation role.

archivebox.plugins.discovery.**get_snapshot_role_result**(*results*, *role: str*)

Select the first successful result declared for a presentation role.

archivebox.plugins.discovery.**get_plugin_presentation_modules**() → tuple[types.ModuleType, ...]

Load optional plugin-owned presentation hooks declared in manifests.

archivebox.plugins.discovery.**get_plugin_presentation_module**(*plugin_name: str*) → types.ModuleType | None

Load one plugin's optional presentation hooks module.

archivebox.plugins.discovery.**plugin_has_custom_full_response**(*plugin_name: str*) → bool

Return whether a plugin renders its full template through a response hook.

archivebox.plugins.discovery.**get_extra_snapshot_output_cards**(*outputs: list[dict[str, Any]]*) → list[dict[str, Any]]

Collect additional cards from plugin-owned presentation hooks.

archivebox.plugins.discovery.**serve_plugin_replay_asset**(*path: str*, *config*, *response_class*)

Give plugin presentation hooks the first chance to serve replay assets.

archivebox.plugins.discovery.**render_plugin_full_response**(*path: str*, *output_url: str*, ***kwargs*)

Let plugin presentation hooks render a saved output's full template.

archivebox.plugins.discovery.**get_enabled_plugins**(*config:* archivebox.plugins.discovery.ConfigLookup | *None = None*, ***config_kwargs: Any*) → list[str]

Get the list of enabled plugins based on config and available hooks.

Filters plugins by USE_/SAVE_ flags. Only returns plugins that are enabled.

archivebox.plugins.discovery.**get_search_backends**()

Return plugins that declare both standalone search commands.

archivebox.plugins.discovery.**discover_plugin_configs**() → dict[str, dict[str, Any]]

Discover all plugin config.json schemas.

Each plugin can define a config.json file with JSONSchema defining its configuration options. This is intentionally cached because these schemas are plugin package metadata, not live user config; runtime values still come from env/db config at each callsite.

archivebox.plugins.discovery.**get_plugin_special_config**(*plugin_name: str*, *config:* archivebox.plugins.discovery.ConfigLookup, *_visited: set[str] | None = None*) → *archivebox.plugins.discovery.PluginSpecialConfig*

Extract special config keys for a plugin following naming conventions.

ArchiveBox recognizes 3 special config key patterns per plugin: - {PLUGIN}_ENABLED: Enable/disable toggle (default True) - {PLUGIN}_TIMEOUT: Plugin-specific timeout (fallback to TIMEOUT, default 300) - {PLUGIN}_BINARY: Primary binary path (default to plugin_name)

archivebox.plugins.discovery.**DEFAULT_TEMPLATES**

None

archivebox.plugins.discovery.**get_plugin_template**(*plugin: str*, *template_name: str*, *fallback: bool = True*) → str | None

Get a plugin template by plugin name and template type.

Args: plugin: Plugin name (e.g., 'screenshot', '15_singlefile') template_name: A template slot such as 'icon', 'card', or 'full'. fallback: If True, return default template if plugin template not found

archivebox.plugins.discovery.**get_plugin_icon**(*plugin: str*) → str

Get the icon for a plugin from its icon.html template.

archivebox.plugins.views

Module Contents

Functions

render_code_block	
render_highlighted_json_block	
get_plugin_docs_url	
get_plugin_hook_source_url	
get_machine_admin_url	
render_code_tag_list	
render_link_tag_list	
render_plugin_metadata_html	
render_property_links	
render_config_properties_html	
render_hook_links_html	
get_filesystem_plugins	Discover plugins from filesystem directories.
find_plugin_for_config_key	
get_config_definition_link	
plugins_list_view	
plugin_detail_view	

Data

ABX_PLUGINS_DOCS_BASE_URL	
ABX_PLUGINS_GITHUB_BASE_URL	
LIVE_PLUGIN_BASE_URL	
JSON_TOKEN_RE	

API

archivebox.plugins.views.**ABX_PLUGINS_DOCS_BASE_URL**

'https://plugins.archivebox.io/'

archivebox.plugins.views.**ABX_PLUGINS_GITHUB_BASE_URL**

'https://github.com/ArchiveBox/abx-plugins/tree/main/abx_plugins/plugins/'

archivebox.plugins.views.**LIVE_PLUGIN_BASE_URL**

'/admin/environment/plugins/'

archivebox.plugins.views.**JSON_TOKEN_RE**

'compile(...)'

archivebox.plugins.views.**render_code_block**(*text: str*, *, *highlighted: bool = False*) → str

archivebox.plugins.views.**render_highlighted_json_block**(*value: Any*) → str

archivebox.plugins.views.**get_plugin_docs_url**(*plugin_name: str*) → str

archivebox.plugins.views.**get_plugin_hook_source_url**(*plugin_name: str*, *hook_name: str*) → str

archivebox.plugins.views.**get_machine_admin_url**() → str | None

archivebox.plugins.views.**render_code_tag_list**(*values: list[str]*) → str

archivebox.plugins.views.**render_link_tag_list**(*values: list[str]*, *url_resolver: collections.abc.Callable[[str], str] | None = None*) → str

archivebox.plugins.views.**render_plugin_metadata_html**(*config: dict[str, Any]*) → str

archivebox.plugins.views.**render_property_links**(*prop_name: str*, *prop_info: dict[str, Any]*, *machine_admin_url: str | None*) → str

archivebox.plugins.views.**render_config_properties_html**(*properties: dict[str, Any]*, *machine_admin_url: str | None*) → str

archivebox.plugins.views.**render_hook_links_html**(*plugin_name: str*, *hooks: list[str]*, *source: str*) → str

archivebox.plugins.views.**get_filesystem_plugins**() → dict[str, dict[str, Any]]

Discover plugins from filesystem directories.

archivebox.plugins.views.**find_plugin_for_config_key**(*key: str*) → str | None

archivebox.plugins.views.**get_config_definition_link**(*key: str*) → tuple[str, str]

archivebox.plugins.views.**plugins_list_view**(*request: django.http.HttpRequest*, ***kwargs*) → admin_data_views.typing.TableContext

archivebox.plugins.views.**plugin_detail_view**(*request: django.http.HttpRequest*, *key: str*, ***kwargs*) → admin_data_views.typing.ItemContext

archivebox.plugins.hooks

ArchiveBox adapters around the framework-free abx-dl plugin runtime.

Discovery and execution are owned by abx-dl. ArchiveBox keeps only the small Django projection adapter and its application-specific URL-output reader.

Module Contents

Functions

is_background_hook

continues on next page

Table 287 – continued from previous page

normalize_hook_event_name	
discover_hooks	Return the exact catalog hooks used by abx-dl, in execution order.
collect_urls_from_plugins	Read the durable urls.jsonl interface emitted by parser plugins.

API

archivebox.plugins.hooks.**is_background_hook**(*hook_name: str*) → bool

archivebox.plugins.hooks.**normalize_hook_event_name**(*event_name: str*) → str | None

archivebox.plugins.hooks.**discover_hooks**(*event_name: str*, *filter_disabled: bool = True*, *config:* archivebox.plugins.discovery.ConfigLookup | *None = None*, ***config_kwargs: Any*) → list[pathlib.Path]

Return the exact catalog hooks used by abx-dl, in execution order.

archivebox.plugins.hooks.**collect_urls_from_plugins**(*snapshot_dir: pathlib.Path*) → list[dict[str, Any]]

Read the durable urls.jsonl interface emitted by parser plugins.

Submodules

archivebox.uuid_compat

UUID7 compatibility layer.

Module Contents

Classes

CompactUUID	
CompactUUIDField	

Functions

compact_uuid	
uuid7	

Data

__all__	

API

class archivebox.uuid_compat.**CompactUUID**(*hex=None*, *bytes=None*, *bytes_le=None*, *fields=None*, *int=None*, *version=None*, ***, *is_safe=SafeUUID.unknown*)

Bases: uuid.UUID

__str__() → str

archivebox.uuid_compat.**compact_uuid**(*value: uuid.UUID | str | None*) → *archivebox.uuid_compat.CompactUUID* | None

class archivebox.uuid_compat.**CompactUUIDField**

Bases: django.db.models.UUIDField

to_python(*value*)

from_db_value(*value*, *expression*, *connection*)

deconstruct()

archivebox.uuid_compat.**uuid7**() → *archivebox.uuid_compat.CompactUUID*

archivebox.uuid_compat.**__all__**

['CompactUUID', 'CompactUUIDField', 'compact_uuid', 'uuid7']

archivebox.__main__

This is the entrypoint for python -m archivebox ...

Module Contents

Data

ASCII_LOGO_MINI

API

archivebox.__main__.**ASCII_LOGO_MINI** = <Multiline-String>

archivebox.manage

Package Contents

Classes

_ReconfigurableStream

Functions

__getattr__

Data

ASCII_LOGO
PACKAGE_DIR

continues on next page

Table 294 – continued from previous page

__version__
__author__
__license__
__all__
ASCII_ICON

API

class archivebox.**_ReconfigurableStream**

Bases: typing.Protocol

reconfigure(*, *line_buffering: bool*) → object

archivebox.**ASCII_LOGO = <Multiline-String>**

archivebox.**PACKAGE_DIR**

None

archivebox.**__version__**

None

archivebox.**__author__**

'ArchiveBox'

archivebox.**__license__**

'MIT'

archivebox.**__getattr__**(*name: str*)

archivebox.**__all__**

('ASCII_LOGO', 'ASCII_ICON', 'PACKAGE_DIR', 'DATA_DIR', 'CONSTANTS', 'VERSION', 'BUILTIN_PLUGINS_DIR…

archivebox.**ASCII_ICON = <Multiline-String>**

1.7 Meta

1.7.1 Roadmap

Comment here to discuss the contribution roadmap:
Official Roadmap Discussion.

Planned Specification

(this is not set in stone, just a rough estimate)

v0.7: Schema improvements

- move config loading logic into settings.py
- move all the extractors into "plugin" style folders that register their own config
- right now, the paths of the extractor output are scattered all over the codebase, e.g. `output.pdf` (should be moved to constants at the top of the plugin config file)

- make out_dir, link_dir, extractor_dir, naming consistent across codebase
- remove `timestamps` as primary keys in favor of hashes, UUIDs, or some other slug https://github.com/ArchiveBox/ArchiveBox/issues/74
- create a migration system for folder layout independent of the index
- make `Tag` a real model `ManyToMany` with Snapshots
- allow multiple Snapshots of the same site over time + CLI / UI to manage those, + migration from old style `#2020-01-01` hack to proper versioned snapshots
- upgrade from Django 3 to Django 5 https://github.com/ArchiveBox/ArchiveBox/issues/988

v0.8: Security

- Add CSRF/CSP/XSS protection to rendered archive pages
- Provide secure reverse proxy in front of archivebox server in docker-compose.yml
- Create UX flow for users to setup session cookies / auth for archiving private sites
  - cookies for wget, curl, etc low-level commands
  - localstorage, cookies, indexedb setup for chrome archiving methods

v0.9: Performance

- setup huey, break up archiving process into tasks on a queue that a worker pool executes
- setup pyppeteer2 to wrap chrome so that it's not open/closed during each extractor

v1.0: Full headless browser control

- run user-scripts / extensions in the context of the page during archiving
- community userscripts for unrolling twitter threads, reddit threads, youtube comment sections, etc.
- pywb-based headless browser session recording and warc replay
- archive proxy support
  - support sending upstream requests through an external proxy
  - support for exposing a proxy that archives all downstream traffic

...

v2.0 Federated or distributed archiving

- ZFS / merkel tree for storing archive output subresource hashes
- DHT for assigning merkel tree hash:file shards to nodes
- tag system for tagging certain hashes with human-readable names, e.g. title, url, tags, filetype etc.
- distributed tag lookup system

Major long-term changes

- release **`pip`, `apt`, `pkg`, and `brew` packaged distributions** for installing ArchiveBox
- add an **optional web GUI** for managing sources, adding new links, and viewing the archive
- switch to django + **sqlite db with migrations system** & json/html export for managing archive schema changes and persistence
- modularize internals to allow importing individual components
- switch to sha256 of URL as unique link ID
- support **storing multiple snapshots** of pages over time
- support **custom user puppeteer scripts to run while archiving** (e.g. for expanding reddit threads, scrolling thread on twitter, etc)
- support named collections of archived content with different user access permissions
- support sharing archived assets via DHT + torrent / ipfs / ZeroNet / other sharing system

Smaller planned features

- support pushing pages to multiple 3rd party services using ArchiveNow instead of just archive.org
- body text extraction to markdown (using ~~fathom~~ readability and mercury)
- featured image / thumbnail extraction
- auto-tagging links based on important/frequent keywords in extracted text (like pocket)
- automatic article summary paragraphs from extracted text with nlp summarization library
- full-text search of extracted text with ~~elasticsearch/elasticlunr/ag~~ sonic and ripgrep
- download closed-caption subtitles from Youtube and other video sites (TODO: submit the subtitle files to the full-text search index)
- try pulling dead sites from archive.org and other sources if original is down (https://github.com/hartator/wayback-machine-downloader)
- And more in the issues list…

IMPORTANT: *Please don't work on any of these major long-term tasks without contacting me first, work is already in progress for many of these, and I may have to reject your PR if it doesn't align with the existing work!*

Past Releases

To see how this spec has been scheduled / implemented / released so far, read these pull requests:

- v0.1.x pre-git-history (~2017)
- v0.2.x (~2018/12)
- v0.3.x (~2019/03)
- v0.4.x (~2019/04)
- v0.5.x (~2020/11)
- v0.6.x (~2021/03)

- `sabbatical / coding hiatus during 2022`
- v0.7.x (~2023/11)
- v0.8.x (~2024/05)
- v0.9.x up next...

UI / UX Improvements Planned

- https://github.com/ArchiveBox/ArchiveBox/issues/1358
- https://github.com/ArchiveBox/ArchiveBox/issues/1273
- https://github.com/ArchiveBox/ArchiveBox/issues/988
- https://github.com/ArchiveBox/ArchiveBox/issues/930

New Extractors Planned

- `gallery-dl`: https://github.com/ArchiveBox/ArchiveBox/issues/564
- `forum-dl`: https://github.com/ArchiveBox/ArchiveBox/issues/1368
- `scihub-dl`: https://github.com/ArchiveBox/ArchiveBox/issues/720
- `cad-dl`: https://github.com/ArchiveBox/ArchiveBox/issues/668
- `aria2`: https://github.com/ArchiveBox/ArchiveBox/issues/1355
- `podcast-archiver`: https://github.com/ArchiveBox/ArchiveBox/issues/1357
- `bdfr`: https://github.com/ArchiveBox/ArchiveBox/issues/778
- `cutycapt` screenshots: https://github.com/ArchiveBox/ArchiveBox/issues/253
- sourcemap downloader: https://github.com/ArchiveBox/ArchiveBox/issues/1291

ArchiveBox Developer Documentation: Contributing a New Extractor

And others we're considering for the future:

Social Media

- Instagram
  - https://github.com/instaloader/instaloader (instagram downloader)
  - https://github.com/althonos/InstaLooter (stale)
- Telegram
  - https://github.com/iyear/tdl (telegram downloader)
- TikTok
  - https://github.com/charmparticle/tiktokget (tiktok downloader using yt-dlp)
  - https://github.com/TerminalWarlord/TikTok-Downloader-Bot
  - https://github.com/n0l3r/tiktok-downloader
  - https://github.com/hansputera/tiktok-dl

  - https://github.com/naseif/tiktok-scraper
  - https://github.com/irevenko/tiktik
  - https://github.com/samirelanduk/tiktok-save
  - https://github.com/Dinoosauro/tiktok-to-ytdlp
  - https://github.com/krypton-byte/tiktok-downloader
- Twitter
  - https://github.com/HoloArchivists/twspace-dl (stale, twitter spaces archiver)

Video/Streams

- https://github.com/soimort/you-get
- https://github.com/lay295/TwitchDownloader
- https://github.com/ihabunek/twitch-dl
- https://github.com/iawia002/lux (generic video/audio downloader)
- https://github.com/wukko/cobalt (generic video/audio downloader)
- https://github.com/jaysonlong/webvideo-downloader (Bilibili, iQIYI, Tencent Video, MGTV and WeTV)
- https://github.com/spaam/svtplay-dl (comedy central, twitch, HBO, etc. video downloader)
- https://github.com/aajanki/yle-dl (Yle Areena Finnish broadcasting video downloader)
- https://github.com/WHTJEON/widevine-dl (encrypted widevine video downloader)

Audio/Music

- https://github.com/nathom/streamrip (Qobuz, Tidal, Deezer and SoundCloud)
- https://github.com/0xHJK/music-dl
- https://github.com/guanguans/music-dl
- https://github.com/CharlesPikachu/musicdl
- https://github.com/iheanyi/bandcamp-dl
- https://github.com/spotDL/spotify-downloader
- https://github.com/Shabinder/SpotiFlyer
- https://github.com/SathyaBhat/spotify-dl / https://github.com/SwapnilSoni1999/spotify-dl / https://github.com/dhruv-ahuja/spoti-dl
- https://github.com/vitiko98/qobuz-dl (Qobuz music downloader)
- https://github.com/akhilrex/podgrab (stale)
- https://github.com/yaronzz/Tidal-Media-Downloader-PRO (stale)
- https://github.com/flyingrub/scdl (stale)
- https://github.com/ravishi/rdio-dl (stale, Rdio song downloader)
- https://github.com/carlosflorencio/laracasts-downloader (stale?)

Photos/Images/Comics

- https://github.com/mikf/gallery-dl
- https://github.com/Bionus/imgbrd-grabber (generic image board downloader like gallery-dl)
- https://github.com/Xonshiz/comic-dl (comic, anime, manga, etc. downloader)
- https://github.com/justfoolingaround/animdl (anime downloader)
- https://github.com/metafates/mangal (manga downloader)
- https://github.com/boredazfcuk/docker-icloudpd (iCloud Photos downloader)
- https://github.com/Oshan96/monkey-dl (stale? anime downloader)
- https://github.com/QianyanTech/Image-Downloader (stale?)
- https://github.com/Xonshiz/anime-dl (stale?)

Text/Forums

- https://github.com/mikwielgus/forum-dl
- https://github.com/AndyTheFactory/newspaper4k
- https://github.com/AAndyProgram/SCrawler (Twitter, Reddit, Instagram, Threads, Facebook, Pinterest, nsfw sites downloader)
- https://github.com/extractus/article-extractor
- https://github.com/shadowmoose/RedditDownloader (stale?)
- https://github.com/aliparlakci/bulk-downloader-for-reddit (stale?)

MOOC/Educational Content

- https://github.com/coursera-dl/coursera-dl
- https://github.com/rand-net/khan-dl
- https://github.com/C0D3D3V/Moodle-DL
- https://github.com/r0oth3x49/acloud-dl
- https://github.com/Puyodead1/udemy-downloader
- https://github.com/PyJun/Mooc_Downloader (stale)
- https://github.com/yann0917/dedao-dl (stale, MOOC course downloader)
- https://github.com/coursera-dl/edx-dl (stale?)
- https://github.com/SigureMo/mooc-dl (stale?)
- https://github.com/calvinhobbes23/Skillshare-DL (stale)
- https://github.com/r0oth3x49/lynda-dl (stale, Lynda.com course downloader)

Re-Archiving / WARC Creation

- https://github.com/hartator/wayback-machine-downloader
- https://github.com/MiniGlome/Archive.org-Downloader
- https://github.com/ArchiveTeam/grab-site

- https://github.com/oduwsdl/archivenow
- https://github.com/wabarc/warcraft
- https://github.com/sul-dlss/wasapi-downloader
- https://github.com/KellyStathis/warc_downloader
- https://github.com/internetarchive/heritrix3
- https://github.com/AhmadIbrahiim/Website-downloader (wget wrapper)
- https://github.com/igrigorik/gharchive.org (stale? Github downloader)

Other

- https://github.com/KurtBestor/Hitomi-Downloader
- https://github.com/nilaoda/BBDown
- https://github.com/biliup/biliup
- https://github.com/yutto-dev/bilili
- https://github.com/nICEnnnnnnnLee/BilibiliDown
- https://github.com/matlink/gplaycli (Google Play store Android app downloader)
- https://github.com/AlphaSlayer1964/kemono-dl (Patreon, gumroad, etc. archiver)
- https://github.com/manga-download/hakuneko
- https://github.com/cancerian0684/dli-downloader (Digital Library of India ebook downloader)
- https://github.com/tusharbabbar/gaana-dl (gaana.com bollywood song downloader)
- https://github.com/rebane2001/matterport-dl (stale? virtual house tour downloader)

1.7.2 Changelog

If you're having an issue with a breaking change, or migrating your data between versions, open an issue to get help.

`ArchiveBox` was previously named `Pocket Archive Stream` and then `Bookmark Archiver`.

`THIS PAGE HAS BEEN MOVED`: See the releases page for versioned source downloads and full changelog.

Many thanks to our 100+ contributors and everyone in the web archiving community!

- v0.4.9 released
  - `pip install archivebox` https://pypi.org/project/archivebox/
  - `docker run archivebox/archivebox` https://hub.docker.com/r/archivebox/archivebox
  - https://archivebox.readthedocs.io/en/latest/
  - https://github.com/ArchiveBox/ArchiveBox/releases
- easy migration from previous versions

```
cd path/to/your/archive/folder
archivebox init
archivebox add 'https://example.com'
archivebox add 'https://getpocket.com/users/USERNAME/feed/all' --depth=1
```

- full transition to Django Sqlite DB with migrations (making upgrades between versions much safer now)
- maintains an intuitive and helpful CLI that's backwards-compatible with all previous archivebox data versions
- uses argparse instead of hand-written CLI system: see `archivebox/cli/archivebox.py`
- new subcommands-based CLI for `archivebox` (see below)
- new Web UI with pagination, better search, filtering, permissions, and more
- 30+ assorted bugfixes, new features, and tickets closed
- for more info, see: https://github.com/ArchiveBox/ArchiveBox/releases/tag/v0.4.9

- v0.2.4 released
- better archive corruption guards (check structure invariants on every parse & save)
- remove title prefetching in favor of new FETCH_TITLE archive method
- slightly improved CLI output for parsing and remote url downloading
- re-save index after archiving completes to update titles and urls
- remove redundant derivable data from link json schema
- markdown link parsing support
- faster link parsing and better symbol handling using a new compiled URL_REGEX

- v0.2.3 released
- fixed issues with parsing titles including trailing tags
- fixed issues with titles defaulting to URLs instead of attempting to fetch
- fixed issue where bookmark timestamps from RSS would be ignored and current ts used instead
- fixed issue where ONLY_NEW would overwrite existing links in archive with only new ones
- fixed lots of issues with URL parsing by using `urllib.parse` instead of hand-written lambdas
- ignore robots.txt when using wget (sssshhh don't tell anyone)
- fix RSS parser bailing out when there's whitespace around XML tags
- fix issue with browser history export trying to run ls on wrong directory

---

- v0.2.2 released
- Shaarli RSS export support
- Fix issues with plain text link parsing including quotes, whitespace, and closing tags in URLs
- add USER_AGENT to archive.org submissions so they can track archivebox usage
- remove all icons similar to archive.org branding from archive UI
- hide some of the noisier youtubedl and wget errors
- set permissions on youtubedl media folder
- fix chrome data dir incorrect path and quoting
- better chrome binary finding

- show which parser is used when importing links, show progress when fetching titles

---

- v0.2.1 released with new logo
- ability to import plain lists of links and almost all other raw filetypes
- WARC saving support via wget
- Git repository downloading with git clone
- Media downloading with youtube-dl (video, audio, subtitles, description, playlist, etc)

- v0.2.0 released with new name
- renamed from **Bookmark Archiver** -> **ArchiveBox**

- v0.1.0 released
- support for browser history exporting added with `./bin/archivebox-export-browser-history`
- support for chrome `--dump-dom` to output full page HTML after JS executes

---

- v0.0.3 released
- support for chrome `--user-data-dir` to archive sites that need logins
- fancy individual html & json indexes for each link
- smartly append new links to existing index instead of overwriting

---

- v0.0.2 released
- proper HTML templating instead of format strings (thanks to https://github.com/bardisty!)
- refactored into separate files, wip audio & video archiving

---

- v0.0.1 released
- Index links now work without nginx url rewrites, archive can now be hosted on github pages
- added setup.sh script & docstrings & help commands
- made Chromium the default instead of Google Chrome (yay free software)
- added env-variable configuration (thanks to https://github.com/hannah98!)
- renamed from **Pocket Archive Stream** -> **Bookmark Archiver**
- added Netscape-format export support (thanks to https://github.com/ilvar!)
- added Pinboard-format export support (thanks to https://github.com/sconeyard!)
- front-page of HN, oops! apparently I have users to support now :grin:?
- added Pocket-format export support

- v0.0.0 released: created Pocket Archive Stream 2017/05/05

1.7.3 Supporting Development

ArchiveBox operates as a US 501(c)(3) nonprofit, donations are tax-deductible.
(ArchiveBox is fiscally sponsored by HCB EIN: 81-2908499)

Donations directly support ArchiveBox open-source development.

- **Direct donation via our `501(c)(3)` non-profit:** https://hcb.hackclub.com/donations/start/archivebox
- **Github Sponsors:** https://github.com/sponsors/pirate
- **Buy some swag:** https://archivebox-shop.fourthwall.com/
- **Patreon:** https://www.patreon.com/theSquashSH
- **Paypal:** https://paypal.me/NicholasSweeting
- **Crypto:** *Contact me to request wallet address...*

If you have any questions or want to partner with this project, contact me at: `donations-hello @ archivebox . io`.

1.7.4 Web Archiving Community

Join us on our new ArchiveBox community chat server: https://Zulip.ArchiveBox.io

Just getting started and want to learn more about why Web Archiving is important? Check out this article: On the Importance of Web Archiving.

The internet archiving community is surprisingly far-reaching and almost universally friendly! It has some overlap with the scraping and OSINT worlds, but it's also kinda its own thing.

Whether you want to learn which organizations are the big players in the web archiving space, want to find a specific open source tool for your web archiving need, or just want to see where archivists hang out online, this is my attempt at an index of the entire web archiving community. I cant promise that this list is up-to-date, the bulk of it was written in ~2022.

- *The Master Lists*
  Community-maintained indexes of web archiving tools and groups by IIPC, COPTR, ArchiveTeam, Wikipedia, & the ASA.
- *Web Archiving Software*
  Open source tools and projects in the internet archiving space.
  - *Bookmarking Services*
  - *Well-Known Open Source Projects*
  - *Public Archiving Services*
  - *ArchiveBox Alternatives*
  - *Smaller Utilities*
- *Reading List*
  Articles, posts, and blogs relevant to ArchiveBox and web archiving in general.
  - *Blogs*
  - *Articles*
  - *ArchiveBox-Specific Posts, Tutorials, and Guides*
  - *ArchiveBox Discussions in News & Social Media*

- *Communities*
  A collection of the most active internet archiving communities and initiatives.
  - *Most Active Web-Archiving Communities*
  - *Other Web Archiving Communities*
  - *General Archiving Foundations, Coalitions, Initiatives, and Institutes*

The Master Lists

Indexes of archiving institutions and software maintained by other people. If there's anything archivists love doing, it's making lists.

- **COPTR Wiki of Web Archiving Tools (COPTR)**
- **Awesome Web Archiving Tools (IIPC)**
- **My up-to-date list of starred archiving github projects**
- Spreadsheet Comparison of Archiving Tools (DataTogether)
- Awesome Web Crawling Tools
- Awesome Web Scraping Tools
- ArchiveTeam's List of Software (ArchiveTeam.org)
- List of Web Archiving Initiatives (Wikipedia.org)
- Directory of Archiving Organizations (American Society of Archivists)

Web Archiving Projects

Bookmarking Services

- **Linkwarden** Modern bookmarking UI with singlefile archiving
- Hoarder
- **Gosuki A lightweight, open-source, privacy-first bookmark manager that unifies bookmarks across multiple browsers**
- ~Pocket Premium Bookmarking tool that provides an archiving service in their paid version, run by Mozilla~
- **Pinboard Bookmarking tool that provides archiving in a paid version, run by a single independent developer**
- **Raindrop Bookmarking tool with archiving in their paid version, run by a company est. 2011**
- Instapaper Bookmarking alternative to Pocket/Pinboard (with no archiving)
- Wallabag / Wallabag.it Self-hostable web archiving server that can import via RSS
- Shaarli Self-hostable bookmark tagging, archiving, and sharing service
- ReadWise A paid Pocket/Pinboard alternative that includes article snippet and highlight saving
- Diigo Another brookmarking/annotation service with archiving as a paid feature

---

From the Archive.org & Archive-It teams

- **Archive.org The O.G. Wayback Machine provided publicly by the Internet Archive (Archive.org)**
- **Archive.it commercial Wayback Machine solution**
- **Heritrix The king of internet archiving crawlers, powers the Wayback Machine**
- **Brozzler chrome headless crawler + WARC archiver maintained by Archive.org**
- WarcProx WARC proxy recording and playback utility
- WarcTools utilities for dealing with WARCs
- Grab-Site An easy preconfigured web crawler designed for backing up websites
- WPull A pure python implementation of wget with WARC saving
- More on their GitHub…

From Webrecorder

Webrecorder develops a suite of open source tools, to capture websites and replay them at a later time as accurately as possible. Webrecorder also publishes the WACZ file format spec.

- **Browsertrix** Fully integrated (self hostable) SaaS web archiving platform
- **ArchiveWeb.page** Chrome extension for manual, interactive archiving of websites as you browse the web. Good for capturing high-fidelity complex interactions
- **ReplayWeb.page** Web archive viewer that runs entirely in the browser and doesn't require any server-hosted component to view WARC and WACZ files. Also available as a standalone electron app for local desktop use
- **Browsertrix Crawler** Command-line crawling application that powers Browsertrix's core crawling features
- pywb aka *Python Wayback*, the open source toolkit forked from archive.org for self-hosting your own wayback machine among other web archiving tools
- warcit Create a WARC file out of a folder full of assets
- warcio fast streaming asynchronous WARC reader and writer
- More on their GitHub…

From Rhizome.org (Conifer)

- **Conifer by Rhizome.org An open-source personal archiving server that uses pywb under the hood.** Previously affiliated with Webrecorder

---

From the Old Dominion University: Web Science Team

- **ipwb A distributed web archiving solution using pywb with IPFS for storage**
- **archivenow tool that pushes urls into all the online archive services like Archive.is and Archive.org**
- node-warc Parse And Create Web ARChive (WARC) files with node.js
- WAIL Web archiver GUI using Heritrix and OpenWayback

- Squidwarc User-scriptable, archival crawler using Chrome
- WAIL (Electron) Electron app version of the original wail for creating and interacting with web archives
- **warcreate a Chrome extension for creating WARCs from any webpage**
- More on their GitHub…

From the Archives Unleashed Team

- AUT Archives Unleashed Toolkit for analyzing web archives (formerly WarcBase)
- Warclight A Rails engine for finding and searching web archives
- More on their GitHub…

From the IIPC team

- **OpenWayback Open source project developing core Wayback Machine components**
- **awesome-web-archiving Large list of archiving projects and orgs**
- JWARC A Java library for reading and writing WARC files.
- More on their GitHub…

Other Public Archiving Services

- https://archive.is / https://archive.today
- https://ghostarchive.org
- https://perma.cc
- https://arquivo.pt
- https://www.pagefreezer.com
- https://www.smarsh.com
- https://www.stillio.com
- https://archive.st
- https://theoldnet.com/
- https://timetravel.mementoweb.org/
- https://freezepage.com/
- https://webcitation.org/archive
- https://archiveofourown.org/
- https://megalodon.jp/
- https://www.webarchive.org.uk/ukwa/
- https://github.com/HelloZeroNet/ZeroNet (super cool project)
- Google, Bing, DuckDuckGo, and other search engine caches

Other ArchiveBox Alternatives

*There are **much more recent** projects listed here: https://github.com/stars/pirate/lists/internet-archiving*

- **Browsertrix + ArchiveWeb.page + ReplayWeb.page Webrecorder's archiving suite has the highest fidelity, and can flawlessly archive YouTube, X, Facebook, and other complex, JS-heavy SPAs**
- **SingleFile Web Extension / CLI util for Firefox and Chrome to save a web page as a single HTML file**
- **Memex by Worldbrain.io a beautiful, user-friendly browser extension that archives all history with full-text search, annotation support, and more**
- **Hypothes.is a web/pdf/ebook annotation tool that also archives content**
- **Reminiscence extremely similar to ArchiveBox, uses a Django backend + UI and provides auto-tagging and summary features with NLTK**
- **Shaarchiver very similar project that archives Firefox, Shaarli, or Delicious bookmarks and all linked media, generating a markdown/HTML index**
- **Archivy Python-based self-hosted knowledge base embedded into your filesystem**
- **Polarized a desktop application for bookmarking, annotating, and archiving articles offline**
- **LinkWarden Link archival and curation web app, very similar to ArchiveBox**
- **Photon a fast crawler with archiving and asset extraction support**
- **Scoop** Create high-fidelity WARC/WACZ captures using a playwright browser, with support for signing, media extraction, PDFs, etc. (by the Perma.cc team)

Ones I haven't personally vetted:

- Shiori Simple bookmark manager + readability archiver built with Go (like a clone of Pocket)
- Percollate A command-line tool to turn web pages into beautiful, readable PDF, EPUB, or HTML docs.
- LinkAce A self-hosted bookmark management tool that saves snapshots to archive.org
- LinkDing Self-hosted bookmark manager that is designed be to be minimal, fast, and easy to set up using Docker.
- LinkWallet A self-hosted bookmark database with full-text page content search and limited archiving features
- Espial Bookmark manager and search tool with limited archiving features
- Diskernet Archiving tool that uses the Chrome debugger protocol to save each page as-loaded in the browser** (aka 22120 by c0fe or i5ik)
- Trilium Personal web UI based knowledge-base with web clipping and note-taking
- Herodotus Django-based web archiving tool with a focus on collecting text-based content
- Buku Browser-independent bookmark manager CLI written in Python3 and SQLite3
- ReadableWebProxy A proxying archiver that downloads content from sites and can snapshot multiple versions of sites over time
- Perkeep "Perkeep lets you permanently keep your stuff, for life."
- Fossilo A commercial archiving solution that appears to be very similar to ArchiveBox
- NeonLink Simple self-hosted bookmark management + Benotes note-taking app with limited archiving features
- Archivematica web GUI for institutional long-term archiving of web and other content
- Headless Chrome Crawler distributed web crawler built on puppeteer with screenshots
- WWWofle old proxying recorder software similar to ArchiveBox

- Erised Super simple CLI utility to bookmark and archive webpages
- Zotero collect, organize, cite, and share research (mainly for technical/scientific papers & citations)
- TiddlyWiki Non-linear bookmark and note-taking tool with archiving support
- Joplin Desktop + mobile app for knowledge-base-style info collection and notes (w/ optional plugin for archiving)
- Hunchly A paid web archiving / session recording tool design for OSINT
- Monolith CLI tool for saving complete web pages as a single HTML file
- Obelisk Go package and CLI tool for saving web page as single HTML file
- Munin Archiver Social media archiver for Facebook, Instagram and VKontakte accounts.
- **Wayback Archiving in style like ArchiveBox, but with a chat.**

Smaller Utilities

Random helpful utilities for web archiving, WARC creation and replay, and more…

- https://github.com/TheCakeIsNaOH/xbs-to-archivebox A utility to sync xBrowserSync bookmarks with ArchiveBox
- https://github.com/karlicoss/promnesia A browser extension that collects and collates all the URLs you visit into a hierarchical/graph structure with metadata
- https://github.com/vrtdev/save-page-state A Chrome extension for saving the state of a page in multiple formats
- https://github.com/jsvine/waybackpack command-line tool that lets you download the entire Wayback Machine archive for a given URL
- https://github.com/hartator/wayback-machine-downloader Download an entire website from the Internet Archive Wayback Machine.
- https://github.com/Lifesgood123/prevent-link-rot Replace any broken URLs in some content with Wayback machine URL equivalents
- https://en.archivarix.com download an archived page or entire site from the Wayback Machine
- https://proofofexistence.com prove that a certain file existed at a given time using the blockchain
- https://github.com/chfoo/warcat for merging, extracting, and verifying WARC files
- https://github.com/mozilla/readability tool for extracting article contents and text
- https://github.com/mholt/timeliner All your digital life on a single timeline, stored locally
- https://github.com/wkhtmltopdf/wkhtmltopdf Webkit HTML to PDF archiver/saver
- Sheetsee-Pocket project that provides a pretty auto-updating index of your Pocket links (without archiving them)
- Pocket -> IFTTT -> Dropbox Post by Christopher Su on his Pocket saving IFTTT recipe
- http://squidman.net/squidman/index.html
- https://wordpress.org/plugins/broken-link-checker/
- https://github.com/ArchiveTeam/wpull
- http://freedup.org/
- https://en.wikipedia.org/wiki/Furl

- https://preservica.com/digital-archive-software-1/active-digital-preservation For-profit company offering a digital preservation software suite
- https://github.com/karlicoss/grasp capture webpages from Firefox and Chrome into Org-mode documents
- https://github.com/dgtlmoon/changedetection.io Change detection and monitoring of web page content changes
- *And many more on the other lists…*

Reading List

A collection of blog posts and articles about internet archiving, contact me / open an issue if you want to add a link here!

Blogs Friends of ArchiveBox

- https://blog.archive.org
- https://webrecorder.net/blog
- https://netpreserveblog.wordpress.com
- https://blog.conifer.rhizome.org/
- https://ws-dl.blogspot.com
- https://siarchives.si.edu/blog
- https://parameters.ssrc.org
- https://sr.ithaka.org/publications
- https://ait.blog.archive.org
- https://brewster.kahle.org
- https://ianmilligan.ca
- https://medium.com/@giovannidamiola

Articles We Like About Internet Archiving

- https://items.ssrc.org/parameters/on-the-importance-of-web-archiving/
- https://theconversation.com/your-internet-data-is-rotting-115891
- https://www.bbc.com/future/story/20190401-why-theres-so-little-left-of-the-early-internet
- https://sr.ithaka.org/publications/the-state-of-digital-preservation-in-2018/
- https://gizmodo.com/delete-never-the-digital-hoarders-who-collect-tumblrs-1832900423
- https://siarchives.si.edu/blog/we-are-not-alone-progress-digital-preservation-community
- https://www.gwern.net/Archiving-URLs
- http://brewster.kahle.org/2015/08/11/locking-the-web-open-a-call-for-a-distributed-web-2/
- https://lwn.net/Articles/766374/

- https://en.wikipedia.org/wiki/List_of_Web_archiving_initiatives
- https://medium.com/@giovannidamiola/making-the-internet-archives-full-text-search-faster-30fb11574ea9
- https://xkcd.com/1909/
- https://samsaffron.com/archive/2012/06/07/testing-3-million-hyperlinks-lessons-learned#comment-31366
- https://www.gwern.net/docs/linkrot/2011-muflax-backup.pdf
- https://thoughtstreams.io/higgins/permalinking-vs-transience/
- http://ait.blog.archive.org/files/2014/04/archiveit_life_cycle_model.pdf
- https://blog.archive.org/2016/05/26/web-archiving-with-national-libraries/
- https://blog.archive.org/2014/10/28/building-libraries-together/
- https://ianmilligan.ca/2018/03/27/ethics-and-the-archived-web-presentation-the-ethics-of-studying-geocities/
- https://ianmilligan.ca/2018/05/22/new-article-if-these-crawls-could-talk-studying-and-documenting-web-archives-provenance/
- https://ws-dl.blogspot.com/2019/02/2019-02-08-google-is-being-shuttered.html

If any of these links are dead, you can find an archived version on https://archive.sweeting.me or https://web.archive.org.

ArchiveBox-Specific Posts, Tutorials, and Guides

Beware: many of these may be outdated, as ArchiveBox has frequent updates and continual improvement.

- "Install ArchiveBox on SaltBox.dev" https://docs.saltbox.dev/sandbox/apps/archivebox/#3-setup
- "ArchiveBox is an open-source self-hosted web archiving system for the web and the desktop" https://medevel.com/archivebox/
- "Install ArchiveBox on a One-Click Docker Application" https://www.vultr.com/docs/install-archivebox-on-a-oneclick-docker-application/
- "ArchiveBox, una solución para crear nuestro propio Archive.org en miniatura y personalizado" https://www.genbeta.com/herramientas/archivebox-solucion-para-crear-nuestro-propio-archive-org-miniatura-personalizado
- "ArchiveBoxPython" https://www.bilibili.com/s/video/BV1ib4y1X7SL
- " - " https://habr.com/ru/company/vdsina/blog/550180/
- "ArchiveBox, una solución para crear nuestro propio Archive.org en miniatura y personalizado" https://www.genbeta.com/herramientas/archivebox-solucion-para-crear-nuestro-propio-archive-org-miniatura-personalizado
- "Preserve the Internet With ArchiveBox" https://www.cyberpunks.com/preserve-the-internet-with-archivebox/
- " . ArchiveBox" https://xakep.ru/2021/02/01/archivebox/
- "Internet" http://www.diglog.com/story/1045192.html
- "How to Make Your Own Internet Archive With ArchiveBox" https://nixintel.info/osint-tools/make-your-own-internet-archive-with-archive-box/
- "Mit ArchiveBox Webseiten auf der Festplatte archivieren" https://www.linux-community.de/ausgaben/linuxuser/2020/12/mit-archivebox-webseiten-auf-der-festplatte-archivieren/
- "ArchiveBoxWEB" https://zhen.bushini.de/14738.html / https://www.1fishsauce.com/?p=4206
- ": Kiwix & ArchiveBox" https://blog.csdn.net/JackLang/article/details/108328791
- "" https://www.pcpc.me/tech/self-hosted-read-later-app

- "How to install ArchiveBox to preserve websites you care about" https://blog.sleeplessbeastie.eu/2019/06/19/how-to-install-archivebox-to-preserve-websites-you-care-about/
- "How to remotely archive websites using ArchiveBox" https://blog.sleeplessbeastie.eu/2019/06/26/how-to-remotely-archive-websites-using-archivebox/
- "How to Create Your Own Private Self-Hosted Read-It-Later App" https://www.makeuseof.com/tag/self-hosted-read-later-app/
- "How to use CutyCapt inside ArchiveBox" https://blog.sleeplessbeastie.eu/2019/07/10/how-to-use-cutycapt-inside-archivebox/
- "Automate ArchiveBox with Google Spreadsheet to Backup your internet" https://manfred.life/archivebox
- "♪ConoHaArchiveBox" https://qiita.com/CloudRemix/items/691caf91efa3ef19a7ad
- "WEB-ARCHIV TEIL 8: WALLABAG UND ARCHIVEBOX" http://webermartin.net/blog/web-archiv-teil-8-wallabag-und-archivebox/
- https://metaxyntax.neocities.org/entries/7.html

ArchiveBox Discussions in News & Social Media

- **Aggregators:**
  ProductHunt, **AlternativeTo**, **SaaSHub**, Logiciels, SteemHunt, Recurse Center: The Joy of Computing, GitHub Changelog, Dev.To Ultra List, O'Reilly 4 Short Links, JaxEnter
- **Blog Posts & Podcasts:**
  Korben.info, Defining Desktop Linux Podcast #296 (0:55:00), Binärgewitter Podcast #221, Schrankmonster.de, La Ferme Du Web
- **Hacker News threads and comments:**
  #1, #2, #3, **#4**, and many more…
- **Reddit r/DataHoarder, r/SelfHosted, etc. posts and comments**:
  #1, #2, #3, #4, #5 , **#6**, #7, #8, and many more…
- **Twitter:**
  Python Trending, PyCoder's Weekly, Python Hub, Smashing Magazine, and many more…

Communities

Most Active Communities

- **The Internet Archive (Archive.org)** (USA)
- **International Internet Preservation Consortium (IIPC)** (International)
- **The Archive Team, URL Team, r/ArchiveTeam** (International)
- **Rhizome.org** The digital preservation group that works on Conifer by Rhizome formerly Webrecorder.io (USA)
- **Webrecorder** (formerly known[1] as Webrecorder.io) is a company led by Ilya Kreymer, that researches and develops web archiving tools, widely used by the community.
- **Old Dominion University: Web Science and Digital Libraries (WS-DL @ ODU)** (Virginia, USA)
- **r/DataHoarder, r/Archivists, r/DHExchange** (International)
- The Eye Non-profit working on content archival and long-term preservation (Europe)
- Digital Preservation Coalition & their Software Tool Registry (COPTR) (UK & Wales)

- Archives Unleashed Project and UAP GitHub (Canada)

Web Archiving Communities

Follow these technological and organizational archiving hubs for the latest archiving news.

- Canadian Web Archiving Coalition (Canada)
- Web Archives for Historical Research Group (Canada)
- Smithsonian Institution Archives: Digital Curation (Washington D.C., USA)
- National Digital Stewardship Alliance (NDSA) (USA)
- Digital Library Federation (DLF) (USA)
- Council on Library and Information Resources (CLIR) (USA)
- Digital Curation Centre (DCC) (UK)
- ArchiveMatica & their Community Wiki (International)
- Professional Development Institutes for Digital Preservation (POWRR) (USA)
- Institute of Museum and Library Services (IMLS) (USA)
- Stanford Libraries Web Archiving (USA)
- Society of American Archivists: Electronic Records (SAA) (USA)
- BitCurator Consortium (BCC) (USA)
- Ethics & Archiving the Web Conference (Rhizome) (USA)
- Archivists Round Table of NYC (USA)

General Archiving Foundations, Coalitions, Initiatives, and Institutes

Find your local archiving group in the list and see how you can contribute!

- Community Archives and Heritage Group (UK & Ireland)
- Open Preservation Foundation (OPF) (UK & Europe)
- Software Preservation Network (International)
- ITHAKA, Portico, JSTOR, ARTSTOR, S+R (USA)
- Archives and Records Association (UK & Ireland)
- Arkivrådet (Sweden)
- Asociación Española de Archiveros, Bibliotecarios, Museologos y Documentalistas (ANABAD) (Spain)
- Associação dos Arquivistas Brasileiros (AAB) (Brazil)
- Associação Portuguesa de Bibliotecários, Archivistas e Documentalistas (BAD) (Portugal)
- Association des archivistes français (AAF) (France)
- Associazione Nazionale Archivistica Italiana (ANAI) (Italy)
- Australian Society of Archivists Inc. (Australia)

- International Council on Archives (ICA)
- International Records Management Trust (IRMT)
- Irish Society for Archives (Ireland)
- Koninklijke Vereniging van Archivarissen in Nederland (Netherlands)
- State Archives Administration of the People's Republic of China (China)
- Academy of Certified Archivists
- Archivists and Librarians in the History of the Health Sciences
- Archivists for Congregations of Women Religious
- Archivists of Religious Institutions
- Association of Catholic Diocesan Archivists
- Association of Moving Image Archivists
- Council of State Archivists
- National Association of Government Archives and Records Administrators
- National Episcopal Historians and Archivists
- Archival Education and Research Institute
- Archives Leadership Institute
- Georgia Archives Institute
- Modern Archives Institute
- Western Archives Institute
- Association des archivistes du Québec
- Association of Canadian Archivists
- Canadian Council of Archives/Conseil canadien des archives
- Archives Association of British Columbia
- Archives Association of Ontario
- Archives Council of Prince Edward Island
- Archives Society of Alberta
- Association for Manitoba Archives
- Association of Newfoundland and Labrador Archives
- Council of Nova Scotia Archives
- Réseau des services d'archives du Québec
- Saskatchewan Council for Archives and Archivists

You can find more organizations and initiatives on these other lists:

- Wikipedia.org List of Web Archiving Initiatives
- SAA List of USA & Canada Based Archiving Organizations
- SAA List of International Archiving Organizations
- Digital Preservation Coalition's Member List

ArchiveBox Community Resources

ArchiveBox Chat Rooms

- Official ArchiveBox Zulip Chat Server
- Unofficial ArchiveBox Matrix chat room (old)
- GitHub Discussions

ArchiveBox on Social Media

- Twitter: @ArchiveBoxApp
- LinkedIn: ArchiveBox
- YouTube: @ArchiveBoxApp
- Reddit: r/ArchiveBox
- Alternative.to
- ReposHub

ArchiveBox on Package Distribution Platforms

- Python PyPI
- Docker Hub
- ArchLinux AUR
- Ubuntu/Debian apt wrapper

---

^ Back to Top ^

PYTHON MODULE INDEX

a

archivebox, 92
archivebox.__main__, 378
archivebox.api, 167
archivebox.api.admin, 203
archivebox.api.apps, 167
archivebox.api.auth, 193
archivebox.api.middleware, 193
archivebox.api.models, 190
archivebox.api.urls, 171
archivebox.api.v1_api, 172
archivebox.api.v1_auth, 192
archivebox.api.v1_cli, 198
archivebox.api.v1_core, 175
archivebox.api.v1_crawls, 168
archivebox.api.v1_machine, 194
archivebox.api.v1_personas, 173
archivebox.api.webhooks, 171
archivebox.base_models, 349
archivebox.base_models.admin, 352
archivebox.base_models.apps, 349
archivebox.base_models.models, 349
archivebox.cli, 138
archivebox.cli.archivebox_add, 162
archivebox.cli.archivebox_archiveresult, 153
archivebox.cli.archivebox_binary, 144
archivebox.cli.archivebox_config, 152
archivebox.cli.archivebox_crawl, 139
archivebox.cli.archivebox_extract, 155
archivebox.cli.archivebox_help, 143
archivebox.cli.archivebox_init, 152
archivebox.cli.archivebox_install, 158
archivebox.cli.archivebox_list, 166
archivebox.cli.archivebox_machine, 147
archivebox.cli.archivebox_manage, 156
archivebox.cli.archivebox_mcp, 150
archivebox.cli.archivebox_oneshot, 152
archivebox.cli.archivebox_persona, 148
archivebox.cli.archivebox_pluginmap, 163
archivebox.cli.archivebox_process, 157
archivebox.cli.archivebox_remove, 138
archivebox.cli.archivebox_run, 141
archivebox.cli.archivebox_schedule, 141
archivebox.cli.archivebox_search, 141
archivebox.cli.archivebox_server, 161
archivebox.cli.archivebox_shell, 148
archivebox.cli.archivebox_snapshot, 159
archivebox.cli.archivebox_status, 158
archivebox.cli.archivebox_tag, 164
archivebox.cli.archivebox_update, 145
archivebox.cli.archivebox_version, 151
archivebox.cli.cli_util, 164
archivebox.cli.rescan, 156
archivebox.config, 321
archivebox.config.collection, 321
archivebox.config.common, 336
archivebox.config.configset, 331
archivebox.config.constants, 326
archivebox.config.django, 335
archivebox.config.ldap, 347
archivebox.config.paths, 333
archivebox.config.permissions, 323
archivebox.config.version, 348
archivebox.config.views, 329
archivebox.core, 215
archivebox.core.admin, 287
archivebox.core.admin_archiveresults, 223
archivebox.core.admin_site, 253
archivebox.core.admin_snapshots, 279
archivebox.core.admin_tags, 251
archivebox.core.admin_users, 285
archivebox.core.apps, 236
archivebox.core.asgi, 286
archivebox.core.context_processors, 242
archivebox.core.forms, 233
archivebox.core.management, 221
archivebox.core.management.commands, 221
archivebox.core.management.commands.archivebox, 221
archivebox.core.middleware, 276
archivebox.core.models, 259
archivebox.core.permissions, 238
archivebox.core.recovery_util, 242
archivebox.core.routes_util, 254

archivebox.core.settings, 227
archivebox.core.settings_logging, 236
archivebox.core.setup_wizard, 253
archivebox.core.shutdown_util, 249
archivebox.core.snapshot_status, 236
archivebox.core.sqlite_backend, 221
archivebox.core.sqlite_backend.base, 221
archivebox.core.tag_util, 277
archivebox.core.takeover_util, 240
archivebox.core.templatetags, 215
archivebox.core.templatetags.core_tags, 215
archivebox.core.urls, 248
archivebox.core.views, 242
archivebox.core.widgets, 226
archivebox.core.wsgi, 285
archivebox.crawls, 308
archivebox.crawls.admin, 316
archivebox.crawls.apps, 308
archivebox.crawls.locks, 315
archivebox.crawls.models, 309
archivebox.crawls.schedule_util, 309
archivebox.ldap, 117
archivebox.ldap.apps, 118
archivebox.ldap.auth, 118
archivebox.machine, 92
archivebox.machine.admin, 112
archivebox.machine.apps, 92
archivebox.machine.detect, 93
archivebox.machine.env_util, 111
archivebox.machine.models, 94
archivebox.manage, 378
archivebox.mcp, 294
archivebox.mcp.apps, 294
archivebox.mcp.server, 295
archivebox.misc, 119
archivebox.misc.checks, 123
archivebox.misc.db, 124
archivebox.misc.jsonl, 132
archivebox.misc.logging, 120
archivebox.misc.logging_util, 121
archivebox.misc.monkey_patches, 134
archivebox.misc.paginators, 119
archivebox.misc.serve_static, 127
archivebox.misc.shell_welcome_message, 120
archivebox.misc.system, 119
archivebox.misc.toml_util, 119
archivebox.misc.util, 134
archivebox.personas, 298
archivebox.personas.admin, 307
archivebox.personas.apps, 304
archivebox.personas.forms, 303
archivebox.personas.importers, 298
archivebox.personas.models, 304
archivebox.plugins, 367
archivebox.plugins.apps, 370
archivebox.plugins.discovery, 371
archivebox.plugins.forms, 367
archivebox.plugins.hooks, 376
archivebox.plugins.output_groups, 370
archivebox.plugins.views, 375
archivebox.progressmonitor, 116
archivebox.progressmonitor.apps, 116
archivebox.progressmonitor.collection, 116
archivebox.progressmonitor.views, 117
archivebox.search, 288
archivebox.search.admin, 293
archivebox.search.apps, 288
archivebox.search.backends, 288
archivebox.search.config, 292
archivebox.search.query, 291
archivebox.search.sonic_daemon, 288
archivebox.search.views, 289
archivebox.services, 355
archivebox.services.archive_result_service, 358
archivebox.services.binary_service, 356
archivebox.services.crawl_service, 367
archivebox.services.machine_service, 366
archivebox.services.process_service, 355
archivebox.services.resource_admission, 359
archivebox.services.runner, 361
archivebox.services.snapshot_service, 364
archivebox.services.tag_service, 365
archivebox.uuid_compat, 377
archivebox.workers, 204
archivebox.workers.admin, 214
archivebox.workers.apps, 205
archivebox.workers.management, 204
archivebox.workers.management.commands, 204
archivebox.workers.management.commands.runner_watch, 204
archivebox.workers.management.commands.supervisord_watchdo 205
archivebox.workers.models, 212
archivebox.workers.supervisord_parent_watchdog, 212
archivebox.workers.supervisord_util, 206

INDEX

Symbols

_ACTIVE_WORKER_STATES (*in module archivebox.workers.supervisord_util*), 208
_AUDIO_FILE_EXTS (*in module archivebox.core.templatetags.core_tags*), 217
_BINARY_TEMPLATE_PATTERN (*in module archivebox.plugins.forms*), 369
_BROWSER_AUDIO_FILE_EXTS (*in module archivebox.core.templatetags.core_tags*), 218
_BROWSER_VIDEO_FILE_EXTS (*in module archivebox.core.templatetags.core_tags*), 217
_CURRENT_BINARIES (*in module archivebox.machine.models*), 95
_CURRENT_INTERFACE (*in module archivebox.machine.models*), 95
_CURRENT_MACHINE (*in module archivebox.machine.models*), 95
_CURRENT_PROCESS (*in module archivebox.machine.models*), 95
_FS_VERSION_MIGRATION_PATHS (*archivebox.core.models.Snapshot attribute*), 265
_GROUP_LABELS (*in module archivebox.plugins.output_groups*), 371
_HASHES_CACHE (*in module archivebox.misc.serve_static*), 128
_HEALTH_CHECK_INTERVAL_SECONDS (*in module archivebox.core.templatetags.core_tags*), 219
_HIGH_LOAD_MULTIPLE (*in module archivebox.core.templatetags.core_tags*), 219
_HIGH_MEMORY_THRESHOLD_PCT (*in module archivebox.core.templatetags.core_tags*), 219
_IMAGE_PREVIEW_EXTS (*in module archivebox.core.templatetags.core_tags*), 217
_INITIAL_SYNC_DONE (*in module archivebox.config.collection*), 322
_IPV4_RE (*in module archivebox.cli.archivebox_server*), 161
_IPV6_CHARS_RE (*in module archivebox.cli.archivebox_server*), 161
_IS_POSTGRES (*in module archivebox.misc.db*), 125
_LOCAL_BIND_HOSTS (*in module archivebox.cli.archivebox_server*), 161
_LOW_DISK_THRESHOLD_GB (*in module archivebox.core.templatetags.core_tags*), 219
_LifecycleLockState (*class in archivebox.crawls.locks*), 315
_MEDIA_FILE_EXTS (*in module archivebox.core.templatetags.core_tags*), 217
_MIRROR_IN_PROGRESS (*in module archivebox.config.collection*), 322
_PROCESS_STATE_ERRORS (*in module archivebox.workers.supervisord_util*), 208
_PROC_ACCESS_EXCEPTIONS (*in module archivebox.workers.supervisord_util*), 211
_PROC_INFO_EXCEPTIONS (*in module archivebox.workers.supervisord_util*), 211
_ROLE_SUBDOMAIN_LABELS (*in module archivebox.core.routes_util*), 256
_RUNTIME_COMPONENT_ORDER (*in module archivebox.workers.supervisord_util*), 208
_ReconfigurableStream (*class in archivebox*), 379
_SCOPE_CRAWL_EXECUTION (*in module archivebox.config.common*), 338
_SCOPE_CRAWL_FROZEN (*in module archivebox.config.common*), 338
_SCOPE_SERVER (*in module archivebox.config.common*), 338
_SENSITIVE_CONFIG_KEY_NEEDLES (*in module archivebox.config.common*), 338
_SNAPSHOT_ID_RE (*in module archivebox.core.routes_util*), 256
_SNAPSHOT_SUBDOMAIN_RE (*in module archivebox.core.routes_util*), 256
_STATIC_URL_SAFE (*in module archivebox.core.templatetags.core_tags*), 217
_STDERR_CONSOLE (*in module archivebox.config.common*), 338
_STDOUT_CONSOLE (*in module archivebox.config.common*), 338
_SUPERVISORD_ERRORS (*in module archivebox.workers.supervisord_util*), 208
_StreamingQueueWriter (*class in archivebox.misc.serve_static*), 129
_TEXT_PREVIEW_EXTS (*in module archive-*

box.core.templatetags.core_tags), 217
_URL_FRAGMENT_SAFE (*in module archivebox.core.routes_util*), 256
_URL_PATH_SAFE (*in module archivebox.core.routes_util*), 256
_VIDEO_FILE_EXTS (*in module archivebox.core.templatetags.core_tags*), 217
_WARNED_ARCHIVING_CONFIGS (*in module archivebox.config.common*), 338
__all__ (*in module archivebox*), 379
__all__ (*in module archivebox.config*), 349
__all__ (*in module archivebox.services*), 367
__all__ (*in module archivebox.uuid_compat*), 378
__author__ (*in module archivebox*), 379
__call__() (*archivebox.api.middleware.ApiCorsMiddleware method*), 193
__call__() (*archivebox.config.configset.IniConfigSettingsSource method*), 332
__call__() (*archivebox.misc.monkey_patches.ModifiedAccessLogGenerator method*), 134
__command__ (*in module archivebox.cli*), 166
__command__ (*in module archivebox.cli.archivebox_add*), 163
__command__ (*in module archivebox.cli.archivebox_archiveresult*), 154
__command__ (*in module archivebox.cli.archivebox_binary*), 144
__command__ (*in module archivebox.cli.archivebox_crawl*), 140
__command__ (*in module archivebox.cli.archivebox_extract*), 155
__command__ (*in module archivebox.cli.archivebox_help*), 143
__command__ (*in module archivebox.cli.archivebox_list*), 166
__command__ (*in module archivebox.cli.archivebox_machine*), 148
__command__ (*in module archivebox.cli.archivebox_mcp*), 151
__command__ (*in module archivebox.cli.archivebox_oneshot*), 152
__command__ (*in module archivebox.cli.archivebox_persona*), 149
__command__ (*in module archivebox.cli.archivebox_process*), 157
__command__ (*in module archivebox.cli.archivebox_remove*), 138
__command__ (*in module archivebox.cli.archivebox_run*), 142
__command__ (*in module archivebox.cli.archivebox_search*), 141
__command__ (*in module archivebox.cli.archivebox_snapshot*), 160
__command__ (*in module archivebox.cli.archivebox_tag*), 165
__contains__() (*archivebox.config.configset.BaseConfigSet method*), 333
__getattr__() (*archivebox.misc.logging.AttrDict method*), 121
__getattr__() (*in module archivebox*), 379
__getattr__() (*in module archivebox.config*), 349
__getitem__() (*archivebox.config.configset.BaseConfigSet method*), 333
__getitem__() (*archivebox.config.constants.ConstantsDict class method*), 329
__iter__() (*archivebox.misc.serve_static.RangedFileReader method*), 132
__license__ (*in module archivebox*), 379
__order__ (*in module archivebox.config*), 349
__order__ (*in module archivebox.core*), 287
__order__ (*in module archivebox.crawls*), 321
__order__ (*in module archivebox.workers*), 215
__setitem__() (*archivebox.config.configset.BaseConfigSet method*), 333
__setitem__() (*archivebox.misc.logging.AttrDict method*), 121
__str__() (*archivebox.api.models.APIToken method*), 191
__str__() (*archivebox.api.models.OutboundWebhook method*), 191
__str__() (*archivebox.base_models.models.ModelWithUUID method*), 350
__str__() (*archivebox.core.models.ArchiveResult method*), 274
__str__() (*archivebox.core.models.Snapshot method*), 264
__str__() (*archivebox.core.models.Tag method*), 260
__str__() (*archivebox.crawls.models.Crawl method*), 312
__str__() (*archivebox.crawls.models.CrawlSchedule method*), 310
__str__() (*archivebox.machine.models.Binary method*), 101
__str__() (*archivebox.machine.models.Process method*), 105
__str__() (*archivebox.personas.models.Persona method*), 306
__str__() (*archivebox.uuid_compat.CompactUUID method*), 377
__version__ (*in module archivebox*), 379
_abort_locked_database() (*in module archivebox.core.sqlite_backend.base*), 222
_active_shutdown_state (*in module archivebox.core.shutdown_util*), 250

_add_cors_headers() (*archivebox.api.middleware.ApiCorsMiddleware method*), 193
_admin_login_hint_cookie_domain() (*in module archivebox.core.middleware*), 277
_admin_login_redirect_or_forbidden() (*in module archivebox.core.views*), 244
_allowed_binproviders() (*archivebox.machine.models.Binary method*), 101
_ansi_re (*archivebox.core.settings_logging.StripANSIColorCodesFilter attribute*), 238
_apply_archive_replay_headers() (*in module archivebox.misc.serve_static*), 131
_apply_imported_user_agent() (*in module archivebox.personas.importers*), 302
_apply_transformed_html_preview_style() (*in module archivebox.misc.serve_static*), 131
_archivebox_config_input_names() (*in module archivebox.config.common*), 344
_archivebox_worker_environment() (*in module archivebox.workers.supervisord_util*), 208
_attach_archiveresult_summaries() (*archivebox.core.admin_snapshots.SnapshotChangeList method*), 281
_bare_re (*archivebox.core.settings_logging.StripANSIColorCodesFilter attribute*), 238
_base_public_snapshot_fields() (*archivebox.core.views.PublicIndexView method*), 247
_binary_event_json() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
_binary_row_dedupe_key() (*in module archivebox.cli.archivebox_version*), 151
_binary_sort_key() (*in module archivebox.config.views*), 330
_bind_host_looks_like_ip() (*in module archivebox.cli.archivebox_server*), 162
_binproviders_to_str() (*in module archivebox.services.binary_service*), 357
_build_archivebox_config_model() (*in module archivebox.config.common*), 345
_build_base_host() (*in module archivebox.core.routes_util*), 257
_build_base_url_for_host() (*in module archivebox.core.routes_util*), 258
_build_directory_zip_response() (*in module archivebox.misc.serve_static*), 129
_build_env() (*archivebox.machine.models.Process method*), 109
_build_filtered_snapshots_queryset() (*in module archivebox.cli.archivebox_update*), 146
_build_plugin_cards() (*archivebox.plugins.forms.PluginConfigFormMixin method*), 369
_build_plugin_config_field() (*archivebox.plugins.forms.PluginConfigFormMixin method*), 369
_build_required_binary_links() (*in module archivebox.plugins.forms*), 369
_build_required_binary_url_lookup() (*in module archivebox.plugins.forms*), 369
_build_snapshot_preview() (*in module archivebox.core.tag_util*), 279
_build_snapshot_preview_map() (*in module archivebox.core.tag_util*), 279
_build_snapshot_preview_url() (*in module archivebox.core.templatetags.core_tags*), 218
_build_snapshot_replay_response() (*in module archivebox.core.views*), 246
_build_url() (*in module archivebox.core.routes_util*), 259
_bus_name() (*in module archivebox.services.runner*), 362
_cache_policy() (*in module archivebox.misc.serve_static*), 129
_can_override_crawl_config() (*archivebox.core.views.AddView method*), 247
_canonical_binary_name() (*in module archivebox.machine.models*), 96
_canonical_hook_name() (*in module archivebox.core.recovery_util*), 242
_choose_interrupt_action() (*archivebox.services.runner.CrawlRunner method*), 362
_clean_replay_next() (*in module archivebox.core.views*), 244
_clean_timeout_seconds() (*archivebox.core.forms.AddLinkForm method*), 235
_cleanup_old_migration_dir() (*archivebox.core.models.Snapshot method*), 266
_coerce_from_str_dict() (*in module archivebox.config.collection*), 322
_coerce_output_file_size() (*archivebox.core.models.ArchiveResult static method*), 275
_coerce_output_file_size() (*in module archivebox.core.templatetags.core_tags*), 218
_coerce_plugin_config_value() (*in module archivebox.plugins.forms*), 369
_coerce_to_str_dict() (*in module archivebox.config.collection*), 322
_collect_input_urls() (*in module archivebox.cli.archivebox_add*), 163
_command_doc() (*in module archivebox.cli.archivebox_help*), 143
_commit() (*archivebox.core.sqlite_backend.base.DatabaseWrapper method*), 222

_completed_worker_loop() (*archivebox.services.process_service.ProcessService method*), 356
_config_value() (*archivebox.crawls.models.Crawl static method*), 314
_connection_in_transaction() (*in module archivebox.core.sqlite_backend.base*), 222
_content_type_base() (*in module archivebox.misc.serve_static*), 131
_core_config_classes() (*archivebox.config.common.ArchiveBoxBaseConfig class method*), 345
_core_field_scope() (*archivebox.config.common.ArchiveBoxBaseConfig class method*), 345
_count_media_files() (*in module archivebox.core.templatetags.core_tags*), 218
_count_selected_hooks() (*in module archivebox.services.runner*), 362
_crawl_frozen_keys() (*archivebox.config.common.ArchiveBoxBaseConfig class method*), 345
_crawl_limit_stop_reason() (*in module archivebox.services.snapshot_service*), 365
_crawl_runtime_keys() (*archivebox.config.common.ArchiveBoxBaseConfig class method*), 345
_create_archive_result_if_missing() (*archivebox.core.models.Snapshot method*), 268
_create_crawl_from_form() (*archivebox.core.views.AddView method*), 248
_create_live_ui() (*archivebox.services.runner.CrawlRunner method*), 363
_csrf_trusted_origins() (*in module archivebox.core.routes_util*), 256
_current_foreground_supervisord_watchdog_args() (*in module archivebox.workers.supervisord_util*), 210
_dates() (*in module archivebox.cli.rescan*), 156
_default_exit_code_for_unowned_process() (*in module archivebox.machine.models*), 96
_default_sqlite_journal_mode() (*in module archivebox.config.common*), 341
_delete_signal_registered (*archivebox.base_models.models.ModelWithOutputDir attribute*), 352
_derive_plugin_enabled_config() (*archivebox.config.common.ArchiveBoxBaseConfig method*), 345
_desired_supervisord_workers (*in module archivebox.workers.supervisord_util*), 208
_detect_fs_version_from_index() (*archivebox.core.models.Snapshot static method*), 267
_detect_process_type() (*archivebox.machine.models.Process class method*), 107
_directories() (*in module archivebox.cli.rescan*), 156
_discover_command() (*archivebox.mcp.server.MCPServer method*), 297
_discover_plugin_config_schemas() (*in module archivebox.config.common*), 344
_display_data_path() (*in module archivebox.cli.archivebox_init*), 153
_display_snapshot_title() (*in module archivebox.core.tag_util*), 279
_enable_requested_plugins() (*in module archivebox.services.runner*), 362
_encoded_responses_image_url() (*in module archivebox.misc.serve_static*), 130
_ensure_completed_worker() (*archivebox.services.process_service.ProcessService method*), 356
_ensure_unique_timestamp() (*archivebox.core.models.Snapshot class method*), 267
_env_path() (*in module archivebox.config.paths*), 334
_escape() (*archivebox.base_models.admin.KeyValueWidget method*), 354
_escape() (*archivebox.core.widgets.TagEditorWidget method*), 227
_existing_output_path() (*archivebox.core.models.ArchiveResult method*), 275
_exit_daemon_runner_on_signal() (*in module archivebox.cli.archivebox_run*), 142
_exit_on_migration_interrupt() (*in module archivebox.misc.checks*), 123
_explicit_plugin_enabled_keys() (*in module archivebox.config.common*), 344
_extract_markdown_candidate() (*in module archivebox.misc.serve_static*), 130
_extract_snapshot_title() (*in module archivebox.services.archive_result_service*), 358
_fallback_output_file_path() (*archivebox.core.models.ArchiveResult static method*), 275
_fallback_supervisord_process_from_db() (*in module archivebox.workers.supervisord_util*), 209
_fcntl (*in module archivebox.personas.models*), 305
_fields() (*in module archivebox.cli.rescan*), 156
_files_index_target() (*in module archivebox.core.views*), 244
_filter_snapshots_for_rss() (*in module archivebox.api.v1_core*), 184
_finalize_missing_process() (*archive-*

box.services.binary_service.ArchiveBoxBinaryService method), 357
_finalize_request_when_done() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
_find_best_output_file() (*archivebox.core.models.ArchiveResult static method*), 275
_find_existing_binary_for_reference() (*in module archivebox.machine.models*), 96
_find_parent_process() (*archivebox.machine.models.Process class method*), 107
_find_snapshot_by_ref() (*in module archivebox.core.views*), 244
_finish_warmup_snapshot() (*archivebox.services.runner.CrawlRunner method*), 363
_first_due_id() (*in module archivebox.services.runner*), 364
_format_binary_abspath() (*in module archivebox.cli.archivebox_version*), 151
_format_direntry_timestamp() (*in module archivebox.misc.serve_static*), 129
_format_object_count() (*archivebox.core.admin_site.ArchiveBoxAdmin static method*), 254
_format_output_line_for_display() (*archivebox.core.models.ArchiveResult static method*), 274
_format_process_duration_seconds() (*in module archivebox.machine.admin*), 112
_format_size_column() (*in module archivebox.core.admin_snapshots*), 280
_format_sql() (*in module archivebox.core.sqlite_backend.base*), 222
_format_toml() (*in module archivebox.cli.archivebox_config*), 152
_fs_current_version() (*archivebox.core.models.Snapshot static method*), 265
_fs_migrate_from_0_7_0_to_0_9_0() (*archivebox.core.models.Snapshot method*), 266
_fs_migrate_from_0_8_0_to_0_9_0() (*archivebox.core.models.Snapshot method*), 266
_fs_migrate_from_0_9_0_to_0_9_4() (*archivebox.core.models.Snapshot method*), 266
_fs_migrate_legacy_to_0_9_0() (*archivebox.core.models.Snapshot method*), 266
_fs_next_version() (*archivebox.core.models.Snapshot method*), 266
_get_archiveresult_upload_data() (*in module archivebox.api.v1_core*), 181
_get_archiveresult_upload_files() (*in module archivebox.api.v1_core*), 181
_get_archiveresult_upload_form_value() (*in module archivebox.api.v1_core*), 181
_get_archiveresult_upload_form_values() (*in module archivebox.api.v1_core*), 181
_get_collection_id() (*in module archivebox.config.paths*), 334
_get_config_options() (*archivebox.base_models.admin.KeyValueWidget method*), 354
_get_custom_config_overrides() (*archivebox.core.views.AddView method*), 247
_get_expected_hook_total() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
_get_favicon_data() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
_get_or_create_binary() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
_get_ordering_fields() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
_get_prefetched_results() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
_get_prefetched_tags() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
_get_process_binary_env_keys() (*in module archivebox.machine.models*), 96
_get_progress_stats() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
_get_replay_source_url() (*in module archivebox.core.admin_archiveresults*), 223
_get_search_indexing_plugins() (*in module archivebox.cli.archivebox_update*), 146
_get_snapshot_by_ref() (*in module archivebox.api.v1_core*), 181
_get_snapshot_crawl() (*in module archivebox.cli.archivebox_update*), 146
_get_snapshot_for_tag_edit() (*in module archivebox.api.v1_core*), 189
_has_live_archivebox_parent() (*in module archivebox.workers.supervisord_util*), 211
_has_replay_cookie() (*in module archivebox.core.views*), 244
_hash_for_path() (*in module archivebox.misc.serve_static*), 129
_health_cache (*in module archivebox.core.templatetags.core_tags*), 219
_index_responses_paths_for_html_images() (*in*

module archivebox.misc.serve_static), 130
_install_raw_binary_names() (*in module archivebox.cli.archivebox_install*), 158
_interrupt_hook() (*archivebox.services.runner.CrawlRunner method*), 362
_is_asgi_request() (*in module archivebox.misc.serve_static*), 131
_is_binary_event_key() (*in module archivebox.services.machine_service*), 366
_is_cookie_file_config_key() (*in module archivebox.config.common*), 338
_is_copyable_profile_entry() (*in module archivebox.personas.importers*), 300
_is_external_task_cancelled() (*in module archivebox.services.runner*), 362
_is_image_preview_path() (*in module archivebox.core.templatetags.core_tags*), 218
_is_inside_atomic() (*in module archivebox.core.sqlite_backend.base*), 222
_is_ipv4_literal() (*in module archivebox.cli.archivebox_server*), 162
_is_ipv6_literal() (*in module archivebox.cli.archivebox_server*), 162
_is_local_bind_host() (*in module archivebox.core.routes_util*), 257
_is_nonfatal_setup_hook() (*in module archivebox.services.runner*), 362
_is_risky_replay_document() (*in module archivebox.misc.serve_static*), 131
_is_root_snapshot_output_path() (*in module archivebox.core.templatetags.core_tags*), 218
_is_signal_interrupted_exit() (*in module archivebox.core.recovery_util*), 242
_is_sonic_process() (*in module archivebox.workers.supervisord_util*), 211
_is_supervisord_process() (*in module archivebox.workers.supervisord_util*), 211
_is_text_preview_path() (*in module archivebox.core.templatetags.core_tags*), 218
_is_web_subdomain_alias() (*in module archivebox.core.templatetags.core_tags*), 218
_iter_children() (*in module archivebox.personas.importers*), 300
_iter_required_binary_names() (*in module archivebox.plugins.forms*), 369
_iter_url_lines() (*archivebox.crawls.models.Crawl method*), 313
_iter_visible_files() (*in module archivebox.misc.serve_static*), 129
_json_for_inline_script() (*archivebox.core.widgets.TagEditorWidget method*), 227
_jsonish() (*in module archivebox.plugins.forms*), 368
_latest_response_match() (*in module archivebox.core.views*), 246
_latest_responses_root() (*in module archivebox.core.views*), 246
_latest_snapshot_for_url() (*archivebox.core.views.WebAddView method*), 248
_lazy_load() (*archivebox.cli.ArchiveBoxGroup class method*), 167
_legacy_bool() (*in module archivebox.config.common*), 338
_lifecycle_lock() (*in module archivebox.crawls.locks*), 316
_list_media_files() (*in module archivebox.core.templatetags.core_tags*), 218
_list_profile_names() (*in module archivebox.personas.importers*), 302
_live_progress_plugin_names() (*in module archivebox.progressmonitor.views*), 117
_live_supervisord_processes_from_db() (*in module archivebox.workers.supervisord_util*), 209
_load_auth_storage() (*in module archivebox.personas.importers*), 302
_load_config_file() (*archivebox.config.configset.IniConfigSettingsSource method*), 332
_load_file_config_dict() (*in module archivebox.config.collection*), 322
_load_hash_map() (*in module archivebox.misc.serve_static*), 129
_log_locked_database() (*in module archivebox.core.sqlite_backend.base*), 222
_looks_like_cdp_url() (*in module archivebox.personas.importers*), 302
_looks_like_markdown() (*in module archivebox.misc.serve_static*), 130
_looks_like_output_path() (*archivebox.core.models.ArchiveResult static method*), 275
_looks_like_profile_dir() (*in module archivebox.personas.importers*), 302
_machine_health_stats() (*in module archivebox.core.templatetags.core_tags*), 219
_manifest_metadata() (*in module archivebox.services.archive_result_service*), 358
_merge_archive_results_from_index() (*archivebox.core.models.Snapshot method*), 268
_merge_auth_storage() (*in module archivebox.personas.importers*), 302
_merge_cookie_dicts() (*in module archivebox.personas.importers*), 302
_merge_netscape_cookies() (*in module archivebox.personas.importers*), 302
_merge_snapshots() (*archivebox.core.models.Snapshot class method*),

268
_merge_tags_from_index() (*archivebox.core.models.Snapshot method*), 267
_merge_title_from_index() (*archivebox.core.models.Snapshot method*), 267
_migration_interrupt_message() (*in module archivebox.misc.checks*), 123
_migration_trees_match() (*archivebox.core.models.Snapshot static method*), 266
_mirror_file_to_machine_config() (*in module archivebox.config.collection*), 322
_mode_label() (*in module archivebox.personas.forms*), 303
_needs_django_for_lazy_import() (*archivebox.cli.ArchiveBoxGroup class method*), 167
_normalize_add_url() (*archivebox.core.views.WebAddView method*), 248
_normalize_base_url() (*in module archivebox.core.routes_util*), 256
_normalize_id() (*archivebox.core.widgets.TagEditorWidget method*), 227
_normalize_output_files() (*archivebox.core.models.ArchiveResult static method*), 275
_normalize_output_files() (*in module archivebox.core.templatetags.core_tags*), 218
_normalize_plugins_config_value() (*in module archivebox.config.common*), 346
_normalize_snapshot_title() (*in module archivebox.services.archive_result_service*), 358
_normalize_status() (*in module archivebox.services.archive_result_service*), 358
_normalize_title_candidate() (*archivebox.core.models.Snapshot static method*), 269
_normalize_uploaded_archiveresult_output_path() (*in module archivebox.api.v1_core*), 181
_normalize_uploaded_archiveresult_plugin() (*in module archivebox.api.v1_core*), 181
_observe_snapshot_memory() (*archivebox.services.runner.CrawlRunner method*), 363
_option_args() (*in module archivebox.mcp.server*), 297
_ordered_public_page_from_order_index() (*archivebox.core.views.PublicIndexView method*), 247
_original_request_url() (*in module archivebox.core.views*), 246
_parse_and_validate_bind_spec() (*in module archivebox.cli.archivebox_server*), 162
_parse_archiveresult_output_json() (*in module archivebox.api.v1_core*), 181
_parse_archiveresult_upload_int() (*in module archivebox.api.v1_core*), 181
_parse_netscape_cookies() (*in module archivebox.personas.importers*), 302
_parse_rss_before() (*in module archivebox.api.v1_core*), 184
_parse_value() (*archivebox.base_models.admin.KeyValueWidget method*), 354
_path_exists() (*in module archivebox.personas.importers*), 299
_path_is_dir() (*in module archivebox.personas.importers*), 299
_pattern_matches_url() (*archivebox.crawls.models.Crawl class method*), 313
_perf_span() (*in module archivebox.services.archive_result_service*), 358
_perf_trace() (*in module archivebox.services.archive_result_service*), 358
_persisted_overrides_for_request() (*in module archivebox.services.binary_service*), 357
_persona_ids_for_permissions() (*in module archivebox.core.permissions*), 239
_plugin_card_document_response() (*in module archivebox.core.views*), 245
_plugin_config_input_name() (*in module archivebox.plugins.forms*), 368
_plugin_config_model() (*in module archivebox.config.common*), 344
_plugin_config_properties() (*in module archivebox.config.common*), 344
_plugin_enabled_config_keys() (*in module archivebox.config.common*), 346
_plugin_field_scope() (*archivebox.config.common.ArchiveBoxBaseConfig class method*), 345
_plugin_full_preview_response() (*in module archivebox.core.views*), 245
_plugin_input_config() (*in module archivebox.config.common*), 344
_plugin_presentation() (*in module archivebox.plugins.output_groups*), 371
_plugin_sensitive_config_keys() (*in module archivebox.config.common*), 338
_plugin_user_config() (*in module archivebox.config.common*), 344
_plugin_user_config_value() (*in module archivebox.config.common*), 344
_prefetched_objects_cache (*archivebox.core.models.Snapshot attribute*), 263
_print_server_startup_warnings() (*in module archivebox.cli.archivebox_server*), 162

_private_snapshot_auth_redirect() (*in module archivebox.core.views*), 244
_proc_cmdline() (*in module archivebox.workers.supervisord_util*), 211
_process_cmd() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
_process_output_dir() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
_prompt_in_foreground() (*archivebox.services.runner.CrawlRunner method*), 362
_provider_names() (*in module archivebox.services.binary_service*), 357
_psutil (*in module archivebox.machine.models*), 95
_psycopg_connect() (*in module archivebox.misc.db*), 125
_public_tag_listing_enabled() (*in module archivebox.api.v1_core*), 190
_queue_archiveresult_snapshot_maintenance() (*in module archivebox.api.v1_core*), 181
_raise_test_error() (*in module archivebox.core.urls*), 248
_read_cgroup_limit() (*in module archivebox.workers.supervisord_util*), 208
_read_text_tail() (*in module archivebox.config.views*), 330
_reconcile_directory() (*in module archivebox.cli.rescan*), 156
_record_supervisord_process() (*in module archivebox.workers.supervisord_util*), 208
_recover_non_atomic_connection() (*in module archivebox.core.sqlite_backend.base*), 222
_redirect_to_changelist() (*archivebox.core.admin_tags.TagAdmin method*), 252
_refresh_lock (*in module archivebox.progressmonitor.collection*), 117
_registry (*in module archivebox.crawls.locks*), 315
_registry_lock (*in module archivebox.crawls.locks*), 315
_registry_pid (*in module archivebox.crawls.locks*), 315
_render_binary_abspath() (*in module archivebox.cli.archivebox_version*), 151
_render_config_file_content() (*in module archivebox.config.collection*), 322
_render_copy_block() (*in module archivebox.machine.admin*), 112
_render_directory_index() (*in module archivebox.misc.serve_static*), 130
_render_fallback_card() (*in module archivebox.core.templatetags.core_tags*), 218
_render_image_preview_document() (*in module archivebox.misc.serve_static*), 130
_render_markdown_document() (*in module archivebox.misc.serve_static*), 131
_render_markdown_fallback() (*in module archivebox.misc.serve_static*), 131
_render_row() (*archivebox.base_models.admin.KeyValueWidget method*), 354
_render_text_file_preview() (*in module archivebox.core.templatetags.core_tags*), 219
_render_text_preview() (*in module archivebox.core.templatetags.core_tags*), 218
_render_text_preview_document() (*in module archivebox.misc.serve_static*), 130
_render_thumbnail() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
_replace_url_query_params() (*in module archivebox.core.routes_util*), 259
_replay_auth_response() (*in module archivebox.core.views*), 244
_replay_cookie_name() (*in module archivebox.core.views*), 244
_replay_payload_is_valid() (*in module archivebox.core.views*), 244
_request_abort_from_signal() (*archivebox.services.runner.CrawlRunner method*), 362
_request_has_tag_autocomplete_access() (*in module archivebox.api.v1_core*), 190
_request_scheme() (*in module archivebox.core.setup_wizard*), 253
_require_available_memory() (*in module archivebox.workers.supervisord_util*), 208
_require_superuser() (*in module archivebox.api.auth*), 194
_resolve_archive_path() (*in module archivebox.misc.serve_static*), 129
_resolve_archiveresult_relpath() (*in module archivebox.core.views*), 245
_resolve_cookie_file_paths() (*in module archivebox.config.common*), 338
_resolve_install_targets() (*in module archivebox.cli.archivebox_install*), 158
_resolve_output_metadata() (*in module archivebox.services.archive_result_service*), 358
_resolve_requests() (*in module archivebox.cli.archivebox_extract*), 155
_resolve_required_binary_name() (*in module archivebox.plugins.forms*), 369
_resolve_section_for_key() (*in module archivebox.config.collection*), 322
_resolve_snapshot_output_file() (*in module*

archivebox.core.templatetags.core_tags), 218
_restore_dates() (*in module archivebox.cli.rescan*), 156
_retry_locked_database() (*in module archivebox.core.sqlite_backend.base*), 222
_rewrite_html_image_sources_for_request() (*in module archivebox.misc.serve_static*), 130
_rewrite_html_image_sources_to_responses() (*in module archivebox.misc.serve_static*), 130
_rollback() (*archivebox.core.sqlite_backend.base.DatabaseWrapper method*), 222
_root_host_from_listen() (*in module archivebox.core.routes_util*), 257
_run_abxpkg_host_binary() (*in module archivebox.machine.detect*), 94
_run_binary() (*in module archivebox.services.runner*), 363
_run_crawl_locked() (*in module archivebox.services.runner*), 363
_run_due_binary() (*in module archivebox.services.runner*), 364
_run_due_crawl_locked() (*in module archivebox.services.runner*), 364
_run_due_crawl_status() (*in module archivebox.services.runner*), 364
_run_due_snapshot_id() (*in module archivebox.services.runner*), 364
_run_due_snapshot_locked() (*in module archivebox.services.runner*), 364
_run_due_snapshot_query() (*in module archivebox.services.runner*), 364
_run_event_now() (*in module archivebox.services.runner*), 362
_run_install() (*in module archivebox.services.runner*), 364
_run_snapshot_requests() (*in module archivebox.cli.archivebox_extract*), 155
_runner_console_line() (*in module archivebox.services.runner*), 362
_runner_label() (*in module archivebox.services.runner*), 362
_runner_short_id() (*in module archivebox.services.runner*), 362
_runner_task_context() (*in module archivebox.services.runner*), 362
_safe_archive_relpath() (*in module archivebox.core.views*), 245
_safe_zip_stem() (*in module archivebox.misc.serve_static*), 129
_same_config_value() (*in module archivebox.plugins.forms*), 368
_sanitize_config() (*archivebox.machine.models.Machine class method*), 97
_sanitize_machine_config() (*in module archivebox.machine.models*), 96
_save_archiveresult_event_to_db() (*in module archivebox.services.archive_result_service*), 358
_save_completed_process_to_db() (*archivebox.services.process_service.ProcessService method*), 356
_scan_output_file_map() (*archivebox.core.models.ArchiveResult static method*), 275
_schedule_missing_finalize() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
_schema_for_prefix() (*archivebox.api.v1_api.NinjaAPIWithIOCapture method*), 173
_schema_types() (*in module archivebox.plugins.forms*), 368
_scheme_from_request() (*in module archivebox.core.routes_util*), 258
_scope (*archivebox.config.common.ArchivingConfig attribute*), 342
_scope (*archivebox.config.common.DatabaseConfig attribute*), 342
_scope (*archivebox.config.common.GeneralConfig attribute*), 340
_scope (*archivebox.config.common.SearchBackendConfig attribute*), 344
_scope (*archivebox.config.common.ServerConfig attribute*), 340
_scope (*archivebox.config.common.ShellConfig attribute*), 339
_scope (*archivebox.config.common.StorageConfig attribute*), 340
_scope (*archivebox.config.ldap.LDAPConfig attribute*), 347
_scope_by_key() (*archivebox.config.common.ArchiveBoxBaseConfig class method*), 345
_scoped_config() (*archivebox.config.common.ArchiveBoxBaseConfig method*), 345
_search_backends_cache (*in module archivebox.search.backends*), 289
_select_best_timestamp() (*archivebox.core.models.Snapshot static method*), 267
_serve_original_domain_replay() (*in module archivebox.core.views*), 246
_serve_responses_path() (*in module archivebox.core.views*), 246
_serve_snapshot_replay() (*in module archive-*

box.core.views), 246
_set_model_object_count() (*archivebox.core.admin_site.ArchiveBoxAdmin method*), 254
_set_transformed_response_headers() (*in module archivebox.misc.serve_static*), 131
_setup_django_for_lazy_import() (*archivebox.cli.ArchiveBoxGroup class method*), 167
_shell_join() (*in module archivebox.workers.supervisord_util*), 208
_should_update_snapshot_title() (*in module archivebox.services.archive_result_service*), 358
_snapshot_base_url_for_context() (*in module archivebox.core.templatetags.core_tags*), 218
_snapshot_directories() (*in module archivebox.cli.rescan*), 156
_snapshot_id() (*in module archivebox.core.templatetags.core_tags*), 218
_snapshot_url_for_context() (*in module archivebox.core.templatetags.core_tags*), 218
_snapshots_rss_response() (*in module archivebox.api.v1_core*), 184
_sonic_listeners() (*in module archivebox.workers.supervisord_util*), 211
_sonic_worker_bind_target() (*in module archivebox.workers.supervisord_util*), 209
_split_bind_spec() (*in module archivebox.cli.archivebox_server*), 162
_split_strip() (*in module archivebox.core.forms*), 233
_sql_index_location() (*in module archivebox.config.paths*), 335
_sqlite_lock_retry_interval() (*in module archivebox.core.sqlite_backend.base*), 222
_sqlite_lock_retry_timeout() (*in module archivebox.core.sqlite_backend.base*), 222
_static_snapshot_base_url() (*in module archivebox.core.templatetags.core_tags*), 218
_stop_older_supervisord_processes() (*in module archivebox.workers.supervisord_util*), 209
_stream_ranged_file_async() (*in module archivebox.misc.serve_static*), 129
_strip_to_binary_keys() (*in module archivebox.services.machine_service*), 366
_summarize_archiveresult_output_files() (*in module archivebox.api.v1_core*), 181
_supervisord_proc (*in module archivebox.workers.supervisord_util*), 208
_tag_style() (*archivebox.core.widgets.TagEditorWidget method*), 227
_terminate_process_tree() (*in module archivebox.workers.supervisord_util*), 211
_terminate_processes() (*archivebox.machine.admin.ProcessAdmin method*), 115
_tool_result() (*in module archivebox.mcp.server*), 297
_trusted_snapshot_frame_csp() (*in module archivebox.core.views*), 245
_ts_to_date_str() (*archivebox.core.models.Snapshot static method*), 271
_url_passes_filters() (*archivebox.core.models.ArchiveResult method*), 276
_uuid_ref_query() (*in module archivebox.api.v1_core*), 181
_visible_response_snapshots_for_domain() (*in module archivebox.core.views*), 245
_warn_background_cleanup() (*in module archivebox.workers.supervisord_util*), 208
_webhook_fields() (*in module archivebox.api.admin*), 203
_with_archiveresult_processes() (*in module archivebox.api.v1_core*), 181
_with_port() (*in module archivebox.core.routes_util*), 256
_worker_environment_value() (*in module archivebox.workers.supervisord_util*), 209
_worker_log_base_dir() (*in module archivebox.workers.supervisord_util*), 209
_wrap() (*archivebox.misc.logging.AttrDict class method*), 121
_write_archiveresult_files() (*in module archivebox.api.v1_core*), 181
_write_binary_index() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
_write_file_if_changed() (*in module archivebox.config.collection*), 322
_write_netscape_cookies() (*in module archivebox.personas.importers*), 302

A

abspath (*archivebox.api.v1_machine.BinarySchema attribute*), 197
abspath (*archivebox.machine.models.Binary attribute*), 100
abstract (*archivebox.base_models.models.ModelWithConfig.Meta attribute*), 351
abstract (*archivebox.base_models.models.ModelWithDeleteAfter.Meta attribute*), 351
abstract (*archivebox.base_models.models.ModelWithHealthStats.Meta attribute*), 351
abstract (*archivebox.base_models.models.ModelWithNotes.Meta attribute*), 350

abstract (*archivebox.base_models.models.ModelWithOutputDir.Meta attribute*), 352
abstract (*archivebox.base_models.models.ModelWithUUID.Meta attribute*), 350
abstract (*archivebox.workers.models.ModelWithQueue.Meta attribute*), 214
ABX_PLUGINS_DOCS_BASE_URL (*in module archivebox.plugins.views*), 375
ABX_PLUGINS_GITHUB_BASE_URL (*in module archivebox.plugins.views*), 375
ABXPKG_LIB_DIR (*archivebox.config.common.StorageConfig attribute*), 340
AcceleratedPaginator (*class in archivebox.misc.paginators*), 119
action (*archivebox.api.v1_core.SnapshotUpdateSchema attribute*), 183
action (*archivebox.api.v1_crawls.CrawlUpdateSchema attribute*), 169
action_form (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
ACTION_TOOLS (*in module archivebox.mcp.server*), 296
actions (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 225
actions (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
actions (*archivebox.core.admin_tags.TagAdmin attribute*), 252
actions (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
actions (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
actions (*archivebox.machine.admin.BinaryAdmin attribute*), 114
actions (*archivebox.machine.admin.MachineAdmin attribute*), 113
actions (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 114
actions (*archivebox.machine.admin.ProcessAdmin attribute*), 115
ACTIVE_STATE (*archivebox.core.models.ArchiveResult attribute*), 272
ACTIVE_STATE (*archivebox.core.models.Snapshot attribute*), 263
active_state (*archivebox.core.models.Snapshot attribute*), 263
ACTIVE_STATE (*archivebox.crawls.models.Crawl attribute*), 312
active_state (*archivebox.crawls.models.Crawl attribute*), 312
ACTIVE_STATE (*archivebox.machine.models.Binary attribute*), 100
active_state (*archivebox.machine.models.Binary attribute*), 101
ACTIVE_STATE (*archivebox.workers.models.ModelWithQueue attribute*), 213
ACTIVE_STATE_LEASE_SECONDS (*in module archivebox.workers.models*), 213
active_supervisord_runtime_components() (*in module archivebox.workers.supervisord_util*), 211
add (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 201
ADD (*archivebox.machine.models.Process.TypeChoices attribute*), 103
add() (*in module archivebox.cli.archivebox_add*), 163
add_arguments() (*archivebox.core.management.commands.archivebox.Command method*), 221
add_arguments() (*archivebox.workers.management.commands.runner_watch.Command method*), 205
add_arguments() (*archivebox.workers.management.commands.supervisord_watchdog.Com method*), 205
add_fieldsets (*archivebox.core.admin_tags.TagAdmin attribute*), 252
add_fieldsets (*archivebox.core.admin_users.CustomUserAdmin attribute*), 286
add_fieldsets (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
add_fieldsets (*archivebox.personas.admin.PersonaAdmin attribute*), 308
add_replay_image_fallback() (*in module archivebox.misc.serve_static*), 128
add_snapshot_counts() (*in module archivebox.core.tag_util*), 278
add_tag_ids() (*archivebox.core.models.Snapshot method*), 264
add_tags() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 285
add_url() (*archivebox.crawls.models.Crawl method*), 314
add_view() (*archivebox.crawls.admin.CrawlAdmin method*), 319
add_view() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
AddCommandSchema (*class in archivebox.api.v1_cli*), 199
additional_snapshot_slots() (*archivebox.services.resource_admission.ResourceAdmission method*), 360
AddLinkForm (*class in archivebox.core.forms*), 233
AddView (*class in archivebox.core.views*), 247
admin_actions() (*archive-*

box.core.admin_archiveresults.ArchiveResultAdmin method), 225
admin_actions() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
admin_base_url (*in module archivebox.core.settings*), 231
admin_base_url() (*in module archivebox.core.templatetags.core_tags*), 220
admin_change_url (*archivebox.base_models.models.ModelWithHealthStats property*), 351
admin_change_url (*archivebox.base_models.models.ModelWithUUID property*), 350
admin_data_index_view() (*archivebox.core.admin_site.ArchiveBoxAdmin method*), 254
ADMIN_DATA_VIEWS (*in module archivebox.core.settings*), 233
ADMIN_LOGIN_HINT_COOKIE (*in module archivebox.core.middleware*), 277
ADMIN_PASSWORD (*archivebox.config.common.ServerConfig attribute*), 341
admin_snapshot_search_stream_view() (*in module archivebox.search.views*), 291
admin_url (*archivebox.api.v1_core.TagSearchSnapshotSchema attribute*), 187
ADMIN_USERNAME (*archivebox.config.common.ServerConfig attribute*), 341
AdminCookieIsolationMiddleware() (*in module archivebox.core.middleware*), 277
advance_lifecycle() (*archivebox.core.models.Snapshot method*), 265
advance_lifecycle() (*archivebox.crawls.models.Crawl method*), 315
advance_lifecycle() (*archivebox.machine.models.Binary method*), 102
after (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 200
ALL_PLUGINS (*in module archivebox.core.settings*), 229
all_subcommands (*archivebox.cli.ArchiveBoxGroup attribute*), 167
allow_crawl_execution_config_fields (*archivebox.core.forms.AddLinkForm attribute*), 234
allow_crawl_execution_config_fields (*archivebox.plugins.forms.PluginConfigFormMixin attribute*), 369
allow_maintenance_on_inactive_crawl (*archivebox.services.runner.CrawlRunner property*), 362
ALLOW_NO_UNIX_SOCKETS (*archivebox.config.common.StorageConfig attribute*), 340
ALLOWDENYLIST_REGEX_FLAGS (*archivebox.config.constants.ConstantsDict attribute*), 329
ALLOWED_HOSTS (*archivebox.config.common.ServerConfig attribute*), 341
ALLOWED_HOSTS (*in module archivebox.core.settings*), 231
ALLOWED_IN_DATA_DIR (*archivebox.config.constants.ConstantsDict attribute*), 329
ancestors (*archivebox.machine.models.Process property*), 107
ANSI (*archivebox.config.common.ShellConfig attribute*), 339
ANSI (*in module archivebox.misc.logging*), 121
ansi_to_html() (*in module archivebox.misc.util*), 137
api (*in module archivebox.api.v1_api*), 173
API_AUTH_METHODS (*in module archivebox.api.auth*), 194
api_base_url (*in module archivebox.core.settings*), 231
api_docs_url (*archivebox.base_models.models.ModelWithUUID property*), 350
api_token() (*in module archivebox.core.templatetags.core_tags*), 221
api_url (*archivebox.base_models.models.ModelWithUUID property*), 350
api_url (*archivebox.core.models.ArchiveResult property*), 275
api_url (*archivebox.core.models.Snapshot property*), 269
api_url (*archivebox.core.models.Tag property*), 260
api_url (*archivebox.crawls.models.Crawl property*), 313
api_url (*archivebox.crawls.models.CrawlSchedule property*), 310
APIConfig (*class in archivebox.api.apps*), 168
ApiCorsMiddleware (*class in archivebox.api.middleware*), 193
APIToken (*class in archivebox.api.models*), 190
APIToken.Meta (*class in archivebox.api.models*), 191
apitoken_set() (*archivebox.core.admin_users.CustomUserAdmin method*), 286
APITokenAdmin (*class in archivebox.api.admin*), 203
app_label (*archivebox.api.models.APIToken.Meta attribute*), 191
app_label (*archivebox.api.models.OutboundWebhook.Meta attribute*), 191
app_label (*archivebox.core.models.ArchiveResult.Meta attribute*), 274

app_label (*archivebox.core.models.Snapshot.Meta attribute*), 264
app_label (*archivebox.core.models.SnapshotTag.Meta attribute*), 261
app_label (*archivebox.core.models.Tag.Meta attribute*), 260
app_label (*archivebox.crawls.models.Crawl.Meta attribute*), 312
app_label (*archivebox.crawls.models.CrawlSchedule.Meta attribute*), 310
app_label (*archivebox.machine.models.Binary.Meta attribute*), 101
app_label (*archivebox.machine.models.Machine.Meta attribute*), 97
app_label (*archivebox.machine.models.NetworkInterface.Meta attribute*), 99
app_label (*archivebox.machine.models.Process.Meta attribute*), 105
app_label (*archivebox.personas.models.Persona.Meta attribute*), 306
app_label (*archivebox.workers.models.ModelWithQueue.Meta attribute*), 214
APPEND_SLASH (*in module archivebox.core.settings*), 230
application (*in module archivebox.core.wsgi*), 285
application() (*in module archivebox.core.asgi*), 287
apply_crawl_config_filters() (*archivebox.crawls.models.Crawl method*), 313
apply_filters() (*in module archivebox.cli.cli_util*), 164
apply_import() (*archivebox.personas.forms.PersonaAdminForm method*), 304
apply_migrations() (*in module archivebox.misc.db*), 126
apply_snapshot_search() (*in module archivebox.search.query*), 292
apply_snapshot_tag_diff() (*archivebox.crawls.models.Crawl method*), 313
approximate_row_counts() (*in module archivebox.misc.db*), 125
archive() (*archivebox.core.models.Snapshot method*), 268
archive_commands (*archivebox.cli.ArchiveBoxGroup attribute*), 167
archive_dates() (*archivebox.core.models.Snapshot method*), 270
ARCHIVE_DIR (*archivebox.config.constants.ConstantsDict attribute*), 326
ARCHIVE_DIR (*in module archivebox.config.paths*), 334
ARCHIVE_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
archive_path (*archivebox.api.v1_core.SnapshotSchema attribute*), 183
archive_path() (*archivebox.core.models.Snapshot method*), 269
archive_path_for_values() (*archivebox.core.models.Snapshot class method*), 269
archive_path_from_db() (*archivebox.core.models.Snapshot method*), 269
archive_redirect_view() (*in module archivebox.api.urls*), 171
archive_size (*archivebox.api.v1_core.SnapshotSchema attribute*), 183
archive_size() (*archivebox.core.models.Snapshot method*), 269
archive_url (*archivebox.api.v1_core.TagSearchSnapshotSchema attribute*), 187
archivebox
    module, 92
archivebox.__main__
    module, 378
archivebox.api
    module, 167
archivebox.api.admin
    module, 203
archivebox.api.apps
    module, 167
archivebox.api.auth
    module, 193
archivebox.api.middleware
    module, 193
archivebox.api.models
    module, 190
archivebox.api.urls
    module, 171
archivebox.api.v1_api
    module, 172
archivebox.api.v1_auth
    module, 192
archivebox.api.v1_cli
    module, 198
archivebox.api.v1_core
    module, 175
archivebox.api.v1_crawls
    module, 168
archivebox.api.v1_machine
    module, 194
archivebox.api.v1_personas
    module, 173
archivebox.api.webhooks
    module, 171
archivebox.base_models

module, 349
archivebox.base_models.admin
module, 352
archivebox.base_models.apps
module, 349
archivebox.base_models.models
module, 349
archivebox.cli
module, 138
archivebox.cli.archivebox_add
module, 162
archivebox.cli.archivebox_archiveresult
module, 153
archivebox.cli.archivebox_binary
module, 144
archivebox.cli.archivebox_config
module, 152
archivebox.cli.archivebox_crawl
module, 139
archivebox.cli.archivebox_extract
module, 155
archivebox.cli.archivebox_help
module, 143
archivebox.cli.archivebox_init
module, 152
archivebox.cli.archivebox_install
module, 158
archivebox.cli.archivebox_list
module, 166
archivebox.cli.archivebox_machine
module, 147
archivebox.cli.archivebox_manage
module, 156
archivebox.cli.archivebox_mcp
module, 150
archivebox.cli.archivebox_oneshot
module, 152
archivebox.cli.archivebox_persona
module, 148
archivebox.cli.archivebox_pluginmap
module, 163
archivebox.cli.archivebox_process
module, 157
archivebox.cli.archivebox_remove
module, 138
archivebox.cli.archivebox_run
module, 141
archivebox.cli.archivebox_schedule
module, 141
archivebox.cli.archivebox_search
module, 141
archivebox.cli.archivebox_server
module, 161
archivebox.cli.archivebox_shell
module, 148
archivebox.cli.archivebox_snapshot
module, 159
archivebox.cli.archivebox_status
module, 158
archivebox.cli.archivebox_tag
module, 164
archivebox.cli.archivebox_update
module, 145
archivebox.cli.archivebox_version
module, 151
archivebox.cli.cli_util
module, 164
archivebox.cli.rescan
module, 156
archivebox.config
module, 321
archivebox.config.collection
module, 321
archivebox.config.common
module, 336
archivebox.config.configset
module, 331
archivebox.config.constants
module, 326
archivebox.config.django
module, 335
archivebox.config.ldap
module, 347
archivebox.config.paths
module, 333
archivebox.config.permissions
module, 323
archivebox.config.version
module, 348
archivebox.config.views
module, 329
archivebox.core
module, 215
archivebox.core.admin
module, 287
archivebox.core.admin_archiveresults
module, 223
archivebox.core.admin_site
module, 253
archivebox.core.admin_snapshots
module, 279
archivebox.core.admin_tags
module, 251
archivebox.core.admin_users
module, 285
archivebox.core.apps
module, 236
archivebox.core.asgi

module, 286
archivebox.core.context_processors
module, 242
archivebox.core.forms
module, 233
archivebox.core.management
module, 221
archivebox.core.management.commands
module, 221
archivebox.core.management.commands.archivebox
module, 221
archivebox.core.middleware
module, 276
archivebox.core.models
module, 259
archivebox.core.permissions
module, 238
archivebox.core.recovery_util
module, 242
archivebox.core.routes_util
module, 254
archivebox.core.settings
module, 227
archivebox.core.settings_logging
module, 236
archivebox.core.setup_wizard
module, 253
archivebox.core.shutdown_util
module, 249
archivebox.core.snapshot_status
module, 236
archivebox.core.sqlite_backend
module, 221
archivebox.core.sqlite_backend.base
module, 221
archivebox.core.tag_util
module, 277
archivebox.core.takeover_util
module, 240
archivebox.core.templatetags
module, 215
archivebox.core.templatetags.core_tags
module, 215
archivebox.core.urls
module, 248
archivebox.core.views
module, 242
archivebox.core.widgets
module, 226
archivebox.core.wsgi
module, 285
archivebox.crawls
module, 308
archivebox.crawls.admin
module, 316
archivebox.crawls.apps
module, 308
archivebox.crawls.locks
module, 315
archivebox.crawls.models
module, 309
archivebox.crawls.schedule_util
module, 309
archivebox.ldap
module, 117
archivebox.ldap.apps
module, 118
archivebox.ldap.auth
module, 118
archivebox.machine
module, 92
archivebox.machine.admin
module, 112
archivebox.machine.apps
module, 92
archivebox.machine.detect
module, 93
archivebox.machine.env_util
module, 111
archivebox.machine.models
module, 94
archivebox.manage
module, 378
archivebox.mcp
module, 294
archivebox.mcp.apps
module, 294
archivebox.mcp.server
module, 295
archivebox.misc
module, 119
archivebox.misc.checks
module, 123
archivebox.misc.db
module, 124
archivebox.misc.jsonl
module, 132
archivebox.misc.logging
module, 120
archivebox.misc.logging_util
module, 121
archivebox.misc.monkey_patches
module, 134
archivebox.misc.paginators
module, 119
archivebox.misc.serve_static
module, 127
archivebox.misc.shell_welcome_message

module, 120
archivebox.misc.system
module, 119
archivebox.misc.toml_util
module, 119
archivebox.misc.util
module, 134
archivebox.personas
module, 298
archivebox.personas.admin
module, 307
archivebox.personas.apps
module, 304
archivebox.personas.forms
module, 303
archivebox.personas.importers
module, 298
archivebox.personas.models
module, 304
archivebox.plugins
module, 367
archivebox.plugins.apps
module, 370
archivebox.plugins.discovery
module, 371
archivebox.plugins.forms
module, 367
archivebox.plugins.hooks
module, 376
archivebox.plugins.output_groups
module, 370
archivebox.plugins.views
module, 375
archivebox.progressmonitor
module, 116
archivebox.progressmonitor.apps
module, 116
archivebox.progressmonitor.collection
module, 116
archivebox.progressmonitor.views
module, 117
archivebox.search
module, 288
archivebox.search.admin
module, 293
archivebox.search.apps
module, 288
archivebox.search.backends
module, 288
archivebox.search.config
module, 292
archivebox.search.query
module, 291
archivebox.search.sonic_daemon
module, 288
archivebox.search.views
module, 289
archivebox.services
module, 355
archivebox.services.archive_result_service
module, 358
archivebox.services.binary_service
module, 356
archivebox.services.crawl_service
module, 367
archivebox.services.machine_service
module, 366
archivebox.services.process_service
module, 355
archivebox.services.resource_admission
module, 359
archivebox.services.runner
module, 361
archivebox.services.snapshot_service
module, 364
archivebox.services.tag_service
module, 365
archivebox.uuid_compat
module, 377
archivebox.workers
module, 204
archivebox.workers.admin
module, 214
archivebox.workers.apps
module, 205
archivebox.workers.management
module, 204
archivebox.workers.management.commands
module, 204
archivebox.workers.management.commands.runner_watch
module, 204
archivebox.workers.management.commands.supervisord_watchdo
module, 205
archivebox.workers.models
module, 212
archivebox.workers.supervisord_parent_watchdog
module, 212
archivebox.workers.supervisord_util
module, 206
ARCHIVEBOX_ACCOUNT (*in module archivebox.config.permissions*), 325
archivebox_admin (*in module archivebox.core.admin_site*), 254
archivebox_cmd() (*in module archivebox.workers.supervisord_util*), 208
archivebox_globals() (*in module archivebox.core.context_processors*), 242

ARCHIVEBOX_GROUP (*archivebox.config.constants.ConstantsDict attribute*), 327
ARCHIVEBOX_USER (*archivebox.config.constants.ConstantsDict attribute*), 326
ARCHIVEBOX_USER_EXISTS (*in module archivebox.config.permissions*), 325
ArchiveBoxAdmin (*class in archivebox.core.admin_site*), 254
ArchiveBoxASGIHandler (*class in archivebox.core.asgi*), 287
ArchiveBoxBaseConfig (*class in archivebox.config.common*), 344
ArchiveBoxBinaryService (*class in archivebox.services.binary_service*), 356
ArchiveBoxConfig (*in module archivebox.config.common*), 346
ArchiveBoxGroup (*class in archivebox.cli*), 166
ArchiveBoxLDAPBackend (*class in archivebox.ldap.auth*), 118
ArchiveBoxLoginView (*class in archivebox.core.admin_site*), 253
ArchiveBoxSourceSettings (*class in archivebox.config.common*), 346
archiveresult (*archivebox.machine.models.Process attribute*), 105
ArchiveResult (*class in archivebox.core.models*), 271
ArchiveResult.Meta (*class in archivebox.core.models*), 274
ArchiveResult.StatusChoices (*class in archivebox.core.models*), 271
archiveresult_link() (*archivebox.machine.admin.ProcessAdmin method*), 115
archiveresult_output_display() (*archivebox.machine.admin.ProcessAdmin method*), 115
archiveresult_set (*archivebox.core.models.Snapshot attribute*), 264
archiveresult_set() (*archivebox.core.admin_users.CustomUserAdmin method*), 286
ARCHIVERESULT_UPLOAD_HOOK_NAME (*in module archivebox.api.v1_core*), 177
ARCHIVERESULT_UPLOAD_PLUGIN_RE (*in module archivebox.api.v1_core*), 177
ArchiveResultAdmin (*class in archivebox.core.admin_archiveresults*), 224
ArchiveResultAdmin.Meta (*class in archivebox.core.admin_archiveresults*), 225
ArchiveResultFilterSchema (*class in archivebox.api.v1_core*), 180
ArchiveResultInline (*class in archivebox.core.admin_archiveresults*), 223
archiveresults (*archivebox.api.v1_core.SnapshotSchema attribute*), 183
archiveresults_list() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
ArchiveResultSchema (*class in archivebox.api.v1_core*), 179
ArchiveResultService (*class in archivebox.services.archive_result_service*), 359
ArchivingConfig (*class in archivebox.config.common*), 342
arguments_to_cli() (*in module archivebox.mcp.server*), 297
as_choice_label() (*archivebox.personas.importers.PersonaImportSource method*), 300
as_csv (*archivebox.api.v1_cli.ListCommandSchema attribute*), 202
as_dict() (*archivebox.config.configset.BaseConfigSet method*), 333
as_html (*archivebox.api.v1_cli.ListCommandSchema attribute*), 202
as_json (*archivebox.api.v1_cli.ListCommandSchema attribute*), 202
ASCII_ICON (*in module archivebox*), 379
ASCII_LOGO (*in module archivebox*), 379
ASCII_LOGO_MINI (*in module archivebox.__main__*), 378
ASGI_APPLICATION (*in module archivebox.core.settings*), 229
assert_dir_can_contain_unix_sockets() (*in module archivebox.config.paths*), 335
AsyncioCancelledShieldFilter (*class in archivebox.core.settings_logging*), 237
atomic_write() (*in module archivebox.misc.system*), 120
AttrDict (*class in archivebox.misc.logging*), 121
auth_file_written (*archivebox.api.v1_personas.PersonaSyncResponseSchema attribute*), 175
auth_json (*archivebox.api.v1_personas.PersonaSyncSchema attribute*), 174
AUTH_PASSWORD_VALIDATORS (*in module archivebox.core.settings*), 232
auth_state() (*archivebox.personas.admin.PersonaAdmin method*), 308
AUTH_STORAGE_FILE (*archivebox.config.common.ArchivingConfig attribute*), 343
AUTH_STORAGE_FILE (*archivebox.personas.models.Persona property*),

306
auth_using_password() (*in module archivebox.api.auth*), 194
auth_using_token() (*in module archivebox.api.auth*), 194
authenticate() (*archivebox.api.auth.BearerTokenAuth method*), 194
authenticate() (*archivebox.api.auth.HeaderTokenAuth method*), 194
authenticate() (*archivebox.api.auth.QueryParamTokenAuth method*), 194
authenticate_ldap_user() (*archivebox.ldap.auth.ArchiveBoxLDAPBackend method*), 118
authenticated_user_from_request() (*in module archivebox.api.auth*), 194
AUTHENTICATION_BACKENDS (*in module archivebox.core.settings*), 230
autocomplete_fields (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
autocomplete_fields (*archivebox.core.admin_tags.TagInline attribute*), 251
autocomplete_fields (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
AutoDateTimeField (*class in archivebox.base_models.models*), 350

B

BACKOFF (*archivebox.core.models.ArchiveResult.StatusChoices attribute*), 272
BASE_URL (*archivebox.config.common.ServerConfig attribute*), 340
base_url (*in module archivebox.misc.util*), 136
base_url() (*archivebox.core.models.Snapshot method*), 270
base_url_from_legacy_server_config() (*in module archivebox.config.common*), 338
BASE_URL_IS_HTTPS (*in module archivebox.core.settings*), 231
BaseConfigSet (*class in archivebox.config.configset*), 332
BaseModelAdmin (*class in archivebox.base_models.admin*), 354
basename (*in module archivebox.misc.util*), 136
basename() (*archivebox.core.models.Snapshot method*), 270
batch_size (*archivebox.api.v1_cli.UpdateCommandSchema attribute*), 201
BearerTokenAuth (*class in archivebox.api.auth*), 194
before (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 200
better_toml_dump_str() (*in module archivebox.misc.toml_util*), 119
bin_abspath (*archivebox.machine.models.Process property*), 106
bin_providers (*archivebox.api.v1_machine.MachineFilterSchema attribute*), 196
binaries_list_view() (*in module archivebox.config.views*), 331
binary (*archivebox.core.models.ArchiveResult property*), 276
binary (*archivebox.machine.models.Process attribute*), 105
BINARY (*archivebox.machine.models.Process.TypeChoices attribute*), 104
binary (*archivebox.plugins.discovery.PluginSpecialConfig attribute*), 373
Binary (*class in archivebox.machine.models*), 99
Binary.Meta (*class in archivebox.machine.models*), 101
Binary.StatusChoices (*class in archivebox.machine.models*), 99
binary_detail_view() (*in module archivebox.config.views*), 331
binary_id (*archivebox.machine.models.Process attribute*), 105
binary_info() (*archivebox.machine.admin.ProcessAdmin method*), 115
binary_info() (*archivebox.machine.models.Binary method*), 101
binary_lifecycle_lock() (*in module archivebox.crawls.locks*), 316
binary_link() (*archivebox.machine.admin.ProcessAdmin method*), 115
BINARY_RECHECK_INTERVAL (*in module archivebox.machine.models*), 95
binary_set (*archivebox.core.models.Snapshot property*), 265
BinaryAdmin (*class in archivebox.machine.admin*), 114
BinaryFilterSchema (*class in archivebox.api.v1_machine*), 197
BinaryManager (*class in archivebox.machine.models*), 99
BinarySchema (*class in archivebox.api.v1_machine*), 196
BIND_ADDR (*archivebox.config.common.ServerConfig attribute*), 340
binprovider (*archivebox.api.v1_machine.BinaryFilterSchema attribute*), 198
binprovider (*archive-*

box.api.v1_machine.BinarySchema attribute), 197
binprovider (*archivebox.machine.models.Binary attribute*), 100
binproviders (*archivebox.api.v1_machine.BinarySchema attribute*), 197
binproviders (*archivebox.machine.models.Binary attribute*), 100
block_size (*archivebox.misc.serve_static.RangedFileReader attribute*), 132
BLOCK_UNSAFE_METHODS (*archivebox.config.common.ServerConfig property*), 341
bookmarked_at (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
bookmarked_at (*archivebox.core.models.Snapshot attribute*), 262
bookmarked_at__gte (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
bookmarked_at__lt (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
bookmarked_date() (*archivebox.core.models.Snapshot method*), 270
browser (*archivebox.personas.importers.PersonaImportSource attribute*), 300
browser_binary (*archivebox.personas.importers.PersonaImportSource attribute*), 300
BROWSER_EXTENSION_UPLOAD_HOOK_NAME (*archivebox.core.models.Snapshot attribute*), 262
browser_label (*archivebox.personas.importers.PersonaImportSource property*), 300
BROWSER_LABELS (*in module archivebox.personas.importers*), 299
BROWSER_PROFILE_DIR_NAMES (*in module archivebox.personas.importers*), 299
BROWSER_PROFILE_FINDERS (*in module archivebox.personas.importers*), 301
browser_session() (*in module archivebox.api.v1_auth*), 192
browser_settings_to_config() (*in module archivebox.api.v1_personas*), 175
build_abx_dl_display_command() (*in module archivebox.core.admin_archiveresults*), 223
build_abx_dl_replay_command() (*in module archivebox.core.admin_archiveresults*), 223
build_admin_url() (*in module archivebox.core.routes_util*), 258
build_archiveresult_request() (*in module archivebox.cli.archivebox_archiveresult*), 154
build_crawl_config_snapshot() (*in module archivebox.config.common*), 339
build_original_url() (*in module archivebox.core.routes_util*), 259
build_plugin_groups() (*archivebox.plugins.forms.PluginConfigFormMixin method*), 369
build_server_worker_plan() (*in module archivebox.workers.supervisord_util*), 211
build_snapshot_detail_path() (*in module archivebox.core.routes_util*), 258
build_snapshot_detail_url() (*in module archivebox.core.routes_util*), 258
build_snapshot_files_url() (*in module archivebox.core.routes_util*), 259
build_snapshot_index_path() (*in module archivebox.core.routes_util*), 258
build_snapshot_queryset() (*in module archivebox.cli.archivebox_snapshot*), 160
build_snapshot_role_output_url() (*in module archivebox.core.routes_util*), 258
build_snapshot_url() (*in module archivebox.core.routes_util*), 258
build_snapshot_zip_url() (*in module archivebox.core.routes_util*), 259
build_tag_card() (*in module archivebox.core.tag_util*), 279
build_tag_cards() (*in module archivebox.core.tag_util*), 279
BUILD_TIME (*archivebox.config.common.ShellConfig property*), 340
build_web_url() (*in module archivebox.core.routes_util*), 258
BUILTIN_PLUGINS_DIR (*in module archivebox.plugins.discovery*), 373
bulk_create() (*archivebox.core.models.SnapshotQuerySet method*), 261
bump_retry_at() (*archivebox.workers.models.ModelWithQueue method*), 214

C

CACHE_DIR (*archivebox.config.constants.ConstantsDict attribute*), 328
CACHE_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
CacheControlMiddleware() (*in module archivebox.core.middleware*), 277
CACHES (*in module archivebox.core.settings*), 231
can_install (*archivebox.machine.models.Binary property*), 101

can_start() (*archivebox.crawls.models.Crawl method*), 315
can_view_snapshot() (*in module archivebox.core.permissions*), 239
cancel() (*archivebox.core.models.Snapshot method*), 265
cancel() (*archivebox.crawls.models.Crawl method*), 313
canonical_base_host_for_request() (*in module archivebox.core.routes_util*), 257
CARD_OVERFLOW_STYLE (*in module archivebox.core.templatetags.core_tags*), 217
CaseConfigParser (*class in archivebox.config.configset*), 331
cdp_url (*archivebox.personas.importers.PersonaImportSource attribute*), 300
cgroup_paths() (*archivebox.services.resource_admission.ResourceAdmission static method*), 360
change_actions (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
change_actions (*archivebox.machine.admin.ProcessAdmin attribute*), 115
change_fieldsets (*archivebox.personas.admin.PersonaAdmin attribute*), 308
change_form_template (*archivebox.base_models.admin.BaseModelAdmin attribute*), 354
change_form_template (*archivebox.core.admin_tags.TagAdmin attribute*), 252
change_form_template (*archivebox.core.admin_users.CustomUserAdmin attribute*), 286
change_form_template (*archivebox.crawls.admin.CrawlAdmin attribute*), 317
change_form_template (*archivebox.personas.admin.PersonaAdmin attribute*), 307
change_list_template (*archivebox.core.admin_tags.TagAdmin attribute*), 252
change_view() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
change_view() (*archivebox.crawls.admin.CrawlAdmin method*), 319
change_view() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
changelist_view() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
changelist_view() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
changelist_view() (*archivebox.core.admin_tags.TagAdmin method*), 252
changelist_view() (*archivebox.crawls.admin.CrawlAdmin method*), 318
CHANNEL_LAYERS (*in module archivebox.core.settings*), 231
check_api_token() (*in module archivebox.api.v1_auth*), 192
check_data_dir_permissions() (*in module archivebox.misc.checks*), 124
check_data_folder() (*in module archivebox.misc.checks*), 123
check_io_encoding() (*in module archivebox.misc.checks*), 124
check_lib_dir() (*in module archivebox.misc.checks*), 124
check_migrations() (*in module archivebox.misc.checks*), 124
check_not_inside_source_dir() (*in module archivebox.misc.checks*), 124
check_not_root() (*in module archivebox.misc.checks*), 124
CHECK_SSL_VALIDITY (*archivebox.config.common.ArchivingConfig attribute*), 343
check_tmp_dir() (*in module archivebox.misc.checks*), 124
children (*archivebox.machine.models.Process attribute*), 105
choice_value (*archivebox.personas.importers.PersonaImportSource property*), 300
CHROME_DOWNLOADS_DIR (*archivebox.personas.models.Persona property*), 306
chrome_profile_state() (*archivebox.personas.admin.PersonaAdmin method*), 308
CHROME_USER_DATA_DIR (*archivebox.personas.models.Persona property*), 306
CHROMIUM_BROWSERS (*in module archivebox.personas.importers*), 301
city (*archivebox.machine.models.NetworkInterface attribute*), 98
claim_for_worker() (*archive-*

box.workers.models.ModelWithQueue class method), 214
claim_processing_lock() (*archivebox.workers.models.ModelWithQueue method*), 214
clean() (*archivebox.core.forms.AddLinkForm method*), 235
clean() (*archivebox.core.forms.TagField method*), 235
clean() (*archivebox.personas.forms.PersonaAdminForm method*), 303
clean_crawl_max_concurrent_snapshots() (*archivebox.core.forms.AddLinkForm method*), 235
clean_crawl_max_size() (*archivebox.core.forms.AddLinkForm method*), 235
clean_crawl_timeout() (*archivebox.core.forms.AddLinkForm method*), 235
clean_delete_after() (*archivebox.core.forms.AddLinkForm method*), 235
clean_max_urls() (*archivebox.core.forms.AddLinkForm method*), 235
clean_name() (*archivebox.core.admin_tags.TagAdminForm method*), 252
clean_name() (*archivebox.personas.forms.PersonaAdminForm method*), 303
clean_persona() (*archivebox.core.forms.AddLinkForm method*), 235
clean_plugin_config_overrides() (*archivebox.plugins.forms.PluginConfigFormMixin method*), 369
clean_schedule() (*archivebox.core.forms.AddLinkForm method*), 235
clean_snapshot_max_size() (*archivebox.core.forms.AddLinkForm method*), 235
clean_tags() (*archivebox.core.admin_snapshots.SnapshotActionForm method*), 280
clean_tags_editor() (*archivebox.crawls.admin.CrawlAdminForm method*), 317
clean_timeout() (*archivebox.core.forms.AddLinkForm method*), 235
clean_url() (*archivebox.core.forms.AddLinkForm method*), 235
clean_url_filters() (*archivebox.core.forms.AddLinkForm method*), 235
clean_url_filters() (*archivebox.crawls.admin.CrawlAdminForm method*), 317
cleanup() (*archivebox.machine.models.Binary method*), 102
cleanup_chrome() (*archivebox.personas.models.Persona method*), 306
cleanup_chrome_all() (*archivebox.personas.models.Persona class method*), 307
cleanup_chrome_profile() (*archivebox.personas.models.Persona method*), 306
cleanup_orphaned_chrome() (*archivebox.machine.models.Process class method*), 111
cleanup_orphaned_workers() (*archivebox.machine.models.Process class method*), 111
cleanup_runtime() (*archivebox.crawls.models.Crawl method*), 315
cleanup_runtime_for_crawl() (*archivebox.personas.models.Persona method*), 307
cleanup_stale_running() (*archivebox.machine.models.Process class method*), 107
clear (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 202
clear() (*archivebox.workers.supervisord_util.SupervisordConnectionCach method*), 210
CLI (*archivebox.machine.models.Process.TypeChoices attribute*), 103
cli() (*in module archivebox.cli*), 167
cli_add() (*in module archivebox.api.v1_cli*), 202
cli_remove() (*in module archivebox.api.v1_cli*), 203
cli_schedule() (*in module archivebox.api.v1_cli*), 202
cli_search() (*in module archivebox.api.v1_cli*), 203
cli_update() (*in module archivebox.api.v1_cli*), 202
click_command_to_mcp_tool() (*in module archivebox.mcp.server*), 297
click_group_to_mcp_tool() (*in module archivebox.mcp.server*), 297
click_type_to_json_schema_type() (*in module archivebox.mcp.server*), 297
CLICommandResponseSchema (*class in archivebox.api.v1_cli*), 199
close() (*archivebox.misc.serve_static._StreamingQueueWriter method*), 129
close() (*archivebox.misc.serve_static.RangedFileReader method*), 132
cmd (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
cmd (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
cmd (*archivebox.core.models.ArchiveResult property*), 276
cmd (*archivebox.machine.models.Process attribute*), 104
cmd_display() (*archivebox.machine.admin.ProcessAdmin method*), 115
cmd_str() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin*

method), 226
cmd_str() (*archivebox.machine.admin.ProcessAdmin method*), 115
cmd_version (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
cmd_version (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
cmd_version (*archivebox.core.models.ArchiveResult property*), 276
cmd_version (*archivebox.machine.models.Process property*), 106
collect_urls_from_plugins() (*in module archivebox.plugins.hooks*), 377
COLLECTION_ID (*archivebox.config.constants.ConstantsDict attribute*), 326
collection_summary() (*in module archivebox.progressmonitor.collection*), 117
COLOR_DICT (*in module archivebox.misc.logging*), 121
COLOR_REGEX (*in module archivebox.misc.util*), 137
color_scheme (*archivebox.api.v1_personas.PersonaBrowserSettingsSchema attribute*), 174
command (*archivebox.mcp.server.MCPTool attribute*), 297
Command (*class in archivebox.core.management.commands.archivebox*), 221
Command (*class in archivebox.workers.management.commands.runner_watch*), 204
Command (*class in archivebox.workers.management.commands.supervisord_config*), 205
command() (*archivebox.core.admin_archiveresults.ArchiveResultInline method*), 224
command_accepts_stdin() (*in module archivebox.mcp.server*), 297
command_owns_foreground_runner() (*in module archivebox.core.takeover_util*), 241
command_owns_runtime_stack() (*in module archivebox.core.takeover_util*), 241
command_path (*archivebox.mcp.server.MCPTool attribute*), 297
COMMIT_HASH (*archivebox.config.common.ShellConfig property*), 339
COMMIT_HASH (*in module archivebox.api.v1_api*), 172
compact_command() (*in module archivebox.misc.db*), 126
compact_uuid() (*in module archivebox.uuid_compat*), 378
CompactUUID (*class in archivebox.uuid_compat*), 377
CompactUUIDField (*class in archivebox.uuid_compat*), 378
completed() (*archivebox.core.admin_archiveresults.ArchiveResultInline method*), 224
computed_config_keys (*archivebox.config.common.ArchiveBoxBaseConfig attribute*), 345
computed_config_keys (*archivebox.config.configset.BaseConfigSet attribute*), 332
COMPUTED_CONFIG_KEYS (*in module archivebox.config.configset*), 331
config (*archivebox.api.v1_crawls.CrawlCreateSchema attribute*), 170
config (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
config (*archivebox.api.v1_personas.PersonaSchema attribute*), 174
config (*archivebox.base_models.models.ModelWithConfig attribute*), 351
config (*archivebox.core.forms.AddLinkForm attribute*), 235
config (*archivebox.core.models.Snapshot attribute*), 263
config (*archivebox.crawls.models.Crawl attribute*), 311
config (*archivebox.crawls.models.CrawlSchedule attribute*), 310
config (*archivebox.machine.models.Machine attribute*), 97
CONFIG (*in module archivebox.config.django*), 336
config() (*in module archivebox.cli.archivebox_config*), 152
config_field_metadata() (*in module archivebox.config.common*), 346
config_field_type() (*in module archivebox.config.common*), 346
CONFIG_FILE (*archivebox.config.constants.ConstantsDict attribute*), 328
CONFIG_FILE_HEADER (*in module archivebox.config.collection*), 322
CONFIG_FILE_NAME (*in module archivebox.workers.supervisord_util*), 207
CONFIG_FILENAME (*archivebox.config.constants.ConstantsDict attribute*), 328
ConfigEditorMixin (*class in archivebox.base_models.admin*), 354
ConfigLookup (*class in archivebox.plugins.discovery*), 373
ConfigOption (*class in archivebox.base_models.admin*), 353
ConfigOverrides (*in module archivebox.config.common*), 337

ConfigPayload (*in module archivebox.config.common*), 337
configured_stopwaitsecs() (*in module archivebox.core.shutdown_util*), 250
CONSOLE (*in module archivebox.misc.logging*), 121
CONSTANTS (*in module archivebox.config.constants*), 329
CONSTANTS_CONFIG (*in module archivebox.config.constants*), 329
ConstantsDict (*class in archivebox.config.constants*), 326
constraints (*archivebox.core.models.ArchiveResult.Meta attribute*), 274
constraints (*archivebox.core.models.Snapshot.Meta attribute*), 264
constraints (*archivebox.machine.models.NetworkInterface.Meta attribute*), 99
constraints (*archivebox.machine.models.Process.Meta attribute*), 105
continuous (*archivebox.api.v1_cli.UpdateCommandSchema attribute*), 201
CONTROL_PLANE_ENABLED (*archivebox.config.common.ServerConfig property*), 341
convert_index_json_to_jsonl() (*archivebox.core.models.Snapshot method*), 268
cookie_count (*archivebox.personas.importers.PersonaImportResult attribute*), 300
COOKIES_FILE (*archivebox.config.common.ArchivingConfig attribute*), 343
COOKIES_FILE (*archivebox.personas.models.Persona property*), 306
cookies_file_written (*archivebox.api.v1_personas.PersonaSyncResponseSchema attribute*), 175
cookies_imported (*archivebox.personas.importers.PersonaImportResult attribute*), 300
cookies_state() (*archivebox.personas.admin.PersonaAdmin method*), 308
cookies_txt (*archivebox.api.v1_personas.PersonaSyncSchema attribute*), 174
copy_browser_user_data_dir() (*in module archivebox.personas.importers*), 302
copy_chrome_profile() (*archivebox.personas.models.Persona method*), 307
CoreConfig (*class in archivebox.core.apps*), 236
count (*archivebox.api.v1_core.CustomPagination.Output attribute*), 177
count() (*archivebox.misc.paginators.AcceleratedPaginator method*), 119
count_urls_for_limit() (*archivebox.crawls.models.Crawl method*), 314
country (*archivebox.machine.models.NetworkInterface attribute*), 98
crawl (*archivebox.core.models.Snapshot attribute*), 262
Crawl (*class in archivebox.crawls.models*), 311
Crawl.Meta (*class in archivebox.crawls.models*), 312
crawl_config_values_search_wave() (*in module archivebox.search.query*), 291
crawl_count_expr() (*archivebox.core.models.Snapshot class method*), 264
crawl_count_subquery() (*archivebox.core.models.Snapshot class method*), 264
crawl_file() (*in module archivebox.api.v1_crawls*), 170
crawl_file_nested_1() (*in module archivebox.api.v1_crawls*), 171
crawl_file_nested_2() (*in module archivebox.api.v1_crawls*), 171
crawl_file_root() (*in module archivebox.api.v1_crawls*), 171
crawl_id (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 201
crawl_id (*archivebox.api.v1_core.SnapshotCreateSchema attribute*), 183
crawl_id (*archivebox.api.v1_core.SnapshotDeleteResponseSchema attribute*), 184
crawl_id (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
crawl_id (*archivebox.api.v1_crawls.CrawlDeleteResponseSchema attribute*), 170
crawl_id (*archivebox.core.models.Snapshot attribute*), 263
crawl_is_cancelled() (*archivebox.services.runner.CrawlRunner method*), 362
crawl_is_paused() (*archivebox.services.runner.CrawlRunner method*), 362
crawl_lifecycle_lock() (*in module archivebox.crawls.locks*), 316
crawl_lifecycle_lock_path() (*in module archivebox.crawls.locks*), 316
crawl_link() (*archivebox.machine.admin.ProcessAdmin method*), 115
CRAWL_MAX_CONCURRENT_SNAPSHOTS (*archivebox.config.common.ArchivingConfig attribute*), 343
crawl_max_concurrent_snapshots (*archive-*

box.core.forms.AddLinkForm attribute), 234
crawl_max_size (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
CRAWL_MAX_SIZE (*archivebox.config.common.ArchivingConfig attribute*), 343
crawl_max_size (*archivebox.core.forms.AddLinkForm attribute*), 234
CRAWL_MAX_URLS (*archivebox.config.common.ArchivingConfig attribute*), 343
crawl_set (*archivebox.crawls.models.CrawlSchedule attribute*), 310
crawl_summary() (*in module archivebox.progressmonitor.collection*), 117
crawl_timeout (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
CRAWL_TIMEOUT (*archivebox.config.common.ArchivingConfig attribute*), 343
crawl_timeout (*archivebox.core.forms.AddLinkForm attribute*), 234
crawl_total_and_status_counts() (*archivebox.core.models.Snapshot class method*), 264
CrawlAdmin (*class in archivebox.crawls.admin*), 317
CrawlAdmin.Media (*class in archivebox.crawls.admin*), 318
CrawlAdminForm (*class in archivebox.crawls.admin*), 317
CrawlAdminForm.Meta (*class in archivebox.crawls.admin*), 317
CrawlCreateSchema (*class in archivebox.api.v1_crawls*), 170
CrawlDeleteResponseSchema (*class in archivebox.api.v1_crawls*), 170
CrawlRunner (*class in archivebox.services.runner*), 362
crawls() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
CRAWLS_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
CrawlSchedule (*class in archivebox.crawls.models*), 309
CrawlSchedule.Meta (*class in archivebox.crawls.models*), 310
CrawlScheduleAdmin (*class in archivebox.crawls.admin*), 320
CrawlSchema (*class in archivebox.api.v1_crawls*), 168
CrawlsConfig (*class in archivebox.crawls.apps*), 308
CrawlService (*class in archivebox.services.crawl_service*), 367
CrawlUpdateSchema (*class in archivebox.api.v1_crawls*), 169
create_and_chown_dir() (*in module archivebox.config.paths*), 335
create_archiveresult() (*in module archivebox.api.v1_core*), 181
create_archiveresults() (*in module archivebox.cli.archivebox_archiveresult*), 154
create_binary() (*in module archivebox.cli.archivebox_binary*), 144
create_cmd() (*in module archivebox.cli.archivebox_archiveresult*), 154
create_cmd() (*in module archivebox.cli.archivebox_binary*), 145
create_cmd() (*in module archivebox.cli.archivebox_crawl*), 140
create_cmd() (*in module archivebox.cli.archivebox_persona*), 150
create_cmd() (*in module archivebox.cli.archivebox_snapshot*), 161
create_cmd() (*in module archivebox.cli.archivebox_tag*), 165
create_crawl() (*in module archivebox.api.v1_crawls*), 170
create_crawl() (*in module archivebox.cli.archivebox_crawl*), 140
create_cursor() (*archivebox.core.sqlite_backend.base.DatabaseWrapper method*), 222
create_discovered_snapshot() (*archivebox.crawls.models.Crawl method*), 314
create_discovered_snapshots() (*archivebox.crawls.models.Crawl method*), 314
create_for_archiveresult() (*archivebox.machine.models.ProcessManager method*), 102
create_from_directory() (*archivebox.core.models.Snapshot class method*), 267
create_initial_snapshots() (*archivebox.services.runner.CrawlRunner method*), 363
create_personas() (*in module archivebox.cli.archivebox_persona*), 149
create_request() (*archivebox.core.asgi.ArchiveBoxASGIHandler method*), 287
create_scheduler_row() (*archivebox.crawls.models.Crawl class method*), 314
create_snapshot() (*in module archivebox.api.v1_core*), 186
create_snapshots() (*in module archivebox.cli.archivebox_snapshot*), 160

create_snapshots_from_urls() (*archivebox.crawls.models.Crawl method*), 314
create_supervisord_config() (*in module archivebox.workers.supervisord_util*), 209
create_tags() (*in module archivebox.cli.archivebox_tag*), 165
create_temporal_response() (*archivebox.api.v1_api.NinjaAPIWithIOCapture method*), 173
create_worker_config() (*in module archivebox.workers.supervisord_util*), 210
created (*archivebox.api.v1_core.TagCreateResponseSchema attribute*), 187
created (*archivebox.api.v1_personas.PersonaSyncResponseSchema attribute*), 175
created_at (*archivebox.api.models.APIToken attribute*), 191
created_at (*archivebox.api.models.OutboundWebhook attribute*), 191
created_at (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
created_at (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
created_at (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
created_at (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
created_at (*archivebox.api.v1_core.TagSchema attribute*), 186
created_at (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
created_at (*archivebox.api.v1_machine.BinarySchema attribute*), 197
created_at (*archivebox.api.v1_machine.MachineSchema attribute*), 195
created_at (*archivebox.api.v1_personas.PersonaSchema attribute*), 174
created_at (*archivebox.base_models.models.ModelWithUUID attribute*), 350
created_at (*archivebox.core.models.ArchiveResult attribute*), 273
created_at (*archivebox.core.models.Snapshot attribute*), 262
created_at (*archivebox.core.models.Tag attribute*), 260
created_at (*archivebox.crawls.models.Crawl attribute*), 311
created_at (*archivebox.crawls.models.CrawlSchedule attribute*), 309
created_at (*archivebox.machine.models.Binary attribute*), 100
created_at (*archivebox.machine.models.Machine attribute*), 96
created_at (*archivebox.machine.models.NetworkInterface attribute*), 98
created_at (*archivebox.machine.models.Process attribute*), 104
created_at (*archivebox.personas.models.Persona attribute*), 305
created_at__gte (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
created_at__gte (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 184
created_at__lt (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
created_at__lt (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
created_at_display() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
created_by (*archivebox.api.models.APIToken attribute*), 191
created_by (*archivebox.api.models.OutboundWebhook attribute*), 191
created_by (*archivebox.api.v1_core.TagSearchResponseSchema attribute*), 188
created_by (*archivebox.base_models.models.ModelWithUUID attribute*), 350
created_by (*archivebox.core.models.ArchiveResult property*), 274
created_by (*archivebox.core.models.Snapshot property*), 265
created_by (*archivebox.core.models.Tag attribute*), 260
created_by (*archivebox.crawls.models.Crawl attribute*), 311
created_by (*archivebox.crawls.models.CrawlSchedule attribute*), 310
created_by (*archivebox.personas.models.Persona attribute*), 305
created_by_id (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
created_by_id (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 184
created_by_id (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
created_by_id (*archivebox.api.v1_core.TagSchema attribute*), 186
created_by_id (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
created_by_id (*archivebox.api.v1_personas.PersonaSchema attribute*),

174
created_by_username (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
created_by_username (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 184
created_by_username (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
created_by_username (*archivebox.api.v1_core.TagSchema attribute*), 186
created_by_username (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
created_by_username (*archivebox.api.v1_personas.PersonaSchema attribute*), 174
CSRF_COOKIE_DOMAIN (*in module archivebox.core.settings*), 232
CSRF_COOKIE_NAME (*in module archivebox.core.settings*), 231
CSRF_COOKIE_SECURE (*in module archivebox.core.settings*), 231
CSRF_TRUSTED_ORIGINS (*archivebox.config.common.ServerConfig attribute*), 341
CSRF_TRUSTED_ORIGINS (*in module archivebox.core.settings*), 231
css (*archivebox.base_models.admin.KeyValueWidget.Media attribute*), 353
css (*archivebox.core.widgets.TagEditorWidget.Media attribute*), 226
css (*archivebox.crawls.admin.CrawlAdmin.Media attribute*), 318
current() (*archivebox.machine.models.Machine class method*), 97
current() (*archivebox.machine.models.MachineManager method*), 96
current() (*archivebox.machine.models.NetworkInterface class method*), 99
current() (*archivebox.machine.models.NetworkInterfaceManager method*), 98
current() (*archivebox.machine.models.Process class method*), 106
current() (*archivebox.machine.models.ProcessManager method*), 102
current_command() (*in module archivebox.core.takeover_util*), 241
current_iface() (*archivebox.services.process_service.ProcessService method*), 356
current_network_interface_with_machine() (*in module archivebox.services.process_service*), 355
current_readonly() (*archivebox.machine.models.Machine class method*), 97
current_readonly() (*archivebox.machine.models.Process class method*), 106
current_tag_names() (*archivebox.crawls.models.Crawl method*), 313
CUSTOM_PLUGINS_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 328
CUSTOM_TEMPLATES_DIR (*archivebox.config.constants.ConstantsDict attribute*), 328
CUSTOM_TEMPLATES_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 328
CUSTOM_TEMPLATES_ENABLED (*in module archivebox.core.settings*), 230
CustomOutboundWebhookLogFormatter (*class in archivebox.core.settings_logging*), 238
CustomPagination (*class in archivebox.api.v1_core*), 177
CustomPagination.Input (*class in archivebox.api.v1_core*), 177
CustomPagination.Output (*class in archivebox.api.v1_core*), 177
CustomTOMLEncoder (*class in archivebox.misc.toml_util*), 119
CustomUserAdmin (*class in archivebox.core.admin_users*), 285
CustomWebhookAdmin (*class in archivebox.api.admin*), 204

D

DaphneCloseTimeoutFilter (*class in archivebox.core.settings_logging*), 237
DATA_DIR (*archivebox.config.constants.ConstantsDict attribute*), 326
DATA_DIR (*in module archivebox.config.paths*), 334
DATA_DIR (*in module archivebox.config.permissions*), 324
DATA_DIR (*in module archivebox.machine.detect*), 94
DATA_UPLOAD_MAX_MEMORY_SIZE (*in module archivebox.core.settings*), 232
DATA_UPLOAD_MAX_NUMBER_FIELDS (*in module archivebox.core.settings*), 232
database_display_location() (*in module archivebox.misc.db*), 125
DATABASE_ENGINE (*archivebox.config.common.DatabaseConfig attribute*), 342

database_exists() (*in module archivebox.misc.db*), 125
DATABASE_FILE (*archivebox.config.constants.ConstantsDict attribute*), 328
DATABASE_FILE (*in module archivebox.config.paths*), 334
DATABASE_HOST (*archivebox.config.common.DatabaseConfig attribute*), 342
DATABASE_NAME (*archivebox.config.common.DatabaseConfig attribute*), 342
DATABASE_NAME (*in module archivebox.core.settings*), 230
DATABASE_PASSWORD (*archivebox.config.common.DatabaseConfig attribute*), 342
DATABASE_PORT (*archivebox.config.common.DatabaseConfig attribute*), 342
DATABASE_ROUTERS (*in module archivebox.core.settings*), 231
DATABASE_USER (*archivebox.config.common.DatabaseConfig attribute*), 342
DatabaseConfig (*class in archivebox.config.common*), 341
DatabaseWrapper (*class in archivebox.core.sqlite_backend.base*), 222
DATETIME_FORMAT (*in module archivebox.core.settings*), 232
db_table (*archivebox.core.models.SnapshotTag.Meta attribute*), 261
DEBUG (*archivebox.config.common.ShellConfig attribute*), 339
DEBUG (*in module archivebox.core.settings*), 230
DEBUG_REQUESTS_TRACKER (*in module archivebox.core.settings*), 233
DEBUG_TOOLBAR (*in module archivebox.core.settings*), 233
decode_config_inputs() (*in module archivebox.config.configset*), 332
deconstruct() (*archivebox.uuid_compat.CompactUUIDField method*), 378
default (*archivebox.base_models.admin.ConfigOption attribute*), 353
default() (*archivebox.mcp.server.MCPJSONEncoder method*), 296
default() (*archivebox.misc.util.ExtendedEncoder method*), 138
DEFAULT_ABXPKG_LIB_DIR (*archivebox.config.constants.ConstantsDict attribute*), 328
default_auto_field (*archivebox.crawls.apps.CrawlsConfig attribute*), 308
default_auto_field (*archivebox.ldap.apps.LDAPConfig attribute*), 118
default_auto_field (*archivebox.machine.apps.MachineConfig attribute*), 93
default_auto_field (*archivebox.mcp.apps.MCPConfig attribute*), 294
default_auto_field (*archivebox.personas.apps.PersonasConfig attribute*), 304
default_auto_field (*archivebox.plugins.apps.PluginsConfig attribute*), 370
default_auto_field (*archivebox.search.apps.SearchConfig attribute*), 288
default_auto_field (*archivebox.workers.apps.WorkersConfig attribute*), 205
DEFAULT_AUTO_FIELD (*in module archivebox.core.settings*), 230
DEFAULT_CLI_COLORS (*archivebox.config.constants.ConstantsDict attribute*), 329
DEFAULT_CLI_COLORS (*in module archivebox.misc.logging*), 121
DEFAULT_GID (*archivebox.config.constants.ConstantsDict attribute*), 327
DEFAULT_GID (*in module archivebox.config.permissions*), 324
DEFAULT_PERSONA (*archivebox.config.common.ArchivingConfig attribute*), 343
default_retry_at_field (*in module archivebox.workers.models*), 213
default_status_field (*in module archivebox.workers.models*), 213
DEFAULT_TEMPLATES (*in module archivebox.plugins.discovery*), 374
DEFAULT_TMP_DIR (*archivebox.config.constants.ConstantsDict attribute*), 328
DEFAULT_UID (*archivebox.config.constants.ConstantsDict attribute*), 327
DEFAULT_UID (*in module archivebox.config.permissions*), 324
DefaultStatusChoices (*class in archivebox.workers.models*), 212

defer_crawl_for_resources() (*in module archivebox.services.runner*), 364
delete() (*archivebox.core.models.Snapshot method*), 264
delete() (*archivebox.core.models.SnapshotQuerySet method*), 261
DELETE_AFTER (*archivebox.config.common.ArchivingConfig attribute*), 343
delete_after (*archivebox.core.forms.AddLinkForm attribute*), 234
delete_after_final_statuses (*archivebox.base_models.models.ModelWithDeleteAfter attribute*), 351
delete_after_final_statuses (*archivebox.core.models.ArchiveResult attribute*), 272
delete_after_final_statuses (*archivebox.core.models.Snapshot attribute*), 263
delete_after_final_statuses (*archivebox.crawls.models.Crawl attribute*), 312
delete_after_final_statuses (*archivebox.machine.models.Process attribute*), 105
delete_archiveresults() (*in module archivebox.cli.archivebox_archiveresult*), 154
delete_at (*archivebox.base_models.models.ModelWithDeleteAfter attribute*), 351
delete_binaries() (*in module archivebox.cli.archivebox_binary*), 145
delete_cmd() (*in module archivebox.cli.archivebox_archiveresult*), 155
delete_cmd() (*in module archivebox.cli.archivebox_binary*), 145
delete_cmd() (*in module archivebox.cli.archivebox_crawl*), 140
delete_cmd() (*in module archivebox.cli.archivebox_persona*), 150
delete_cmd() (*in module archivebox.cli.archivebox_snapshot*), 161
delete_cmd() (*in module archivebox.cli.archivebox_tag*), 165
delete_crawl() (*in module archivebox.api.v1_crawls*), 171
delete_crawls() (*in module archivebox.cli.archivebox_crawl*), 140
delete_expired() (*archivebox.base_models.models.ModelWithDeleteAfter class method*), 352
delete_output_paths() (*archivebox.base_models.models.ModelWithOutputDir class method*), 352
delete_personas() (*in module archivebox.cli.archivebox_persona*), 150
delete_requested() (*archivebox.core.models.Snapshot class method*), 264
delete_selected_batched() (*archivebox.crawls.admin.CrawlAdmin method*), 319
delete_snapshot() (*in module archivebox.api.v1_core*), 186
delete_snapshot_view() (*archivebox.crawls.admin.CrawlAdmin method*), 319
delete_snapshots() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 285
delete_snapshots() (*in module archivebox.cli.archivebox_snapshot*), 160
delete_tag() (*in module archivebox.api.v1_core*), 190
delete_tag() (*in module archivebox.core.tag_util*), 279
delete_tags() (*in module archivebox.cli.archivebox_tag*), 165
delete_url (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
deleted_count (*archivebox.api.v1_core.SnapshotDeleteResponseSchema attribute*), 184
deleted_count (*archivebox.api.v1_core.TagDeleteResponseSchema attribute*), 189
deleted_count (*archivebox.api.v1_crawls.CrawlDeleteResponseSchema attribute*), 170
deleted_snapshots (*archivebox.api.v1_crawls.CrawlDeleteResponseSchema attribute*), 170
DELETING_STATE (*archivebox.core.models.Snapshot attribute*), 262
depth (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
depth (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 202
depth (*archivebox.api.v1_core.SnapshotCreateSchema attribute*), 184
depth (*archivebox.core.forms.AddLinkForm attribute*), 234
depth (*archivebox.core.models.Snapshot attribute*), 262
depth (*archivebox.machine.models.Process property*), 107
depth() (*archivebox.crawls.admin.CrawlAdmin method*), 319
DEPTH_CHOICES (*in module archivebox.core.forms*), 233
derive_base_url_from_csrf() (*in module archivebox.core.routes_util*), 256
derive_filter_toggles() (*archivebox.crawls.admin.CrawlAdminForm class method*), 317

derive_persona_config() (*in module archivebox.personas.models*), 305
derive_plugin_enabled_config() (*archivebox.config.common.ArchiveBoxBaseConfig method*), 345
description (*archivebox.base_models.admin.ConfigOption attribute*), 353
DESTRUCTIVE_ACTIONS (*in module archivebox.mcp.server*), 296
details_link() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
detect_installed_version() (*in module archivebox.config.version*), 348
detect_timezone() (*in module archivebox.core.middleware*), 277
did_work (*archivebox.personas.importers.PersonaImportResult property*), 301
dir_is_writable() (*in module archivebox.config.paths*), 335
direct_snapshots_queryset() (*in module archivebox.core.permissions*), 240
directory_name() (*archivebox.personas.models.Persona static method*), 306
DIRECTORY_PREVIEW_MAX_BYTES (*in module archivebox.misc.serve_static*), 128
DISABLED_CLI_COLORS (*archivebox.config.constants.ConstantsDict attribute*), 329
discover_hooks() (*in module archivebox.plugins.hooks*), 377
discover_local_browser_profiles() (*in module archivebox.personas.importers*), 301
discover_outputs() (*archivebox.core.models.Snapshot method*), 271
discover_persona_template_profiles() (*in module archivebox.personas.importers*), 301
discover_plugin_configs() (*in module archivebox.plugins.discovery*), 374
disk_backed_swap_free_bytes() (*in module archivebox.services.resource_admission*), 360
dispatch() (*archivebox.core.views.WebAddView method*), 248
dispatch() (*archivebox.crawls.models.CrawlSchedule method*), 310
display_label (*archivebox.personas.importers.PersonaImportSource property*), 300
display_plugin_name() (*in module archivebox.plugins.output_groups*), 371
django_application (*in module archivebox.core.asgi*), 287
DJANGO_OBJECT_ACTIONS_DEFAULT_HTTP_METHOD (*in module archivebox.core.settings*), 230
DJANGO_SET_UP (*in module archivebox.config.django*), 336
dns_server (*archivebox.machine.models.NetworkInterface attribute*), 98
docstring() (*in module archivebox.misc.util*), 137
domain (*in module archivebox.misc.util*), 136
domain() (*archivebox.core.models.Snapshot method*), 269
download_url() (*in module archivebox.misc.util*), 137
downloaded_at (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
downloaded_at (*archivebox.api.v1_core.TagSearchSnapshotSchema attribute*), 188
downloaded_at (*archivebox.core.models.Snapshot attribute*), 262
downloaded_datestr() (*archivebox.core.models.Snapshot method*), 270
drain_old_archive_dirs() (*in module archivebox.cli.archivebox_update*), 146
drain_snapshot_tasks() (*archivebox.services.runner.CrawlRunner method*), 363
drop_models_on_postgres() (*in module archivebox.misc.db*), 126
drop_privileges() (*in module archivebox.config.permissions*), 325
duration_display() (*archivebox.machine.admin.ProcessAdmin method*), 115

E

each_context() (*archivebox.core.admin_site.ArchiveBoxAdmin method*), 254
edit_string_for_tag_names() (*in module archivebox.core.forms*), 233
edit_url (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
effective_available_memory_bytes() (*in module archivebox.workers.supervisord_util*), 208
effective_only_new() (*archivebox.crawls.admin.CrawlAdminForm static method*), 317
EGID (*in module archivebox.config.permissions*), 324
EMAIL_BACKEND (*in module archivebox.core.settings*), 231
embed_path() (*archivebox.core.models.ArchiveResult method*), 275
embed_path_db() (*archivebox.core.models.ArchiveResult method*),

275
EMITS (*archivebox.services.archive_result_service.ArchiveResultService attribute*), 359
EMITS (*archivebox.services.binary_service.ArchiveBoxBinaryService attribute*), 357
EMITS (*archivebox.services.crawl_service.CrawlService attribute*), 367
EMITS (*archivebox.services.machine_service.MachineService attribute*), 366
EMITS (*archivebox.services.process_service.ProcessService attribute*), 356
EMITS (*archivebox.services.snapshot_service.SnapshotService attribute*), 365
EMITS (*archivebox.services.tag_service.TagService attribute*), 366
empty_cookie_file_is_unset() (*archivebox.config.common.ArchivingConfig class method*), 343
enabled (*archivebox.plugins.discovery.PluginSpecialConfig attribute*), 373
ENABLES_FULL_JS_REPLAY (*archivebox.config.common.ServerConfig property*), 341
end() (*archivebox.misc.logging_util.TimedProgress method*), 122
end_ts (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
end_ts (*archivebox.core.models.ArchiveResult attribute*), 273
ended_at (*archivebox.machine.models.Process attribute*), 104
ENFORCE_ATOMIC_WRITES (*archivebox.config.common.StorageConfig attribute*), 340
enforce_types() (*in module archivebox.misc.util*), 137
enqueue() (*archivebox.crawls.models.CrawlSchedule method*), 310
enqueue_discovered_snapshots_from_outputs() (*archivebox.services.runner.CrawlRunner method*), 363
enqueue_pending_snapshots_from_projection() (*archivebox.services.runner.CrawlRunner method*), 363
enqueue_snapshot() (*archivebox.services.runner.CrawlRunner method*), 363
ensure_background_runner() (*in module archivebox.services.runner*), 362
ensure_crawl_symlink() (*archivebox.core.models.Snapshot method*), 269
ensure_daemon_stack() (*in module archivebox.core.takeover_util*), 241
ensure_data_dir_lib_symlink() (*in module archivebox.cli.archivebox_install*), 158
ensure_database_ready() (*in module archivebox.misc.db*), 125
ensure_dirs() (*archivebox.personas.models.Persona method*), 306
ensure_log_files() (*archivebox.machine.models.Process method*), 109
ensure_path_within_personas_dir() (*in module archivebox.cli.archivebox_persona*), 149
ensure_permissions_config() (*archivebox.core.models.Snapshot method*), 265
enter_single_runner_gate() (*in module archivebox.core.takeover_util*), 241
enum (*archivebox.base_models.admin.ConfigOption attribute*), 353
env (*archivebox.machine.models.Process attribute*), 104
env_display() (*archivebox.machine.admin.ProcessAdmin method*), 115
env_to_dotenv_text() (*in module archivebox.machine.env_util*), 112
env_to_shell_exports() (*in module archivebox.machine.env_util*), 112
ENVIRONMENT_BINARIES_BASE_URL (*in module archivebox.config.views*), 330
ERROR_LOG (*in module archivebox.core.settings_logging*), 238
errors (*archivebox.api.v1_cli.CLICommandResponseSchema attribute*), 199
escape_like_query() (*in module archivebox.search.query*), 291
EUID (*in module archivebox.config.permissions*), 324
EVENT_FLOW_DIAGRAM (*in module archivebox.cli.archivebox_pluginmap*), 163
every (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 202
exclude_domain() (*archivebox.crawls.models.Crawl method*), 314
exclude_domain_view() (*archivebox.crawls.admin.CrawlAdmin method*), 319
execute() (*archivebox.core.sqlite_backend.base.SQLiteCursorWrapper method*), 222
execute_click_command() (*in module archivebox.mcp.server*), 297
executemany() (*archivebox.core.sqlite_backend.base.SQLiteCursorWrapper method*), 222
execution_interrupted() (*archivebox.services.runner.CrawlRunner method*), 362
exit_code (*archivebox.machine.models.Process attribute*), 104
EXITED (*archivebox.machine.models.Process.StatusChoices attribute*), 103

expires (*archivebox.api.models.APIToken attribute*), 191
export_browser_state() (*in module archivebox.personas.importers*), 302
export_jsonl_url (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
export_profile_cookies() (*in module archivebox.personas.importers*), 302
export_tag_snapshots_jsonl() (*in module archivebox.core.tag_util*), 279
export_tag_urls() (*in module archivebox.core.tag_util*), 279
export_urls_url (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
extend_choices() (*archivebox.workers.models.ModelWithQueue class method*), 214
ExtendedEncoder (*class in archivebox.misc.util*), 137
extension (*in module archivebox.misc.util*), 136
extension() (*archivebox.core.models.Snapshot method*), 270
extension_persona_id (*archivebox.api.v1_personas.PersonaSyncSchema attribute*), 174
extra (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 224
extra (*archivebox.core.admin_tags.TagInline attribute*), 251
extract_domain_from_url() (*archivebox.core.models.Snapshot static method*), 266
extract_url_line() (*archivebox.crawls.admin.CrawlAdminForm static method*), 317

F

FAILED (*archivebox.core.models.ArchiveResult.StatusChoices attribute*), 272
FALLBACK_GID (*in module archivebox.config.permissions*), 325
FALLBACK_UID (*in module archivebox.config.permissions*), 325
FAVICON_CACHE_CONTROL (*in module archivebox.misc.serve_static*), 128
FAVICON_FILENAME (*archivebox.config.constants.ConstantsDict attribute*), 328
favicon_url (*archivebox.api.v1_core.TagSearchSnapshotSchema attribute*), 187
fields (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 224
fields (*archivebox.core.admin_snapshots.SnapshotAdminForm.Meta attribute*), 282
fields (*archivebox.core.admin_tags.TagAdminForm.Meta attribute*), 251
fields (*archivebox.core.admin_tags.TagInline attribute*), 251
fields (*archivebox.crawls.admin.CrawlAdminForm.Meta attribute*), 317
fields (*archivebox.personas.forms.PersonaAdminForm.Meta attribute*), 303
fieldsets (*archivebox.api.admin.APITokenAdmin attribute*), 203
fieldsets (*archivebox.api.admin.CustomWebhookAdmin attribute*), 204
fieldsets (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
fieldsets (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
fieldsets (*archivebox.core.admin_tags.TagAdmin attribute*), 252
fieldsets (*archivebox.core.admin_users.CustomUserAdmin attribute*), 286
fieldsets (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
fieldsets (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
fieldsets (*archivebox.machine.admin.BinaryAdmin attribute*), 114
fieldsets (*archivebox.machine.admin.MachineAdmin attribute*), 113
fieldsets (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 113
fieldsets (*archivebox.machine.admin.ProcessAdmin attribute*), 115
file_size() (*in module archivebox.core.templatetags.core_tags*), 219
files() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
FILESIZE_UNITS (*in module archivebox.misc.util*), 137
filter() (*archivebox.core.models.SnapshotManager method*), 262
filter() (*archivebox.core.settings_logging.AsyncioCancelledShieldFilter method*), 238
filter() (*archivebox.core.settings_logging.DaphneCloseTimeoutFilter method*), 237
filter() (*archivebox.core.settings_logging.NoisyRequestsFilter method*), 237
filter() (*archivebox.core.settings_logging.StripANSIColorCodesFilter method*), 238
FILTER_ARG_KEYS (*archivebox.core.models.SnapshotQuerySet attribute*), 261
filter_by_patterns() (*archivebox.core.models.SnapshotQuerySet method*),

261
filter_patterns (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 200
FILTER_PATTERNS_EXAMPLES (*in module archivebox.api.v1_cli*), 199
filter_personas_by_permissions() (*in module archivebox.core.permissions*), 239
filter_queryset_by_uuid_substring() (*in module archivebox.misc.util*), 136
filter_search() (*archivebox.api.v1_core.SnapshotFilterSchema method*), 185
filter_search_mode() (*archivebox.api.v1_core.SnapshotFilterSchema method*), 185
filter_snapshots_by_permissions() (*in module archivebox.core.permissions*), 239
filter_snapshots_by_status() (*in module archivebox.core.snapshot_status*), 236
filter_status() (*archivebox.api.v1_core.SnapshotFilterSchema method*), 185
filter_type (*archivebox.api.v1_cli.RemoveCommandSchema attribute*), 202
filter_type (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 200
FILTER_TYPE_CHOICES (*archivebox.core.models.SnapshotQuerySet attribute*), 261
FILTER_TYPES (*archivebox.core.models.SnapshotQuerySet attribute*), 261
filter_url (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
FilterTypeChoices (*in module archivebox.api.v1_cli*), 199
FINAL_OR_ACTIVE_STATES (*archivebox.core.models.ArchiveResult attribute*), 272
FINAL_OR_ACTIVE_STATES (*archivebox.core.models.Snapshot attribute*), 263
FINAL_OR_ACTIVE_STATES (*archivebox.crawls.models.Crawl attribute*), 312
FINAL_OR_ACTIVE_STATES (*archivebox.machine.models.Binary attribute*), 100
FINAL_OR_ACTIVE_STATES (*archivebox.workers.models.ModelWithQueue attribute*), 214
FINAL_STATES (*archivebox.core.models.ArchiveResult attribute*), 272
FINAL_STATES (*archivebox.core.models.Snapshot attribute*), 263
FINAL_STATES (*archivebox.crawls.models.Crawl attribute*), 312
FINAL_STATES (*archivebox.machine.models.Binary attribute*), 100
FINAL_STATES (*archivebox.workers.models.ModelWithQueue attribute*), 213
finalize_completed_snapshot() (*in module archivebox.services.snapshot_service*), 365
finalize_completed_upload_results() (*archivebox.core.models.Snapshot method*), 265
finalize_output_metadata() (*archivebox.core.models.Snapshot method*), 269
finalize_run_state() (*archivebox.services.runner.CrawlRunner method*), 363
find_all_urls() (*in module archivebox.misc.util*), 137
find_and_merge_duplicates() (*archivebox.core.models.Snapshot class method*), 268
find_config_default() (*in module archivebox.config.common*), 346
find_config_section() (*in module archivebox.config.common*), 346
find_config_source() (*in module archivebox.config.common*), 346
find_config_type() (*in module archivebox.config.common*), 346
find_named() (*archivebox.personas.models.Persona class method*), 306
find_persona() (*in module archivebox.api.v1_personas*), 175
find_plugin_for_config_key() (*in module archivebox.plugins.views*), 376
find_snapshots_for_id() (*archivebox.core.views.SnapshotView static method*), 245
find_snapshots_for_url() (*archivebox.core.views.SnapshotView static method*), 245
flush() (*archivebox.misc.serve_static._StreamingQueueWriter method*), 129
flush_completed() (*archivebox.services.process_service.ProcessService method*), 356
flush_missing_finalizers() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
flush_search_index() (*in module archivebox.search.query*), 292
FOOTER_INFO (*archivebox.config.common.ServerConfig attribute*), 341
for_crawl() (*archive-*

box.config.common.ArchiveBoxBaseConfig method), 345
for_crawl_frozen() (*archivebox.config.common.ArchiveBoxBaseConfig method*), 345
for_crawl_runtime() (*archivebox.config.common.ArchiveBoxBaseConfig method*), 345
foreground (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 201
foreground_parent_watchdog() (*in module archivebox.core.shutdown_util*), 251
foreground_runner_owner() (*in module archivebox.core.takeover_util*), 241
foreground_shutdown_signals() (*in module archivebox.core.shutdown_util*), 250
form (*archivebox.api.admin.CustomWebhookAdmin attribute*), 204
form (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
form (*archivebox.core.admin_tags.TagAdmin attribute*), 252
form (*archivebox.crawls.admin.CrawlAdmin attribute*), 317
form (*archivebox.personas.admin.PersonaAdmin attribute*), 307
form_class (*archivebox.core.views.AddView attribute*), 247
form_valid() (*archivebox.core.views.AddView method*), 248
format() (*archivebox.core.settings_logging.CustomOutboundWebhookLogFormatter method*), 238
format_duration() (*in module archivebox.misc.logging_util*), 122
format_parsed_datetime() (*in module archivebox.config.views*), 330
format_runtime_components() (*in module archivebox.workers.supervisord_util*), 210
format_value() (*archivebox.core.forms.TagWidget method*), 235
formfield_for_dbfield() (*archivebox.base_models.admin.ConfigEditorMixin method*), 354
fragment (*in module archivebox.misc.util*), 136
from_choice_value() (*archivebox.personas.importers.PersonaImportSource class method*), 300
from_db_value() (*archivebox.uuid_compat.CompactUUIDField method*), 378
from_json() (*archivebox.core.models.ArchiveResult static method*), 274
from_json() (*archivebox.core.models.Snapshot static method*), 269
from_json() (*archivebox.core.models.Tag static method*), 260
from_json() (*archivebox.crawls.models.Crawl static method*), 313
from_json() (*archivebox.machine.models.Binary static method*), 101
from_json() (*archivebox.machine.models.Machine static method*), 97
from_json() (*archivebox.machine.models.Process static method*), 106
fs_migration_needed (*archivebox.core.models.Snapshot property*), 266
fs_version (*archivebox.core.models.Snapshot attribute*), 262

G

GeneralConfig (*class in archivebox.config.common*), 340
generate_secret_token() (*in module archivebox.api.models*), 190
generated_host_allowlist() (*archivebox.crawls.admin.CrawlAdminForm class method*), 317
generated_subpath_allowlist() (*archivebox.crawls.admin.CrawlAdminForm class method*), 317
generic_exception_handler() (*in module archivebox.api.v1_api*), 173
geolocation (*archivebox.api.v1_personas.PersonaBrowserSettingsSchema attribute*), 174
get() (*archivebox.config.configset.BaseConfigSet method*), 333
get() (*archivebox.core.views.HealthCheckView method*), 248
get() (*archivebox.core.views.HomepageView method*), 245
get() (*archivebox.core.views.OriginalDomainHostView method*), 246
get() (*archivebox.core.views.OriginalDomainReplayView method*), 246
get() (*archivebox.core.views.PublicIndexView method*), 247
get() (*archivebox.core.views.SnapshotHostView method*), 246
get() (*archivebox.core.views.SnapshotPathView method*), 245
get() (*archivebox.core.views.SnapshotReplayAuthView method*), 244
get() (*archivebox.core.views.SnapshotReplayView method*), 246
get() (*archivebox.core.views.SnapshotView method*), 245
get() (*archivebox.core.views.WebAddView method*), 248

get() (*archivebox.plugins.discovery.ConfigLookup method*), 373
get() (*archivebox.workers.supervisord_util.SupervisordConnectionProxy method*), 210
get_absolute_url() (*archivebox.core.models.ArchiveResult method*), 275
get_absolute_url() (*archivebox.core.models.Snapshot method*), 269
get_actions() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
get_actions() (*archivebox.crawls.admin.CrawlAdmin method*), 319
get_admin_base_url() (*in module archivebox.core.routes_util*), 258
get_admin_data_urls() (*archivebox.core.admin_site.ArchiveBoxAdmin method*), 254
get_admin_host() (*in module archivebox.core.routes_util*), 257
get_admin_search_cache_key() (*in module archivebox.search.views*), 290
get_admin_toolbar_actions() (*archivebox.base_models.admin.BaseModelAdmin method*), 354
get_admin_toolbar_actions() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
get_all_configs() (*in module archivebox.config.common*), 346
get_any() (*in module archivebox.api.v1_core*), 187
get_api_base_url() (*in module archivebox.core.routes_util*), 258
get_api_host() (*in module archivebox.core.routes_util*), 257
get_api_token() (*in module archivebox.api.v1_auth*), 192
get_app_list() (*archivebox.core.admin_site.ArchiveBoxAdmin method*), 254
get_archive_result_aliases() (*in module archivebox.plugins.discovery*), 373
get_archive_result_names() (*in module archivebox.plugins.discovery*), 373
get_archiveresult() (*in module archivebox.api.v1_core*), 181
get_archiveresults() (*in module archivebox.api.v1_core*), 181
get_available_backends() (*in module archivebox.search.backends*), 289
get_backend() (*in module archivebox.search.backends*), 289
get_base_host() (*in module archivebox.core.routes_util*), 257
get_base_url() (*in module archivebox.core.routes_util*), 257
get_base_url_mismatch_context() (*in module archivebox.core.setup_wizard*), 253
get_binaries() (*in module archivebox.api.v1_machine*), 198
get_binaries_by_name() (*in module archivebox.api.v1_machine*), 198
get_binary() (*in module archivebox.api.v1_machine*), 198
get_brave_user_data_dir() (*in module archivebox.personas.importers*), 301
get_BUILD_TIME() (*in module archivebox.config.version*), 348
get_by_pid() (*archivebox.machine.models.ProcessManager method*), 102
get_cached_admin_search_ids() (*in module archivebox.search.views*), 290
get_cached_public_search_ids() (*in module archivebox.search.views*), 290
get_cached_public_search_state() (*in module archivebox.search.views*), 290
get_canonical_name() (*archivebox.cli.ArchiveBoxGroup class method*), 167
get_changelist() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
get_changelist() (*archivebox.search.admin.SearchResultsAdminMixin method*), 294
get_children_pids() (*archivebox.machine.models.Process method*), 108
get_choice_field() (*in module archivebox.plugins.forms*), 368
get_chrome_user_data_dir() (*in module archivebox.personas.importers*), 301
get_chromium_user_data_dir() (*in module archivebox.personas.importers*), 301
get_code_locations() (*in module archivebox.config.paths*), 335
get_collection_id() (*in module archivebox.config.paths*), 334
get_command() (*archivebox.cli.ArchiveBoxGroup method*), 167
get_COMMIT_HASH() (*in module archivebox.config.version*), 348
get_config() (*in module archivebox.config.common*), 346
get_config_definition_link() (*in module archivebox.plugins.views*), 376
get_context_data() (*archivebox.core.views.AddView*

method), 247
get_context_data() (*archivebox.core.views.PublicIndexView method*), 247
get_cpu_percent() (*archivebox.machine.models.Process method*), 108
get_crawl() (*in module archivebox.api.v1_crawls*), 170
get_crawl_by_ref() (*in module archivebox.api.v1_crawls*), 170
get_crawls() (*in module archivebox.api.v1_crawls*), 170
get_current_config() (*archivebox.crawls.models.Crawl method*), 313
get_current_machine() (*in module archivebox.api.v1_machine*), 198
get_current_pid_namespace() (*in module archivebox.machine.models*), 96
get_data_locations() (*in module archivebox.config.paths*), 335
get_db_binaries_by_name() (*in module archivebox.config.views*), 330
get_default_search_mode() (*archivebox.base_models.admin.BaseModelAdmin method*), 355
get_default_search_mode() (*archivebox.search.admin.SearchResultsAdminMixin method*), 294
get_default_search_mode() (*in module archivebox.search.config*), 293
get_delete_after_config_value() (*archivebox.base_models.models.ModelWithDeleteAfter method*), 351
get_delete_after_config_value() (*archivebox.core.models.ArchiveResult method*), 274
get_delete_after_config_value() (*archivebox.core.models.Snapshot method*), 265
get_delete_after_config_value() (*archivebox.crawls.models.Crawl method*), 312
get_delete_after_config_value() (*archivebox.machine.models.Process method*), 105
get_derived_config() (*archivebox.personas.models.Persona method*), 306
get_descendants() (*archivebox.machine.models.Process method*), 107
get_detail_page_auxiliary_items() (*archivebox.core.models.Snapshot method*), 271
get_dir_size() (*in module archivebox.misc.system*), 120
get_edge_user_data_dir() (*in module archivebox.personas.importers*), 301
get_enabled_plugins() (*in module archivebox.plugins.discovery*), 374
get_environment_binary_url() (*in module archivebox.config.views*), 330
get_existing_supervisord_process() (*in module archivebox.workers.supervisord_util*), 210
get_extra_snapshot_output_cards() (*in module archivebox.plugins.discovery*), 374
get_field_value() (*archivebox.config.configset.IniConfigSettingsSource method*), 332
get_fieldsets() (*archivebox.core.admin_tags.TagAdmin method*), 252
get_fieldsets() (*archivebox.crawls.admin.CrawlAdmin method*), 319
get_fieldsets() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
get_fieldsets() (*archivebox.personas.admin.PersonaAdmin method*), 308
get_filesystem_plugins() (*in module archivebox.plugins.views*), 376
get_filters_params() (*archivebox.search.admin.SearchResultsChangeList method*), 294
get_form() (*archivebox.base_models.admin.BaseModelAdmin method*), 355
get_form() (*archivebox.personas.admin.PersonaAdmin method*), 308
get_form_kwargs() (*archivebox.core.views.AddView method*), 247
get_formset() (*archivebox.core.admin_archiveresults.ArchiveResultInline method*), 224
get_from_db_or_cache() (*archivebox.machine.models.BinaryManager method*), 99
get_group_by_cols() (*archivebox.core.models.UngroupedSubquery method*), 259
get_host_guid() (*in module archivebox.machine.detect*), 94
get_host_network() (*in module archivebox.machine.detect*), 94
get_host_stats() (*in module archivebox.machine.detect*), 94
get_html_details_context() (*archivebox.core.models.Snapshot method*), 271
get_initial() (*archivebox.core.views.AddView method*), 247
get_installed_binary_change_url() (*in module archivebox.config.views*), 330
get_isp_info() (*in module archivebox.machine.detect*), 94
get_list_display() (*archive-*

box.core.admin_users.CustomUserAdmin method), 286
get_listen_host() (*in module archivebox.core.routes_util*), 256
get_listen_subdomain() (*in module archivebox.core.routes_util*), 257
get_live_config_url() (*in module archivebox.config.common*), 346
get_local_ip() (*in module archivebox.machine.detect*), 94
get_machine() (*in module archivebox.api.v1_machine*), 198
get_machine_admin_url() (*in module archivebox.plugins.views*), 376
get_machine_id() (*in module archivebox.config.paths*), 334
get_machine_type() (*in module archivebox.config.paths*), 335
get_machines() (*in module archivebox.api.v1_machine*), 198
get_matching_tags() (*in module archivebox.core.tag_util*), 278
get_memory_info() (*archivebox.machine.models.Process method*), 108
get_next_worker_id() (*archivebox.machine.models.Process class method*), 111
get_openapi_schema() (*archivebox.api.v1_api.NinjaAPIWithIOCapture method*), 173
get_or_create_api_token() (*in module archivebox.api.auth*), 194
get_or_create_archivebox_account() (*in module archivebox.config.permissions*), 325
get_or_create_by_hook() (*archivebox.core.models.ArchiveResult class method*), 272
get_or_create_by_name() (*archivebox.core.models.Tag class method*), 260
get_or_create_default() (*archivebox.personas.models.Persona class method*), 307
get_or_create_named() (*archivebox.personas.models.Persona class method*), 306
get_or_create_supervisord_process() (*in module archivebox.workers.supervisord_util*), 210
get_or_create_system_user_pk() (*in module archivebox.base_models.models*), 350
get_or_create_tag() (*in module archivebox.core.tag_util*), 279
get_or_create_working_lib_dir() (*in module archivebox.config.paths*), 335
get_or_create_working_tmp_dir() (*in module archivebox.config.paths*), 335
get_ordering() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
get_original_host() (*in module archivebox.core.routes_util*), 257
get_os_info() (*in module archivebox.machine.detect*), 94
get_output_files_url() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
get_output_view_url() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
get_output_zip_url() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
get_paginate_by() (*archivebox.core.views.PublicIndexView method*), 247
get_parent_object_from_request() (*archivebox.core.admin_archiveresults.ArchiveResultInline method*), 224
get_personas() (*in module archivebox.api.v1_personas*), 175
get_plugin_admin_url() (*in module archivebox.core.admin_archiveresults*), 223
get_plugin_catalog() (*in module archivebox.plugins.discovery*), 373
get_plugin_choice_label() (*in module archivebox.plugins.forms*), 368
get_plugin_choices() (*archivebox.core.models.ArchiveResult class method*), 272
get_plugin_choices() (*in module archivebox.plugins.forms*), 368
get_plugin_config_binary_urls() (*in module archivebox.plugins.forms*), 369
get_plugin_config_resolver() (*in module archivebox.plugins.discovery*), 373
get_plugin_default_output_path() (*in module archivebox.plugins.discovery*), 373
get_plugin_docs_url() (*in module archivebox.plugins.views*), 376
get_plugin_hook_source_url() (*in module archivebox.plugins.views*), 376
get_plugin_icon() (*in module archivebox.plugins.discovery*), 375
get_plugin_name() (*in module archivebox.plugins.discovery*), 373
get_plugin_output_extension_preference() (*in module archivebox.plugins.discovery*), 373
get_plugin_presentation_module() (*in module archivebox.plugins.discovery*), 374

get_plugin_presentation_modules() (*in module archivebox.plugins.discovery*), 374
get_plugin_special_config() (*in module archivebox.plugins.discovery*), 374
get_plugin_template() (*in module archivebox.plugins.discovery*), 375
get_plugins() (*in module archivebox.plugins.discovery*), 373
get_process_link_label() (*in module archivebox.core.admin_archiveresults*), 223
get_progress_stats() (*archivebox.core.models.Snapshot method*), 270
get_public_ip() (*in module archivebox.machine.detect*), 94
get_public_search_cache_key() (*in module archivebox.search.views*), 290
get_public_tool_definitions() (*archivebox.mcp.server.MCPServer method*), 298
get_queryset() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
get_queryset() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
get_queryset() (*archivebox.core.admin_users.CustomUserAdmin method*), 286
get_queryset() (*archivebox.core.models.SnapshotManager method*), 262
get_queryset() (*archivebox.core.views.PublicIndexView method*), 247
get_queryset() (*archivebox.crawls.admin.CrawlAdmin method*), 318
get_queryset() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
get_queryset() (*archivebox.machine.admin.ProcessAdmin method*), 115
get_queue() (*archivebox.workers.models.ModelWithQueue class method*), 214
get_readonly_fields() (*archivebox.core.admin_archiveresults.ArchiveResultInline method*), 224
get_records() (*archivebox.machine.models.Process method*), 106
get_redirect_url() (*archivebox.core.admin_site.ArchiveBoxLoginView method*), 253
get_request_config() (*in module archivebox.config.common*), 346
get_results() (*archivebox.core.admin_snapshots.SnapshotChangeList method*), 281
get_results() (*archivebox.search.admin.SearchResultsChangeList method*), 294
get_running() (*archivebox.machine.models.Process class method*), 110
get_running_count() (*archivebox.machine.models.Process class method*), 110
get_search_backends() (*in module archivebox.plugins.discovery*), 374
get_search_mode() (*in module archivebox.search.config*), 293
get_search_mode_backend() (*in module archivebox.search.config*), 293
get_search_mode_base() (*in module archivebox.search.config*), 293
get_search_mode_options() (*archivebox.search.admin.SearchResultsAdminMixin method*), 294
get_search_mode_options() (*in module archivebox.search.config*), 293
get_search_results() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
get_search_results() (*archivebox.search.admin.SearchResultsAdminMixin method*), 294
get_setup_wizard_context() (*in module archivebox.core.setup_wizard*), 253
get_snapshot() (*in module archivebox.api.v1_core*), 186
get_snapshot_base_url() (*in module archivebox.core.routes_util*), 258
get_snapshot_files_url() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
get_snapshot_host() (*in module archivebox.core.routes_util*), 257
get_snapshot_lookup_key() (*in module archivebox.core.routes_util*), 257
get_snapshot_output_anchor() (*in module archivebox.core.routes_util*), 258
get_snapshot_permissions() (*in module archivebox.core.permissions*), 239
get_snapshot_role_names() (*in module archivebox.plugins.discovery*), 374
get_snapshot_role_result() (*in module archivebox.plugins.discovery*), 374
get_snapshot_subdomain() (*in module archivebox.core.routes_util*), 257

get_snapshot_thumbnail_card_order() (*in module archivebox.plugins.discovery*), 374
get_snapshot_view_url() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
get_snapshot_view_url() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
get_snapshot_zip_url() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
get_snapshots() (*in module archivebox.api.v1_core*), 185
get_snapshots_rss() (*in module archivebox.api.v1_core*), 186
get_sock_file() (*in module archivebox.workers.supervisord_util*), 209
get_sonic_supervisord_worker_from_plugin() (*in module archivebox.workers.supervisord_util*), 211
get_static_cache_key() (*in module archivebox.core.context_processors*), 242
get_storage_path_for_version() (*archivebox.core.models.Snapshot method*), 266
get_tag() (*in module archivebox.api.v1_core*), 187
get_tag_by_ref() (*in module archivebox.core.tag_util*), 278
get_tag_creator_choices() (*in module archivebox.core.tag_util*), 278
get_tag_year_choices() (*in module archivebox.core.tag_util*), 278
get_tags() (*in module archivebox.api.v1_core*), 187
get_tools() (*archivebox.mcp.server.MCPServer method*), 297
get_url_allowlist() (*archivebox.crawls.models.Crawl method*), 313
get_url_denylist() (*archivebox.crawls.models.Crawl method*), 313
get_urls() (*archivebox.base_models.admin.BaseModelAdmin method*), 355
get_urls() (*archivebox.core.admin_site.ArchiveBoxAdmin method*), 254
get_urls() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
get_urls() (*archivebox.crawls.admin.CrawlAdmin method*), 319
get_urls_list() (*archivebox.crawls.models.Crawl method*), 313
get_valid_binary() (*archivebox.machine.models.BinaryManager method*), 99
get_vm_info() (*in module archivebox.machine.detect*), 94
get_web_base_url() (*in module archivebox.core.routes_util*), 258
get_web_host() (*in module archivebox.core.routes_util*), 257
get_worker() (*in module archivebox.workers.supervisord_util*), 210
GLOBAL_CONTEXT (*in module archivebox.core.admin_snapshots*), 280
grant_archivebox_group_traversal() (*in module archivebox.config.permissions*), 325
grid_file_size() (*in module archivebox.core.templatetags.core_tags*), 219
grid_view() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
guid (*archivebox.api.v1_machine.MachineSchema attribute*), 195
guid (*archivebox.machine.models.Machine attribute*), 96

H

handle() (*archivebox.core.management.commands.archivebox.Command method*), 221
handle() (*archivebox.workers.management.commands.runner_watch.Com method*), 205
handle() (*archivebox.workers.management.commands.supervisord_watch method*), 205
handle_initialize() (*archivebox.mcp.server.MCPServer method*), 298
handle_request() (*archivebox.mcp.server.MCPServer method*), 298
handle_tools_call() (*archivebox.mcp.server.MCPServer method*), 298
handle_tools_list() (*archivebox.mcp.server.MCPServer method*), 298
handoff_root_owned_data_dir() (*in module archivebox.config.permissions*), 325
has_changed() (*archivebox.core.forms.TagField method*), 235
has_finished_snapshots() (*archivebox.crawls.models.Crawl method*), 315
has_remaining_snapshot_capacity() (*archivebox.crawls.models.Crawl method*), 314
has_remaining_url_capacity() (*archivebox.crawls.models.Crawl method*), 314
has_snapshots (*archivebox.api.v1_core.TagSearchResponseSchema attribute*), 188
hashes_index() (*archivebox.core.models.Snapshot method*), 269
header (*archivebox.core.middleware.ReverseProxyAuthMiddleware attribute*), 277
HeaderTokenAuth (*class in archivebox.api.auth*), 194
health (*archivebox.base_models.models.ModelWithHealthStats property*), 351

health_display() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
health_display() (*archivebox.machine.admin.BinaryAdmin method*), 114
health_display() (*archivebox.machine.admin.MachineAdmin method*), 113
health_display() (*archivebox.machine.admin.NetworkInterfaceAdmin method*), 114
HealthCheckView (*class in archivebox.core.views*), 248
heartbeat() (*archivebox.machine.models.Process method*), 106
heartbeat_active_leases() (*archivebox.services.runner.CrawlRunner method*), 363
help (*archivebox.core.management.commands.archivebox.Command attribute*), 221
help (*archivebox.workers.management.commands.runner_watch.Command attribute*), 204
help (*archivebox.workers.management.commands.supervisor_logging.Command attribute*), 205
help() (*in module archivebox.cli.archivebox_help*), 143
HexUUIDConverter (*class in archivebox.base_models.admin*), 353
HIDDEN_CARD_PLUGINS (*in module archivebox.plugins.output_groups*), 371
HIDDEN_ICON_PLUGINS (*in module archivebox.plugins.output_groups*), 371
highlight() (*archivebox.misc.logging.RainbowHighlighter method*), 121
hint() (*in module archivebox.misc.logging*), 121
HISTORICAL_GHOST_MIGRATIONS (*in module archivebox.misc.db*), 126
HomepageView (*class in archivebox.core.views*), 245
HOOK (*archivebox.machine.models.Process.TypeChoices attribute*), 104
hook_name (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
hook_name (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
hook_name (*archivebox.core.models.ArchiveResult attribute*), 273
hook_name (*archivebox.machine.models.Process property*), 106
hook_script_name (*archivebox.machine.models.Process property*), 108
host_matches() (*in module archivebox.core.routes_util*), 257
host_swap_free_bytes() (*in module archivebox.services.resource_admission*), 360
hostname (*archivebox.api.v1_machine.MachineFilterSchema attribute*), 196
hostname (*archivebox.api.v1_machine.MachineSchema attribute*), 195
hostname (*archivebox.machine.models.Machine attribute*), 96
hostname (*archivebox.machine.models.NetworkInterface attribute*), 98
HOSTNAME (*in module archivebox.config.permissions*), 325
HostRoutingMiddleware() (*in module archivebox.core.middleware*), 277
HTML_BODY_RE (*in module archivebox.misc.serve_static*), 130
html_description (*in module archivebox.api.v1_api*), 172
HTML_INDEX_FILENAME (*archivebox.config.constants.ConstantsDict attribute*), 328
HTML_TAG_RE (*in module archivebox.misc.serve_static*), 130
htmldecode (*in module archivebox.misc.util*), 136
htmlencode (*in module archivebox.misc.util*), 136
hw_in_docker (*archivebox.api.v1_machine.MachineFilterSchema attribute*), 196
hw_in_docker (*archivebox.api.v1_machine.MachineSchema attribute*), 195
hw_in_docker (*archivebox.machine.models.Machine attribute*), 96
hw_in_vm (*archivebox.api.v1_machine.MachineFilterSchema attribute*), 196
hw_in_vm (*archivebox.api.v1_machine.MachineSchema attribute*), 195
hw_in_vm (*archivebox.machine.models.Machine attribute*), 96
hw_manufacturer (*archivebox.api.v1_machine.MachineSchema attribute*), 195
hw_manufacturer (*archivebox.machine.models.Machine attribute*), 96
hw_product (*archivebox.api.v1_machine.MachineSchema attribute*), 196
hw_product (*archivebox.machine.models.Machine attribute*), 97
hw_uuid (*archivebox.api.v1_machine.MachineSchema attribute*), 196
hw_uuid (*archivebox.machine.models.Machine attribute*), 97
hydrate_archiveresult_output_metadata() (*archivebox.core.models.Snapshot method*), 266

hydrate_binary_from_context() (*archivebox.machine.models.Process method*), 106
hydrate_visible_snapshot_counts() (*archivebox.crawls.admin.CrawlAdmin method*), 318

I

icons() (*archivebox.core.models.Snapshot method*), 268
id (*archivebox.api.models.APIToken attribute*), 190
id (*archivebox.api.models.OutboundWebhook attribute*), 191
id (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
id (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
id (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 184
id (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
id (*archivebox.api.v1_core.TagSchema attribute*), 186
id (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
id (*archivebox.api.v1_core.TagSearchSnapshotSchema attribute*), 187
id (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
id (*archivebox.api.v1_machine.BinaryFilterSchema attribute*), 197
id (*archivebox.api.v1_machine.BinarySchema attribute*), 197
id (*archivebox.api.v1_machine.MachineFilterSchema attribute*), 196
id (*archivebox.api.v1_machine.MachineSchema attribute*), 195
id (*archivebox.api.v1_personas.PersonaSchema attribute*), 174
id (*archivebox.base_models.models.ModelWithUUID attribute*), 350
id (*archivebox.core.models.ArchiveResult attribute*), 273
id (*archivebox.core.models.Snapshot attribute*), 262
id (*archivebox.core.models.SnapshotTag attribute*), 260
id (*archivebox.core.models.Tag attribute*), 260
id (*archivebox.crawls.models.Crawl attribute*), 311
id (*archivebox.crawls.models.CrawlSchedule attribute*), 309
id (*archivebox.machine.models.Binary attribute*), 99
id (*archivebox.machine.models.Machine attribute*), 96
id (*archivebox.machine.models.NetworkInterface attribute*), 98
id (*archivebox.machine.models.Process attribute*), 104
id (*archivebox.personas.models.Persona attribute*), 305
id_display() (*archivebox.machine.admin.MachineAdmin method*), 113
iface (*archivebox.core.models.ArchiveResult property*), 276
iface (*archivebox.machine.models.NetworkInterface attribute*), 98
iface (*archivebox.machine.models.Process attribute*), 105
iface_link() (*archivebox.machine.admin.ProcessAdmin method*), 115
IGNORABLE_URL_PATTERNS (*in module archivebox.core.settings_logging*), 237
IMG_SRC_ATTR_RE (*in module archivebox.misc.serve_static*), 128
import_artifact_status() (*archivebox.personas.admin.PersonaAdmin method*), 308
import_capture_storage (*archivebox.personas.forms.PersonaAdminForm attribute*), 303
import_copy_profile (*archivebox.personas.forms.PersonaAdminForm attribute*), 303
import_discovered_profile (*archivebox.personas.forms.PersonaAdminForm attribute*), 303
import_extract_cookies (*archivebox.personas.forms.PersonaAdminForm attribute*), 303
import_mode (*archivebox.personas.forms.PersonaAdminForm attribute*), 303
import_path (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 201
import_persona_from_source() (*in module archivebox.personas.importers*), 302
import_profile_name (*archivebox.personas.forms.PersonaAdminForm attribute*), 303
import_source (*archivebox.personas.forms.PersonaAdminForm attribute*), 303
imported_timestamp() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
IN_DOCKER (*archivebox.config.common.ShellConfig attribute*), 339
IN_DOCKER (*archivebox.config.constants.ConstantsDict attribute*), 326
IN_DOCKER (*in module archivebox.config.paths*), 334
IN_DOCKER (*in module archivebox.config.permissions*), 325
IN_DOCKER (*in module archivebox.config.version*), 348
IN_QEMU (*archivebox.config.common.ShellConfig at-*

tribute), 339
INACTIVE_STATES (*archivebox.crawls.models.Crawl attribute*), 312
increment_health_stats() (*archivebox.base_models.models.ModelWithHealthStats method*), 351
index() (*archivebox.core.admin_site.ArchiveBoxAdmin method*), 254
index() (*in module archivebox.core.templatetags.core_tags*), 219
index_only (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
index_only (*archivebox.api.v1_cli.UpdateCommandSchema attribute*), 201
INDEX_ONLY (*archivebox.config.common.ArchivingConfig attribute*), 343
index_title (*archivebox.core.admin_site.ArchiveBoxAdmin attribute*), 254
indexes (*archivebox.core.models.ArchiveResult.Meta attribute*), 274
indexes (*archivebox.core.models.Snapshot.Meta attribute*), 264
indexes (*archivebox.crawls.models.Crawl.Meta attribute*), 312
indexes (*archivebox.machine.models.Process.Meta attribute*), 105
inherited_only_new() (*archivebox.crawls.admin.CrawlAdminForm static method*), 317
IniConfigSettingsSource (*class in archivebox.config.configset*), 332
init() (*in module archivebox.cli.archivebox_init*), 153
INITIAL_STATE (*archivebox.core.models.ArchiveResult attribute*), 272
INITIAL_STATE (*archivebox.core.models.Snapshot attribute*), 263
INITIAL_STATE (*archivebox.crawls.models.Crawl attribute*), 312
INITIAL_STATE (*archivebox.machine.models.Binary attribute*), 100
INITIAL_STATE (*archivebox.workers.models.ModelWithQueue attribute*), 213
inlines (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
InlineTagEditorWidget (*class in archivebox.core.widgets*), 227
install() (*archivebox.machine.models.Binary method*), 102
install() (*in module archivebox.cli.archivebox_install*), 159
install_claimed() (*archivebox.machine.models.Binary method*), 102
INSTALLED (*archivebox.machine.models.Binary.StatusChoices attribute*), 99
INSTALLED_APPS (*in module archivebox.core.settings*), 230
INSTALLED_BINARIES_BASE_URL (*in module archivebox.config.views*), 330
intcomma() (*in module archivebox.core.templatetags.core_tags*), 219
ip_addrs (*in module archivebox.machine.detect*), 94
ip_local (*archivebox.machine.models.NetworkInterface attribute*), 98
ip_public (*archivebox.machine.models.NetworkInterface attribute*), 98
ips() (*archivebox.machine.admin.MachineAdmin method*), 113
IPYTHON_ARGUMENTS (*in module archivebox.core.settings*), 232
IPYTHON_KERNEL_DISPLAY_NAME (*in module archivebox.core.settings*), 232
is_admin_user() (*in module archivebox.core.permissions*), 239
is_alive() (*archivebox.machine.models.Process method*), 108
is_allowed_archivebox_redirect_url() (*in module archivebox.core.routes_util*), 257
is_archivebox_internal_url() (*archivebox.core.models.Snapshot class method*), 265
is_archivebox_source_root() (*in module archivebox.misc.checks*), 124
is_archived() (*archivebox.core.models.Snapshot method*), 270
is_background_hook() (*in module archivebox.plugins.hooks*), 377
IS_DANGEROUS_MODE (*archivebox.config.common.ServerConfig property*), 341
is_due() (*archivebox.crawls.models.CrawlSchedule method*), 310
is_enabled (*archivebox.crawls.models.CrawlSchedule attribute*), 310
is_finished() (*archivebox.crawls.models.Crawl method*), 314
IS_GETTING_VERSION_OR_HELP (*in module archivebox.core.settings*), 229
IS_INSIDE_VENV (*archivebox.config.constants.ConstantsDict attribute*), 327
is_legacy_archive_dir() (*archivebox.core.models.Snapshot static method*), 266
IS_LOWER_SECURITY_MODE (*archivebox.config.common.ServerConfig property*), 341

IS_MIGRATING (*in module archivebox.core.settings*), 229
is_paused (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
is_paused (*archivebox.core.models.ArchiveResult property*), 275
is_paused (*archivebox.workers.models.ModelWithQueue property*), 214
is_port_in_use() (*in module archivebox.workers.supervisord_util*), 209
is_postgres() (*in module archivebox.misc.db*), 125
is_redacted_env_key() (*in module archivebox.machine.env_util*), 112
IS_ROOT (*archivebox.config.constants.ConstantsDict attribute*), 326
IS_ROOT (*in module archivebox.config.permissions*), 325
is_root_identity() (*in module archivebox.config.permissions*), 324
is_running (*archivebox.machine.models.Process property*), 108
is_sensitive_config_key() (*in module archivebox.config.common*), 338
IS_SHELL (*in module archivebox.core.settings*), 229
is_snapshot_subdomain() (*in module archivebox.core.routes_util*), 257
is_static() (*archivebox.core.models.Snapshot method*), 270
is_superuser() (*in module archivebox.config.views*), 330
IS_TESTING (*in module archivebox.core.settings*), 229
IS_TTY (*archivebox.config.common.ShellConfig attribute*), 339
IS_TTY (*in module archivebox.misc.logging*), 121
IS_UNSAFE_MODE (*archivebox.config.common.ServerConfig property*), 341
is_valid (*archivebox.api.v1_machine.BinarySchema attribute*), 197
is_valid (*archivebox.machine.models.Binary property*), 101
is_valid() (*archivebox.api.models.APIToken method*), 191
isp (*archivebox.machine.models.NetworkInterface attribute*), 98
items (*archivebox.api.v1_core.CustomPagination.Output attribute*), 178
items() (*archivebox.config.configset.BaseConfigSet method*), 333
items() (*archivebox.plugins.discovery.ConfigLookup method*), 373
iter_filtered_search_result_ids() (*in module archivebox.search.views*), 290
iter_meta_search_ids() (*in module archivebox.search.views*), 290
iter_plugin_dirs() (*in module archivebox.plugins.discovery*), 373
iter_query_search_ids() (*in module archivebox.search.query*), 292
iter_search_result_ids() (*in module archivebox.search.views*), 291
iter_snapshot_json() (*in module archivebox.cli.archivebox_snapshot*), 160
iter_url_prefix_search_ids() (*in module archivebox.search.views*), 290
iter_url_search_prefixes() (*in module archivebox.search.views*), 290

J

js (*archivebox.base_models.admin.KeyValueWidget.Media attribute*), 354
js (*archivebox.core.widgets.TagEditorWidget.Media attribute*), 226
js (*archivebox.crawls.admin.CrawlAdmin.Media attribute*), 318
JSON_INDEX_FILENAME (*archivebox.config.constants.ConstantsDict attribute*), 328
JSON_TOKEN_RE (*in module archivebox.plugins.views*), 376
JSONL_INDEX_FILENAME (*archivebox.config.constants.ConstantsDict attribute*), 328
JSONType (*in module archivebox.api.v1_cli*), 199

K

keys() (*archivebox.config.configset.BaseConfigSet method*), 333
KeyValueWidget (*class in archivebox.base_models.admin*), 353
KeyValueWidget.Media (*class in archivebox.base_models.admin*), 353
kill() (*archivebox.machine.models.Process method*), 109
kill_children_db() (*archivebox.machine.models.Process method*), 110
kill_process() (*archivebox.machine.admin.ProcessAdmin method*), 115
kill_processes() (*archivebox.machine.admin.ProcessAdmin method*), 115
kill_remaining_processes() (*in module archivebox.core.shutdown_util*), 250
kill_tree() (*archivebox.machine.models.Process method*), 110
kind (*archivebox.crawls.models.CrawlSchedule property*), 310
kind (*archivebox.personas.importers.PersonaImportSource attribute*), 300

L

label (*archivebox.api.apps.APIConfig attribute*), 168
label (*archivebox.api.v1_crawls.CrawlCreateSchema attribute*), 170
label (*archivebox.core.apps.CoreConfig attribute*), 236
label (*archivebox.crawls.apps.CrawlsConfig attribute*), 309
label (*archivebox.crawls.models.Crawl attribute*), 311
label (*archivebox.crawls.models.CrawlSchedule attribute*), 310
label (*archivebox.machine.apps.MachineConfig attribute*), 93
label (*archivebox.personas.apps.PersonasConfig attribute*), 304
label (*archivebox.progressmonitor.apps.ProgressMonitorConfig attribute*), 116
label (*archivebox.workers.apps.WorkersConfig attribute*), 205
language (*archivebox.api.v1_personas.PersonaBrowserSettingsSchema attribute*), 174
LANGUAGE_CODE (*in module archivebox.core.settings*), 232
last_run_at (*archivebox.crawls.models.CrawlSchedule property*), 310
latest_outputs() (*archivebox.core.models.Snapshot method*), 270
launch() (*archivebox.machine.models.Process method*), 109
LDAP_BIND_DN (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_BIND_PASSWORD (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_CREATE_SUPERUSER (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_EMAIL_ATTR (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_ENABLED (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_FIRSTNAME_ATTR (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_LASTNAME_ATTR (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_SERVER_URI (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_USER_BASE (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_USER_FILTER (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAP_USERNAME_ATTR (*archivebox.config.ldap.LDAPConfig attribute*), 347
LDAPConfig (*class in archivebox.config.ldap*), 347
LDAPConfig (*class in archivebox.ldap.apps*), 118
legacy_archive_path() (*archivebox.core.models.Snapshot method*), 269
lifecycle_stop_reason() (*archivebox.crawls.models.Crawl method*), 314
limit (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 201
limit (*archivebox.api.v1_core.CustomPagination.Input attribute*), 177
limit (*archivebox.api.v1_core.CustomPagination.Output attribute*), 178
limit_config_for_crawl() (*archivebox.crawls.admin.CrawlAdmin method*), 319
limit_stop_reason() (*archivebox.crawls.models.Crawl method*), 314
list_archiveresults() (*in module archivebox.cli.archivebox_archiveresult*), 154
list_binaries() (*in module archivebox.cli.archivebox_binary*), 144
list_cmd() (*in module archivebox.cli.archivebox_archiveresult*), 154
list_cmd() (*in module archivebox.cli.archivebox_binary*), 145
list_cmd() (*in module archivebox.cli.archivebox_crawl*), 140
list_cmd() (*in module archivebox.cli.archivebox_machine*), 148
list_cmd() (*in module archivebox.cli.archivebox_persona*), 150
list_cmd() (*in module archivebox.cli.archivebox_process*), 157
list_cmd() (*in module archivebox.cli.archivebox_snapshot*), 161
list_cmd() (*in module archivebox.cli.archivebox_tag*), 165
list_crawls() (*in module archivebox.cli.archivebox_crawl*), 140
list_display (*archivebox.api.admin.APITokenAdmin attribute*), 203
list_display (*archivebox.api.admin.CustomWebhookAdmin attribute*), 204
list_display (*archivebox.base_models.admin.BaseModelAdmin attribute*), 354
list_display (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
list_display (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
list_display (*archivebox.core.admin_tags.TagAdmin attribute*), 252
list_display (*archivebox.core.admin_users.CustomUserAdmin attribute*), 285

list_display (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
list_display (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
list_display (*archivebox.machine.admin.BinaryAdmin attribute*), 114
list_display (*archivebox.machine.admin.MachineAdmin attribute*), 113
list_display (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 113
list_display (*archivebox.machine.admin.ProcessAdmin attribute*), 114
list_display (*archivebox.personas.admin.PersonaAdmin attribute*), 307
list_display_links (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
list_display_links (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
list_filter (*archivebox.api.admin.APITokenAdmin attribute*), 203
list_filter (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 225
list_filter (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
list_filter (*archivebox.core.admin_tags.TagAdmin attribute*), 252
list_filter (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
list_filter (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
list_filter (*archivebox.machine.admin.BinaryAdmin attribute*), 114
list_filter (*archivebox.machine.admin.MachineAdmin attribute*), 113
list_filter (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 113
list_filter (*archivebox.machine.admin.ProcessAdmin attribute*), 115
list_filter (*archivebox.personas.admin.PersonaAdmin attribute*), 307
list_machines() (*in module archivebox.cli.archivebox_machine*), 148
list_per_page (*archivebox.api.admin.APITokenAdmin attribute*), 203
list_per_page (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 225
list_per_page (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
list_per_page (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
list_per_page (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
list_per_page (*archivebox.machine.admin.BinaryAdmin attribute*), 114
list_per_page (*archivebox.machine.admin.MachineAdmin attribute*), 113
list_per_page (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 114
list_per_page (*archivebox.machine.admin.ProcessAdmin attribute*), 115
list_per_page (*archivebox.personas.admin.PersonaAdmin attribute*), 308
list_personas() (*in module archivebox.cli.archivebox_persona*), 149
list_processes() (*in module archivebox.cli.archivebox_process*), 157
list_select_related (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
list_select_related (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
list_select_related (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
list_snapshots() (*in module archivebox.cli.archivebox_snapshot*), 160
list_tags() (*in module archivebox.cli.archivebox_tag*), 165
ListCommandSchema (*class in archivebox.api.v1_cli*), 202
LISTENS_TO (*archivebox.services.archive_result_service.ArchiveResultServ attribute*), 359
LISTENS_TO (*archivebox.services.binary_service.ArchiveBoxBinaryService attribute*), 357
LISTENS_TO (*archivebox.services.crawl_service.CrawlService*

attribute), 367
LISTENS_TO (*archivebox.services.machine_service.MachineService attribute*), 366
LISTENS_TO (*archivebox.services.process_service.ProcessService attribute*), 356
LISTENS_TO (*archivebox.services.snapshot_service.SnapshotService attribute*), 365
LISTENS_TO (*archivebox.services.tag_service.TagService attribute*), 366
LIVE_CONFIG_BASE_URL (*in module archivebox.config.common*), 338
live_config_list_view() (*in module archivebox.core.views*), 248
live_config_value_view() (*in module archivebox.core.views*), 248
LIVE_PLUGIN_BASE_URL (*in module archivebox.plugins.views*), 375
live_progress_view() (*in module archivebox.progressmonitor.views*), 117
live_runner_processes() (*in module archivebox.core.takeover_util*), 241
load_deferred_fields_for_delete_webhook() (*in module archivebox.api.webhooks*), 172
load_from_directory() (*archivebox.core.models.Snapshot class method*), 267
load_from_file() (*archivebox.config.configset.BaseConfigSet class method*), 333
load_run_state() (*archivebox.services.runner.CrawlRunner method*), 363
load_snapshot_payload() (*archivebox.services.runner.CrawlRunner method*), 363
LOADED_PLUGINS (*in module archivebox.core.settings*), 229
lock_runtime_for_crawl() (*archivebox.personas.models.Persona method*), 306
LOG_DETAIL_TAIL_BYTES (*in module archivebox.config.views*), 330
log_detail_view() (*in module archivebox.config.views*), 331
LOG_FILE_NAME (*in module archivebox.workers.supervisord_util*), 207
LOG_LEVEL_DATABASE (*in module archivebox.core.settings_logging*), 238
LOG_LEVEL_REQUEST (*in module archivebox.core.settings_logging*), 238
log_list_finished() (*in module archivebox.misc.logging_util*), 122
log_list_started() (*in module archivebox.misc.logging_util*), 122
log_list_view() (*in module archivebox.config.views*), 331
log_removal_finished() (*in module archivebox.misc.logging_util*), 122
log_removal_started() (*in module archivebox.misc.logging_util*), 122
log_sqlite_lock_holders() (*in module archivebox.misc.db*), 126
log_worker_event() (*in module archivebox.misc.logging_util*), 123
logger (*in module archivebox.api.webhooks*), 172
logger (*in module archivebox.core.settings*), 229
logger (*in module archivebox.workers.models*), 213
LOGGING (*in module archivebox.core.settings*), 232
login() (*archivebox.core.admin_site.ArchiveBoxAdmin method*), 254
LOGIN_URL (*in module archivebox.core.settings*), 229
LOGOUT_REDIRECT_URL (*archivebox.config.common.ServerConfig attribute*), 341
LOGOUT_REDIRECT_URL (*in module archivebox.core.settings*), 230
LOGS_DIR (*archivebox.config.constants.ConstantsDict attribute*), 328
LOGS_DIR (*in module archivebox.core.settings_logging*), 238
LOGS_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
lookup_allowed() (*archivebox.api.admin.CustomWebhookAdmin method*), 204
lookup_allowed() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
lookups() (*archivebox.core.admin_snapshots.SnapshotArchiveStateListFil method*), 281
lookups() (*archivebox.core.admin_snapshots.SnapshotPermissionsListFil method*), 280
lookups() (*archivebox.core.admin_snapshots.SnapshotResultHealthListFi method*), 281
lookups() (*archivebox.core.admin_snapshots.SnapshotSizeListFilter method*), 281
lookups() (*archivebox.core.admin_snapshots.SnapshotStatusListFilter method*), 281
lookups() (*archivebox.core.admin_snapshots.TagNameListFilter method*), 280
lookups() (*archivebox.crawls.admin.MaxDepthListFilter method*), 316

M

mac_address (*archivebox.machine.models.NetworkInterface attribute*), 98
mac_addrs (*in module archivebox.machine.detect*), 94

machine (*archivebox.core.models.ArchiveResult property*), 276
machine (*archivebox.machine.models.Binary attribute*), 100
machine (*archivebox.machine.models.NetworkInterface attribute*), 98
machine (*archivebox.machine.models.Process attribute*), 104
Machine (*class in archivebox.machine.models*), 96
Machine.Meta (*class in archivebox.machine.models*), 97
machine_hostname (*archivebox.api.v1_machine.BinarySchema attribute*), 197
machine_id (*archivebox.api.v1_machine.BinaryFilterSchema attribute*), 198
machine_id (*archivebox.api.v1_machine.BinarySchema attribute*), 197
MACHINE_ID (*archivebox.config.constants.ConstantsDict attribute*), 326
machine_id (*archivebox.machine.models.Binary attribute*), 100
machine_id (*archivebox.machine.models.NetworkInterface attribute*), 99
machine_id (*archivebox.machine.models.Process attribute*), 105
machine_info() (*archivebox.machine.admin.BinaryAdmin method*), 114
machine_info() (*archivebox.machine.admin.NetworkInterfaceAdmin method*), 114
machine_info() (*archivebox.machine.admin.ProcessAdmin method*), 115
machine_link() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 226
MACHINE_RECHECK_INTERVAL (*in module archivebox.machine.models*), 95
MACHINE_TYPE (*archivebox.config.constants.ConstantsDict attribute*), 326
MachineAdmin (*class in archivebox.machine.admin*), 112
MachineConfig (*class in archivebox.machine.apps*), 93
MachineFilterSchema (*class in archivebox.api.v1_machine*), 196
MachineManager (*class in archivebox.machine.models*), 96
MachineSchema (*class in archivebox.api.v1_machine*), 195
MachineService (*class in archivebox.services.machine_service*), 366
main() (*in module archivebox.cli*), 167
main() (*in module archivebox.cli.archivebox_add*), 163
main() (*in module archivebox.cli.archivebox_archiveresult*), 154
main() (*in module archivebox.cli.archivebox_binary*), 145
main() (*in module archivebox.cli.archivebox_config*), 152
main() (*in module archivebox.cli.archivebox_crawl*), 140
main() (*in module archivebox.cli.archivebox_extract*), 156
main() (*in module archivebox.cli.archivebox_help*), 143
main() (*in module archivebox.cli.archivebox_init*), 153
main() (*in module archivebox.cli.archivebox_install*), 159
main() (*in module archivebox.cli.archivebox_list*), 166
main() (*in module archivebox.cli.archivebox_machine*), 148
main() (*in module archivebox.cli.archivebox_manage*), 157
main() (*in module archivebox.cli.archivebox_mcp*), 151
main() (*in module archivebox.cli.archivebox_oneshot*), 152
main() (*in module archivebox.cli.archivebox_persona*), 150
main() (*in module archivebox.cli.archivebox_pluginmap*), 163
main() (*in module archivebox.cli.archivebox_process*), 157
main() (*in module archivebox.cli.archivebox_remove*), 139
main() (*in module archivebox.cli.archivebox_run*), 143
main() (*in module archivebox.cli.archivebox_schedule*), 141
main() (*in module archivebox.cli.archivebox_server*), 162
main() (*in module archivebox.cli.archivebox_shell*), 148
main() (*in module archivebox.cli.archivebox_snapshot*), 161
main() (*in module archivebox.cli.archivebox_status*), 158
main() (*in module archivebox.cli.archivebox_tag*), 165
main() (*in module archivebox.cli.archivebox_update*), 147
main() (*in module archivebox.cli.archivebox_version*), 151
main() (*in module archivebox.workers.supervisord_parent_watchdog*), 212
main_plugins (*archivebox.core.forms.AddLinkForm attribute*), 234
manage() (*in module archivebox.cli.archivebox_manage*), 157
mark_archiveresult_started() (*in module archivebox.services.archive_result_service*), 359
mark_exited() (*archivebox.machine.models.Process method*), 106

mark_running() (*archivebox.machine.models.Process method*), 106
mark_started() (*archivebox.crawls.models.Crawl method*), 315
MARKDOWN_BOLD_RE (*in module archivebox.misc.serve_static*), 130
MARKDOWN_INLINE_IMAGE_RE (*in module archivebox.misc.serve_static*), 130
MARKDOWN_INLINE_LINK_RE (*in module archivebox.misc.serve_static*), 130
MARKDOWN_ITALIC_RE (*in module archivebox.misc.serve_static*), 130
matching_process() (*in module archivebox.workers.supervisord_parent_watchdog*), 212
max_depth (*archivebox.api.v1_crawls.CrawlCreateSchema attribute*), 170
max_depth (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
max_depth (*archivebox.crawls.models.Crawl attribute*), 311
MAX_HOOK_RUNTIME_SECONDS (*archivebox.config.constants.ConstantsDict attribute*), 329
max_num (*archivebox.core.admin_tags.TagInline attribute*), 251
MAX_SEARCH_RANK_IDS (*in module archivebox.search.query*), 291
MAX_SNAPSHOT_RUNTIME_SECONDS (*archivebox.config.constants.ConstantsDict attribute*), 329
MAX_STATUS_FS_DIR_SCAN (*in module archivebox.cli.archivebox_status*), 158
MAX_TMP_SOCKET_URL_LENGTH (*in module archivebox.config.paths*), 334
MAX_URL_LENGTH (*in module archivebox.misc.util*), 137
max_urls (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
max_urls (*archivebox.core.forms.AddLinkForm attribute*), 234
MaxDepthListFilter (*class in archivebox.crawls.admin*), 316
maximum (*archivebox.base_models.admin.ConfigOption attribute*), 353
mcp() (*in module archivebox.cli.archivebox_mcp*), 151
MCPConfig (*class in archivebox.mcp.apps*), 294
MCPJSONEncoder (*class in archivebox.mcp.server*), 296
MCPServer (*class in archivebox.mcp.server*), 297
MCPTool (*class in archivebox.mcp.server*), 296
media_plugins (*archivebox.core.forms.AddLinkForm attribute*), 234
MEMORY_FULL_STALL_PERCENT (*in module archivebox.services.resource_admission*), 360
memory_headroom() (*archivebox.services.resource_admission.ResourceAdmission method*), 360
memory_pressure() (*archivebox.services.resource_admission.ResourceAdmission method*), 361
meta_commands (*archivebox.cli.ArchiveBoxGroup attribute*), 167
metadata_plugins (*archivebox.core.forms.AddLinkForm attribute*), 234
MIDDLEWARE (*in module archivebox.core.settings*), 230
migrate_filesystem_to_current_version() (*archivebox.core.models.Snapshot method*), 266
migrate_only (*archivebox.api.v1_cli.UpdateCommandSchema attribute*), 201
migration_lock() (*in module archivebox.misc.db*), 126
MIGRATION_MODULES (*in module archivebox.core.settings*), 230
migration_state() (*in module archivebox.misc.db*), 126
MIN_DEPENDENCY_INSTALL_AVAILABLE_MEMORY_BYTES (*in module archivebox.workers.supervisord_util*), 208
MIN_SERVER_WORKER_AVAILABLE_MEMORY_BYTES (*in module archivebox.workers.supervisord_util*), 208
MinimalArchiveResultSchema (*class in archivebox.api.v1_core*), 178
minimum (*archivebox.base_models.admin.ConfigOption attribute*), 353
mirror_machine_config_to_file() (*in module archivebox.config.collection*), 322
missing_delete_at_candidates() (*archivebox.base_models.models.ModelWithDeleteAfter class method*), 352
missing_delete_at_candidates() (*archivebox.core.models.ArchiveResult class method*), 274
missing_delete_at_candidates() (*archivebox.core.models.Snapshot class method*), 265
missing_delete_at_candidates() (*archivebox.crawls.models.Crawl class method*), 313
missing_delete_at_candidates() (*archivebox.machine.models.Process class method*), 106
model (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 223
model (*archivebox.core.admin_snapshots.SnapshotAdminForm.Meta attribute*), 282
model (*archivebox.core.admin_tags.TagAdminForm.Meta*

attribute), 251
model (*archivebox.core.admin_tags.TagInline attribute*), 251
model (*archivebox.core.views.PublicIndexView attribute*), 247
model (*archivebox.crawls.admin.CrawlAdminForm.Meta attribute*), 317
model (*archivebox.personas.forms.PersonaAdminForm.Meta attribute*), 303
model_commands (*archivebox.cli.ArchiveBoxGroup attribute*), 167
model_config (*archivebox.config.common.ArchiveBoxBaseConfig attribute*), 345
model_config (*archivebox.config.common.ArchiveBoxSourceSettings attribute*), 346
model_config (*archivebox.config.configset.BaseConfigSet attribute*), 332
ModelWithConfig (*class in archivebox.base_models.models*), 351
ModelWithConfig.Meta (*class in archivebox.base_models.models*), 351
ModelWithDeleteAfter (*class in archivebox.base_models.models*), 351
ModelWithDeleteAfter.Meta (*class in archivebox.base_models.models*), 351
ModelWithHealthStats (*class in archivebox.base_models.models*), 351
ModelWithHealthStats.Meta (*class in archivebox.base_models.models*), 351
ModelWithNotes (*class in archivebox.base_models.models*), 350
ModelWithNotes.Meta (*class in archivebox.base_models.models*), 350
ModelWithOutputDir (*class in archivebox.base_models.models*), 352
ModelWithOutputDir.Meta (*class in archivebox.base_models.models*), 352
ModelWithQueue (*class in archivebox.workers.models*), 213
ModelWithQueue.Meta (*class in archivebox.workers.models*), 214
ModelWithUUID (*class in archivebox.base_models.models*), 350
ModelWithUUID.Meta (*class in archivebox.base_models.models*), 350
modified_at (*archivebox.api.models.APIToken attribute*), 191
modified_at (*archivebox.api.models.OutboundWebhook attribute*), 191
modified_at (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
modified_at (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
modified_at (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
modified_at (*archivebox.api.v1_core.TagSchema attribute*), 186
modified_at (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
modified_at (*archivebox.api.v1_machine.BinarySchema attribute*), 197
modified_at (*archivebox.api.v1_machine.MachineSchema attribute*), 195
modified_at (*archivebox.base_models.models.ModelWithUUID attribute*), 350
modified_at (*archivebox.core.models.ArchiveResult attribute*), 273
modified_at (*archivebox.core.models.Snapshot attribute*), 262
modified_at (*archivebox.core.models.Tag attribute*), 260
modified_at (*archivebox.crawls.models.Crawl attribute*), 311
modified_at (*archivebox.crawls.models.CrawlSchedule attribute*), 310
modified_at (*archivebox.machine.models.Binary attribute*), 100
modified_at (*archivebox.machine.models.Machine attribute*), 96
modified_at (*archivebox.machine.models.NetworkInterface attribute*), 98
modified_at (*archivebox.machine.models.Process attribute*), 104
modified_at__gte (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
modified_at__lt (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
ModifiedAccessLogGenerator (*class in archivebox.misc.monkey_patches*), 134
module
    archivebox, 92
    archivebox.__main__, 378
    archivebox.api, 167
    archivebox.api.admin, 203
    archivebox.api.apps, 167
    archivebox.api.auth, 193

archivebox.api.middleware, 193
archivebox.api.models, 190
archivebox.api.urls, 171
archivebox.api.v1_api, 172
archivebox.api.v1_auth, 192
archivebox.api.v1_cli, 198
archivebox.api.v1_core, 175
archivebox.api.v1_crawls, 168
archivebox.api.v1_machine, 194
archivebox.api.v1_personas, 173
archivebox.api.webhooks, 171
archivebox.base_models, 349
archivebox.base_models.admin, 352
archivebox.base_models.apps, 349
archivebox.base_models.models, 349
archivebox.cli, 138
archivebox.cli.archivebox_add, 162
archivebox.cli.archivebox_archiveresult, 153
archivebox.cli.archivebox_binary, 144
archivebox.cli.archivebox_config, 152
archivebox.cli.archivebox_crawl, 139
archivebox.cli.archivebox_extract, 155
archivebox.cli.archivebox_help, 143
archivebox.cli.archivebox_init, 152
archivebox.cli.archivebox_install, 158
archivebox.cli.archivebox_list, 166
archivebox.cli.archivebox_machine, 147
archivebox.cli.archivebox_manage, 156
archivebox.cli.archivebox_mcp, 150
archivebox.cli.archivebox_oneshot, 152
archivebox.cli.archivebox_persona, 148
archivebox.cli.archivebox_pluginmap, 163
archivebox.cli.archivebox_process, 157
archivebox.cli.archivebox_remove, 138
archivebox.cli.archivebox_run, 141
archivebox.cli.archivebox_schedule, 141
archivebox.cli.archivebox_search, 141
archivebox.cli.archivebox_server, 161
archivebox.cli.archivebox_shell, 148
archivebox.cli.archivebox_snapshot, 159
archivebox.cli.archivebox_status, 158
archivebox.cli.archivebox_tag, 164
archivebox.cli.archivebox_update, 145
archivebox.cli.archivebox_version, 151
archivebox.cli.cli_util, 164
archivebox.cli.rescan, 156
archivebox.config, 321
archivebox.config.collection, 321
archivebox.config.common, 336
archivebox.config.configset, 331
archivebox.config.constants, 326
archivebox.config.django, 335
archivebox.config.ldap, 347
archivebox.config.paths, 333
archivebox.config.permissions, 323
archivebox.config.version, 348
archivebox.config.views, 329
archivebox.core, 215
archivebox.core.admin, 287
archivebox.core.admin_archiveresults, 223
archivebox.core.admin_site, 253
archivebox.core.admin_snapshots, 279
archivebox.core.admin_tags, 251
archivebox.core.admin_users, 285
archivebox.core.apps, 236
archivebox.core.asgi, 286
archivebox.core.context_processors, 242
archivebox.core.forms, 233
archivebox.core.management, 221
archivebox.core.management.commands, 221
archivebox.core.management.commands.archivebox, 221
archivebox.core.middleware, 276
archivebox.core.models, 259
archivebox.core.permissions, 238
archivebox.core.recovery_util, 242
archivebox.core.routes_util, 254
archivebox.core.settings, 227
archivebox.core.settings_logging, 236
archivebox.core.setup_wizard, 253
archivebox.core.shutdown_util, 249
archivebox.core.snapshot_status, 236
archivebox.core.sqlite_backend, 221
archivebox.core.sqlite_backend.base, 221
archivebox.core.tag_util, 277
archivebox.core.takeover_util, 240
archivebox.core.templatetags, 215
archivebox.core.templatetags.core_tags, 215
archivebox.core.urls, 248
archivebox.core.views, 242
archivebox.core.widgets, 226
archivebox.core.wsgi, 285
archivebox.crawls, 308
archivebox.crawls.admin, 316
archivebox.crawls.apps, 308
archivebox.crawls.locks, 315
archivebox.crawls.models, 309
archivebox.crawls.schedule_util, 309
archivebox.ldap, 117
archivebox.ldap.apps, 118
archivebox.ldap.auth, 118
archivebox.machine, 92
archivebox.machine.admin, 112
archivebox.machine.apps, 92
archivebox.machine.detect, 93
archivebox.machine.env_util, 111

archivebox.machine.models, 94
archivebox.manage, 378
archivebox.mcp, 294
archivebox.mcp.apps, 294
archivebox.mcp.server, 295
archivebox.misc, 119
archivebox.misc.checks, 123
archivebox.misc.db, 124
archivebox.misc.jsonl, 132
archivebox.misc.logging, 120
archivebox.misc.logging_util, 121
archivebox.misc.monkey_patches, 134
archivebox.misc.paginators, 119
archivebox.misc.serve_static, 127
archivebox.misc.shell_welcome_message, 120
archivebox.misc.system, 119
archivebox.misc.toml_util, 119
archivebox.misc.util, 134
archivebox.personas, 298
archivebox.personas.admin, 307
archivebox.personas.apps, 304
archivebox.personas.forms, 303
archivebox.personas.importers, 298
archivebox.personas.models, 304
archivebox.plugins, 367
archivebox.plugins.apps, 370
archivebox.plugins.discovery, 371
archivebox.plugins.forms, 367
archivebox.plugins.hooks, 376
archivebox.plugins.output_groups, 370
archivebox.plugins.views, 375
archivebox.progressmonitor, 116
archivebox.progressmonitor.apps, 116
archivebox.progressmonitor.collection, 116
archivebox.progressmonitor.views, 117
archivebox.search, 288
archivebox.search.admin, 293
archivebox.search.apps, 288
archivebox.search.backends, 288
archivebox.search.config, 292
archivebox.search.query, 291
archivebox.search.sonic_daemon, 288
archivebox.search.views, 289
archivebox.services, 355
archivebox.services.archive_result_service, 358
archivebox.services.binary_service, 356
archivebox.services.crawl_service, 367
archivebox.services.machine_service, 366
archivebox.services.process_service, 355
archivebox.services.resource_admission, 359
archivebox.services.runner, 361
archivebox.services.snapshot_service, 364
archivebox.services.tag_service, 365
archivebox.uuid_compat, 377
archivebox.workers, 204
archivebox.workers.admin, 214
archivebox.workers.apps, 205
archivebox.workers.management, 204
archivebox.workers.management.commands, 204
archivebox.workers.management.commands.runner_watch, 204
archivebox.workers.management.commands.supervisord_wat 205
archivebox.workers.models, 212
archivebox.workers.supervisord_parent_watchdog, 212
archivebox.workers.supervisord_util, 206
MODULE_PATH (*in module archivebox.workers.models*), 213
move_directory_to_invalid() (*archivebox.core.models.Snapshot static method*), 268

N

name (*archivebox.api.apps.APIConfig attribute*), 168
name (*archivebox.api.v1_core.TagCreateSchema attribute*), 187
name (*archivebox.api.v1_core.TagSchema attribute*), 186
name (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
name (*archivebox.api.v1_core.TagUpdateSchema attribute*), 189
name (*archivebox.api.v1_machine.BinaryFilterSchema attribute*), 198
name (*archivebox.api.v1_machine.BinarySchema attribute*), 197
name (*archivebox.api.v1_personas.PersonaSchema attribute*), 174
name (*archivebox.api.v1_personas.PersonaSyncSchema attribute*), 174
name (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 223
name (*archivebox.core.apps.CoreConfig attribute*), 236
name (*archivebox.core.models.Tag attribute*), 260
name (*archivebox.crawls.apps.CrawlsConfig attribute*), 308
name (*archivebox.ldap.apps.LDAPConfig attribute*), 118
name (*archivebox.machine.apps.MachineConfig attribute*), 93
name (*archivebox.machine.models.Binary attribute*), 100
name (*archivebox.mcp.apps.MCPConfig attribute*), 294
name (*archivebox.mcp.server.MCPTool attribute*), 296

name (*archivebox.personas.apps.PersonasConfig attribute*), 304
name (*archivebox.personas.models.Persona attribute*), 305
name (*archivebox.plugins.apps.PluginsConfig attribute*), 370
name (*archivebox.progressmonitor.apps.ProgressMonitorConfig attribute*), 116
name (*archivebox.search.apps.SearchConfig attribute*), 288
name (*archivebox.workers.apps.WorkersConfig attribute*), 205
namespace (*archivebox.core.admin_site.ArchiveBoxAdmin attribute*), 254
NETSCAPE_COOKIE_HEADER (*in module archivebox.personas.importers*), 301
NETWORK_INTERFACE_RECHECK_INTERVAL (*in module archivebox.machine.models*), 95
NetworkInterface (*class in archivebox.machine.models*), 98
NetworkInterface.Meta (*class in archivebox.machine.models*), 99
networkinterface_set (*archivebox.machine.models.Machine attribute*), 97
NetworkInterfaceAdmin (*class in archivebox.machine.admin*), 113
NetworkInterfaceManager (*class in archivebox.machine.models*), 98
newest_archive_date() (*archivebox.core.models.Snapshot method*), 270
next_run_at (*archivebox.crawls.models.CrawlSchedule property*), 310
next_run_for_schedule() (*in module archivebox.crawls.schedule_util*), 309
NinjaAPIWithIOCapture (*class in archivebox.api.v1_api*), 173
NoisyRequestsFilter (*class in archivebox.core.settings_logging*), 237
NORESULTS (*archivebox.core.models.ArchiveResult.StatusChoices attribute*), 272
normalize_base_url() (*in module archivebox.core.routes_util*), 256
normalize_config_json_values() (*in module archivebox.base_models.models*), 350
normalize_created_by_filter() (*in module archivebox.core.tag_util*), 278
normalize_created_year_filter() (*in module archivebox.core.tag_util*), 278
normalize_domain() (*archivebox.crawls.models.Crawl static method*), 313
normalize_has_snapshots_filter() (*in module archivebox.core.tag_util*), 278
normalize_hook_event_name() (*in module archivebox.plugins.hooks*), 377
normalize_permissions() (*in module archivebox.core.permissions*), 239
normalize_process_env() (*in module archivebox.services.process_service*), 355
normalize_runtime_config() (*in module archivebox.config.common*), 339
normalize_schedule() (*in module archivebox.crawls.schedule_util*), 309
normalize_search_backend_name() (*in module archivebox.search.backends*), 289
normalize_search_result_id() (*in module archivebox.search.views*), 290
normalize_snapshot_status() (*in module archivebox.core.snapshot_status*), 236
normalize_status() (*archivebox.core.models.ArchiveResult class method*), 272
normalize_tag_list() (*in module archivebox.api.v1_core*), 184
normalize_tag_list() (*in module archivebox.api.v1_crawls*), 170
normalize_tag_name() (*in module archivebox.core.tag_util*), 278
normalize_tag_sort() (*in module archivebox.core.tag_util*), 278
notes (*archivebox.api.v1_crawls.CrawlCreateSchema attribute*), 170
notes (*archivebox.base_models.models.ModelWithNotes attribute*), 350
notes (*archivebox.core.forms.AddLinkForm attribute*), 234
notes (*archivebox.core.models.ArchiveResult attribute*), 274
notes (*archivebox.core.models.Snapshot attribute*), 263
notes (*archivebox.crawls.models.Crawl attribute*), 311
notes (*archivebox.crawls.models.CrawlSchedule attribute*), 310
NPM_RELATED_NAMES (*archivebox.config.constants.ConstantsDict attribute*), 329
num_archived_snapshots() (*archivebox.crawls.admin.CrawlAdmin method*), 319
num_archiveresults (*archivebox.api.v1_core.SnapshotSchema attribute*), 183
num_crawls() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
num_failures() (*archivebox.core.models.Snapshot method*), 270
num_items (*archivebox.api.v1_core.CustomPagination.Output attribute*), 178

num_outputs() (*archivebox.core.models.Snapshot method*), 270
num_snapshots (*archivebox.api.v1_core.TagSchema attribute*), 186
num_snapshots (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
num_snapshots() (*archivebox.core.admin_tags.TagAdmin method*), 252
num_snapshots() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
num_total_snapshots() (*archivebox.crawls.admin.CrawlAdmin method*), 319
num_uses_failed (*archivebox.api.v1_machine.BinarySchema attribute*), 197
num_uses_failed (*archivebox.api.v1_machine.MachineSchema attribute*), 196
num_uses_failed (*archivebox.base_models.models.ModelWithHealthStats attribute*), 351
num_uses_failed (*archivebox.machine.models.Binary attribute*), 100
num_uses_failed (*archivebox.machine.models.Machine attribute*), 97
num_uses_succeeded (*archivebox.api.v1_machine.BinarySchema attribute*), 197
num_uses_succeeded (*archivebox.api.v1_machine.MachineSchema attribute*), 196
num_uses_succeeded (*archivebox.base_models.models.ModelWithHealthStats attribute*), 351
num_uses_succeeded (*archivebox.machine.models.Binary attribute*), 100
num_uses_succeeded (*archivebox.machine.models.Machine attribute*), 97

O

obj_to_yaml() (*in module archivebox.config.views*), 330
objects (*archivebox.core.models.ArchiveResult attribute*), 271
objects (*archivebox.core.models.Snapshot attribute*), 264
objects (*archivebox.crawls.models.Crawl attribute*), 311
objects (*archivebox.machine.models.Binary attribute*), 101
objects (*archivebox.machine.models.Machine attribute*), 97
objects (*archivebox.machine.models.NetworkInterface attribute*), 98
objects (*archivebox.machine.models.Process attribute*), 105
observe_memory_stalls() (*archivebox.services.resource_admission.ResourceAdmission method*), 360
offset (*archivebox.api.v1_core.CustomPagination.Input attribute*), 177
offset (*archivebox.api.v1_core.CustomPagination.Output attribute*), 178
oldest_archive_date() (*archivebox.core.models.Snapshot method*), 270
on_ArchiveResultEvent__save_to_db() (*archivebox.services.archive_result_service.ArchiveResultService method*), 359
on_BinaryEvent__finalize_process() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
on_BinaryRequestEvent__project_process() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
on_BinaryRequestEvent__schedule_missing_finalize() (*archivebox.services.binary_service.ArchiveBoxBinaryService method*), 357
on_CrawlAbortEvent() (*archivebox.services.runner.CrawlRunner method*), 362
on_CrawlCleanupEvent__flush_completed() (*archivebox.services.process_service.ProcessService method*), 356
on_CrawlCleanupEvent__save_to_db() (*archivebox.services.crawl_service.CrawlService method*), 367
on_CrawlCompletedEvent__flush_completed() (*archivebox.services.process_service.ProcessService method*), 356
on_CrawlCompletedEvent__save_to_db() (*archivebox.services.crawl_service.CrawlService method*), 367
on_CrawlSetupEvent__save_to_db() (*archivebox.services.crawl_service.CrawlService method*), 367
on_CrawlStartEvent__save_to_db() (*archivebox.services.crawl_service.CrawlService method*), 367
on_MachineEvent__save_to_db() (*archivebox.services.machine_service.MachineService method*), 366
on_ProcessCompletedEvent__save_to_db()

(*archivebox.services.process_service.ProcessService method*), 356
on_ProcessStartedEvent__save_to_db() (*archivebox.services.process_service.ProcessService method*), 356
on_SnapshotCompletedEvent() (*archivebox.services.snapshot_service.SnapshotService method*), 365
on_SnapshotEvent() (*archivebox.services.snapshot_service.SnapshotService method*), 365
on_TagEvent__save_to_db() (*archivebox.services.tag_service.TagService method*), 366
only_new (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
only_new (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 202
ONLY_NEW (*archivebox.config.common.ArchivingConfig attribute*), 342
open_cmd() (*in module archivebox.cli.archivebox_persona*), 150
OPEN_STATES (*archivebox.core.models.Snapshot attribute*), 263
optionxform() (*archivebox.config.configset.CaseConfigParser method*), 332
ORCHESTRATOR (*archivebox.machine.models.Process.TypeChoices attribute*), 103
order_output_plugins() (*in module archivebox.plugins.output_groups*), 371
order_snapshot_outputs() (*in module archivebox.plugins.output_groups*), 371
ordering (*archivebox.api.admin.APITokenAdmin attribute*), 203
ordering (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 225
ordering (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
ordering (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
ordering (*archivebox.core.admin_tags.TagAdmin attribute*), 252
ordering (*archivebox.core.views.PublicIndexView attribute*), 247
ordering (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
ordering (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
ordering (*archivebox.machine.admin.BinaryAdmin attribute*), 114
ordering (*archivebox.machine.admin.MachineAdmin attribute*), 113
ordering (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 113
ordering (*archivebox.machine.admin.ProcessAdmin attribute*), 115
ordering (*archivebox.personas.admin.PersonaAdmin attribute*), 308
OriginalDomainHostView (*class in archivebox.core.views*), 246
OriginalDomainReplayView (*class in archivebox.core.views*), 246
os_arch (*archivebox.api.v1_machine.MachineFilterSchema attribute*), 196
os_arch (*archivebox.api.v1_machine.MachineSchema attribute*), 196
os_arch (*archivebox.machine.models.Machine attribute*), 97
os_family (*archivebox.api.v1_machine.MachineSchema attribute*), 196
os_family (*archivebox.machine.models.Machine attribute*), 97
os_kernel (*archivebox.api.v1_machine.MachineSchema attribute*), 196
os_kernel (*archivebox.machine.models.Machine attribute*), 97
os_platform (*archivebox.api.v1_machine.MachineFilterSchema attribute*), 196
os_platform (*archivebox.api.v1_machine.MachineSchema attribute*), 196
os_platform (*archivebox.machine.models.Machine attribute*), 97
os_release (*archivebox.api.v1_machine.MachineSchema attribute*), 196
os_release (*archivebox.machine.models.Machine attribute*), 97
other_plugins (*archivebox.core.forms.AddLinkForm attribute*), 234
OutboundWebhook (*class in archivebox.api.models*), 191
OutboundWebhook.Meta (*class in archivebox.api.models*), 191
outboundwebhook_set() (*archivebox.core.admin_users.CustomUserAdmin method*), 286
OutboundWebhookAdminForm (*class in archivebox.api.admin*), 204
output_card() (*in module archivebox.core.templatetags.core_tags*), 220
output_dir (*archivebox.base_models.models.ModelWithOutputDir property*), 352
output_dir (*archivebox.core.models.ArchiveResult property*), 276
output_dir (*archivebox.core.models.Snapshot property*), 269

output_dir (*archivebox.crawls.models.Crawl property*), 313
output_dir (*archivebox.machine.models.Binary property*), 101
output_dir_name (*archivebox.base_models.models.ModelWithOutputDir property*), 352
output_dir_name (*archivebox.core.models.ArchiveResult property*), 275
output_dir_name (*archivebox.core.models.Snapshot property*), 268
output_dir_parent (*archivebox.base_models.models.ModelWithOutputDir property*), 352
output_dir_parent (*archivebox.core.models.ArchiveResult property*), 275
output_dir_parent (*archivebox.core.models.Snapshot property*), 268
output_dir_str (*archivebox.base_models.models.ModelWithOutputDir property*), 352
output_display() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 226
output_file_map() (*archivebox.core.models.ArchiveResult method*), 275
output_file_path() (*archivebox.core.models.ArchiveResult method*), 275
output_file_paths() (*archivebox.core.models.ArchiveResult method*), 275
output_files (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
output_files (*archivebox.core.models.ArchiveResult attribute*), 273
output_files_upload_complete() (*archivebox.core.models.ArchiveResult static method*), 272
output_folder_path() (*archivebox.core.models.ArchiveResult method*), 275
output_group_for_plugin() (*in module archivebox.plugins.output_groups*), 371
OUTPUT_GROUPS (*in module archivebox.plugins.output_groups*), 371
output_json (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
output_json (*archivebox.core.models.ArchiveResult attribute*), 273
output_mimetypes (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
output_mimetypes (*archivebox.core.models.ArchiveResult attribute*), 273
output_paths_for_delete() (*archivebox.base_models.models.ModelWithOutputDir method*), 352
output_paths_for_delete() (*archivebox.core.models.Snapshot method*), 264
OUTPUT_PERMISSIONS (*archivebox.config.common.StorageConfig attribute*), 340
output_size (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
output_size (*archivebox.api.v1_core.SnapshotSchema attribute*), 183
output_size (*archivebox.core.models.ArchiveResult attribute*), 273
output_size (*archivebox.core.models.Snapshot attribute*), 263
output_str (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
output_str (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
output_str (*archivebox.core.models.ArchiveResult attribute*), 273
output_str_display() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 226
output_str_for_display() (*archivebox.core.models.ArchiveResult method*), 274
output_summary() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 226
output_summary() (*archivebox.machine.admin.ProcessAdmin method*), 115
overrides (*archivebox.machine.models.Binary attribute*), 100
overwrite (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
overwrite (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 202
overwrite_snapshots() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 285
owner() (*archivebox.crawls.admin.CrawlAdmin method*), 319

P

PACKAGE_DIR (*archivebox.config.constants.ConstantsDict attribute*), 326
PACKAGE_DIR (*in module archivebox*), 379
PACKAGE_DIR (*in module archivebox.config.paths*), 334
PACKAGE_DIR (*in module archivebox.config.version*), 348
PACKAGE_DIR (*in module archivebox.core.settings*), 229
PACKAGE_DIR (*in module archivebox.machine.detect*), 94
PACKAGE_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
PACKAGE_ROOT (*in module archivebox.workers.models*), 213
page (*archivebox.api.v1_core.CustomPagination.Input attribute*), 177
page (*archivebox.api.v1_core.CustomPagination.Output attribute*), 178
page_setup_plugins (*archivebox.core.forms.AddLinkForm attribute*), 234
paged_iterator() (*archivebox.base_models.models.PagedQuerySet method*), 349
PagedQuerySet (*class in archivebox.base_models.models*), 349
paginate_queryset() (*archivebox.api.v1_core.CustomPagination method*), 178
paginate_queryset() (*archivebox.core.views.PublicIndexView method*), 247
paginator (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 225
paginator (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
paginator (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
paginator_class (*archivebox.core.views.PublicIndexView attribute*), 247
param_name (*archivebox.api.auth.HeaderTokenAuth attribute*), 194
param_name (*archivebox.api.auth.QueryParamTokenAuth attribute*), 194
parameter_name (*archivebox.core.admin_snapshots.SnapshotArchiveStateListFilter attribute*), 281
parameter_name (*archivebox.core.admin_snapshots.SnapshotPermissionsListFilter attribute*), 280
parameter_name (*archivebox.core.admin_snapshots.SnapshotResultHealthListFilter attribute*), 281
parameter_name (*archivebox.core.admin_snapshots.SnapshotSizeListFilter attribute*), 281
parameter_name (*archivebox.core.admin_snapshots.SnapshotStatusListFilter attribute*), 281
parameter_name (*archivebox.core.admin_snapshots.TagNameListFilter attribute*), 280
parameter_name (*archivebox.crawls.admin.MaxDepthListFilter attribute*), 316
parens_are_matched() (*in module archivebox.misc.util*), 137
parent (*archivebox.machine.models.Process attribute*), 104
parent_id (*archivebox.machine.models.Process attribute*), 105
parent_model (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 223
parent_snapshot (*archivebox.core.models.Snapshot attribute*), 262
parent_snapshot_id (*archivebox.core.models.Snapshot attribute*), 263
parse_date() (*in module archivebox.misc.util*), 137
parse_delete_after() (*in module archivebox.config.common*), 344
parse_event_datetime() (*in module archivebox.services.process_service*), 355
parse_filesize_to_bytes() (*in module archivebox.misc.util*), 137
parse_line() (*in module archivebox.misc.jsonl*), 133
parse_range_header() (*in module archivebox.misc.serve_static*), 131
parse_records_from_text() (*archivebox.machine.models.Process class method*), 106
parse_structured_records() (*in module archivebox.mcp.server*), 297
parse_tag_names() (*archivebox.crawls.models.Crawl static method*), 313
parse_tag_string() (*in module archivebox.core.forms*), 233
parsed_host_and_port() (*archivebox.crawls.admin.CrawlAdminForm static method*), 317
parser (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
PARSER (*archivebox.config.common.ArchivingConfig attribute*), 342
password (*archivebox.api.v1_auth.PasswordAuthSchema attribute*), 192
PASSWORD_RESET_URL (*in module archive-*

box.core.settings), 230
PasswordAuthSchema (*class in archivebox.api.v1_auth*), 192
patch_archiveresult() (*in module archivebox.api.v1_core*), 182
patch_crawl() (*in module archivebox.api.v1_crawls*), 171
patch_snapshot() (*in module archivebox.api.v1_core*), 186
path (*archivebox.personas.models.Persona property*), 306
path (*in module archivebox.misc.util*), 136
path() (*archivebox.core.models.Snapshot method*), 270
pattern (*archivebox.base_models.admin.ConfigOption attribute*), 353
pause() (*archivebox.core.models.Snapshot method*), 265
pause() (*archivebox.crawls.models.Crawl method*), 312
pause() (*archivebox.workers.models.ModelWithQueue method*), 214
pause_resume_control() (*archivebox.crawls.admin.CrawlAdmin method*), 319
pause_selected_crawls() (*archivebox.crawls.admin.CrawlAdmin method*), 319
PAUSED (*archivebox.core.models.ArchiveResult.StatusChoices attribute*), 272
PAUSED (*archivebox.workers.models.DefaultStatusChoices attribute*), 213
pending_migrations() (*in module archivebox.misc.db*), 126
PERMISSIONS (*archivebox.config.common.ArchivingConfig attribute*), 343
permissions (*archivebox.core.forms.AddLinkForm attribute*), 235
permissions (*archivebox.core.models.Snapshot attribute*), 263
permissions (*archivebox.crawls.models.Crawl attribute*), 311
permissions (*archivebox.personas.forms.PersonaAdminForm attribute*), 303
permissions (*archivebox.personas.models.Persona attribute*), 305
permissions_badge() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
permissions_badge() (*archivebox.crawls.admin.CrawlAdmin method*), 319
PERMISSIONS_CHOICES (*in module archivebox.core.permissions*), 239
permissions_config (*archivebox.core.admin_snapshots.SnapshotAdminForm attribute*), 282
permissions_from_legacy_public_flags() (*in module archivebox.config.common*), 338
PERMISSIONS_META (*in module archivebox.core.permissions*), 239
PERMISSIONS_PRIVATE (*in module archivebox.core.permissions*), 239
PERMISSIONS_PUBLIC (*in module archivebox.core.permissions*), 239
PERMISSIONS_UNLISTED (*in module archivebox.core.permissions*), 239
PERMISSIONS_VALUES (*in module archivebox.core.permissions*), 239
persona (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
persona (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
persona (*archivebox.api.v1_personas.PersonaSyncResponseSchema attribute*), 175
persona (*archivebox.core.forms.AddLinkForm attribute*), 235
persona (*archivebox.crawls.models.Crawl attribute*), 311
Persona (*class in archivebox.personas.models*), 305
Persona.Meta (*class in archivebox.personas.models*), 305
persona_paths() (*archivebox.personas.admin.PersonaAdmin method*), 308
PERSONA_PROFILE_DIR_CANDIDATES (*in module archivebox.personas.importers*), 299
PersonaAdmin (*class in archivebox.personas.admin*), 307
PersonaAdminForm (*class in archivebox.personas.forms*), 303
PersonaAdminForm.Meta (*class in archivebox.personas.forms*), 303
PersonaBrowserSettingsSchema (*class in archivebox.api.v1_personas*), 173
PersonaImportResult (*class in archivebox.personas.importers*), 300
PersonaImportSource (*class in archivebox.personas.importers*), 300
PERSONAS_DIR (*archivebox.config.constants.ConstantsDict attribute*), 328
PERSONAS_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
PersonaSchema (*class in archivebox.api.v1_personas*), 174
PersonasConfig (*class in archivebox.personas.apps*), 304

PersonaSyncResponseSchema (*class in archivebox.api.v1_personas*), 175
PersonaSyncSchema (*class in archivebox.api.v1_personas*), 174
pick_default_profile_dir() (*in module archivebox.personas.importers*), 301
pid (*archivebox.machine.models.Process attribute*), 104
PID_FILE_NAME (*in module archivebox.workers.supervisord_util*), 207
PID_REUSE_WINDOW (*in module archivebox.machine.models*), 95
PIP_RELATED_NAMES (*archivebox.config.constants.ConstantsDict attribute*), 329
pipe_stderr() (*archivebox.machine.models.Process method*), 109
pipe_stdout() (*archivebox.machine.models.Process method*), 109
plugin (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
plugin (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
plugin (*archivebox.base_models.admin.ConfigOption attribute*), 353
plugin (*archivebox.core.models.ArchiveResult attribute*), 273
plugin (*archivebox.machine.models.Process property*), 106
plugin_card() (*in module archivebox.core.templatetags.core_tags*), 220
plugin_card_is_hidden() (*in module archivebox.plugins.output_groups*), 371
plugin_card_is_interactive() (*in module archivebox.plugins.discovery*), 373
PLUGIN_CONFIG_FIELD_PREFIX (*in module archivebox.plugins.forms*), 368
plugin_config_keys() (*archivebox.plugins.forms.PluginConfigFormMixin method*), 369
PLUGIN_CONFIG_SCHEMAS (*in module archivebox.config.common*), 345
plugin_detail_view() (*in module archivebox.plugins.views*), 376
plugin_full() (*in module archivebox.core.templatetags.core_tags*), 220
plugin_groups (*archivebox.plugins.forms.PluginConfigFormMixin attribute*), 369
PLUGIN_GROUPS (*in module archivebox.plugins.forms*), 368
plugin_has_custom_full_response() (*in module archivebox.plugins.discovery*), 374
plugin_icon() (*in module archivebox.core.templatetags.core_tags*), 220
plugin_icon_is_hidden() (*in module archivebox.plugins.output_groups*), 371
plugin_name() (*in module archivebox.core.templatetags.core_tags*), 220
plugin_output_path() (*archivebox.core.models.Snapshot method*), 268
plugin_output_sizes() (*in module archivebox.plugins.output_groups*), 371
PLUGIN_PRESENTATION (*in module archivebox.plugins.output_groups*), 371
plugin_ui() (*in module archivebox.core.templatetags.core_tags*), 217
plugin_with_icon() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 226
PluginConfigFormMixin (*class in archivebox.plugins.forms*), 369
pluginmap() (*in module archivebox.cli.archivebox_pluginmap*), 163
plugins (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
PLUGINS (*archivebox.config.common.ArchivingConfig attribute*), 342
plugins_list_view() (*in module archivebox.plugins.views*), 376
PluginSchemaDocuments (*in module archivebox.config.common*), 337
PluginsConfig (*class in archivebox.plugins.apps*), 370
PluginSpecialConfig (*class in archivebox.plugins.discovery*), 373
poll() (*archivebox.machine.models.Process method*), 109
post() (*archivebox.core.views.AddView method*), 247
postgres_db_params() (*in module archivebox.misc.db*), 125
postprocessing_plugins (*archivebox.core.forms.AddLinkForm attribute*), 234
pre_save() (*archivebox.base_models.models.AutoDateTimeField method*), 350
prepare_runtime_for_crawl() (*archivebox.personas.models.Persona method*), 307
prepare_runtime_for_snapshot() (*archivebox.personas.models.Persona method*), 307
pretty_path() (*in module archivebox.misc.logging_util*), 122
preview_icon() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
preview_view() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
print_combined_stats() (*in module archivebox.cli.archivebox_update*), 147

print_index_stats() (*in module archivebox.cli.archivebox_update*), 147
print_stats() (*in module archivebox.cli.archivebox_update*), 147
printable_config() (*in module archivebox.misc.logging_util*), 123
printable_dependency_version() (*in module archivebox.misc.logging_util*), 123
printable_filesize() (*in module archivebox.misc.logging_util*), 122
printable_folder_status() (*in module archivebox.misc.logging_util*), 123
printable_folders() (*in module archivebox.misc.logging_util*), 123
prioritize_metadata_matches() (*in module archivebox.search.query*), 292
proc (*archivebox.machine.models.Process property*), 108
process (*archivebox.core.models.ArchiveResult attribute*), 273
Process (*class in archivebox.machine.models*), 103
Process.Meta (*class in archivebox.machine.models*), 105
Process.StatusChoices (*class in archivebox.machine.models*), 103
Process.TypeChoices (*class in archivebox.machine.models*), 103
process_all_db_snapshots() (*in module archivebox.cli.archivebox_update*), 146
process_archiveresult_by_id() (*in module archivebox.cli.archivebox_extract*), 155
process_filtered_snapshots() (*in module archivebox.cli.archivebox_update*), 146
process_id (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
process_id (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
process_id (*archivebox.core.models.ArchiveResult attribute*), 274
process_link() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 226
PROCESS_PID_NAMESPACE_KEY (*in module archivebox.machine.models*), 96
process_record (*archivebox.core.models.ArchiveResult property*), 275
process_request() (*archivebox.core.middleware.ReverseProxyAuthMiddleware method*), 277
process_set (*archivebox.core.models.Snapshot property*), 265
process_stdin_records() (*in module archivebox.cli.archivebox_run*), 142
PROCESS_TIMEOUT_GRACE (*in module archivebox.machine.models*), 95
process_type (*archivebox.machine.models.Process attribute*), 104
ProcessAdmin (*class in archivebox.machine.admin*), 114
ProcessManager (*class in archivebox.machine.models*), 102
ProcessService (*class in archivebox.services.process_service*), 355
profile_copied (*archivebox.personas.importers.PersonaImportResult attribute*), 300
profile_copy_ignore() (*in module archivebox.personas.importers*), 300
profile_dir (*archivebox.personas.importers.PersonaImportSource attribute*), 300
profile_path (*archivebox.personas.importers.PersonaImportSource property*), 300
progress_bar() (*in module archivebox.misc.logging_util*), 122
progress_endpoint() (*in module archivebox.progressmonitor.views*), 117
ProgressMonitorConfig (*class in archivebox.progressmonitor.apps*), 116
project_abxpkg_derived_cache_to_db() (*in module archivebox.services.binary_service*), 357
project_discovered_snapshots() (*in module archivebox.services.snapshot_service*), 365
PROTOCOL_VERSION (*in module archivebox.mcp.server*), 296
prune_url() (*archivebox.crawls.models.Crawl method*), 314
prune_urls() (*archivebox.crawls.models.Crawl method*), 314
PUBLIC_ADD_VIEW (*archivebox.config.common.ServerConfig attribute*), 341
PUBLIC_INDEX (*archivebox.config.common.ServerConfig attribute*), 341
public_snapshot_search_stream_view() (*in module archivebox.search.views*), 291
public_snapshots_queryset() (*in module archivebox.core.permissions*), 240
PUBLIC_TOOLS (*in module archivebox.mcp.server*), 296
PublicIndexView (*class in archivebox.core.views*), 246
pwd (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
pwd (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
pwd (*archivebox.core.models.ArchiveResult property*), 276
pwd (*archivebox.machine.models.Process attribute*), 104

Q

query (*in module archivebox.misc.util*), 136
query_search_index() (*in module archivebox.search.query*), 292
QueryParamTokenAuth (*class in archivebox.api.auth*), 194
queryset() (*archivebox.core.admin_snapshots.SnapshotArchiveStateListFilter method*), 281
queryset() (*archivebox.core.admin_snapshots.SnapshotPermissionsListFilter method*), 280
queryset() (*archivebox.core.admin_snapshots.SnapshotResultHealthListFilter method*), 281
queryset() (*archivebox.core.admin_snapshots.SnapshotSizeListFilter method*), 281
queryset() (*archivebox.core.admin_snapshots.SnapshotStatusListFilter method*), 281
queryset() (*archivebox.core.admin_snapshots.TagNameListFilter method*), 280
queryset() (*archivebox.crawls.admin.MaxDepthListFilter method*), 316
queue_for_extraction() (*archivebox.core.models.Snapshot method*), 264
QUEUED (*archivebox.core.models.ArchiveResult.StatusChoices attribute*), 271
QUEUED (*archivebox.machine.models.Binary.StatusChoices attribute*), 99
QUEUED (*archivebox.machine.models.Process.StatusChoices attribute*), 103
QUEUED (*archivebox.workers.models.DefaultStatusChoices attribute*), 212
queued_count (*archivebox.api.v1_core.SnapshotDeleteResponseSchema attribute*), 184
quiet (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 201

R

rainbow (*in module archivebox.misc.logging*), 121
RainbowHighlighter (*class in archivebox.misc.logging*), 121
raise_if_shutdown_requested() (*in module archivebox.core.shutdown_util*), 250
RangedFileReader (*class in archivebox.misc.serve_static*), 131
raw_id_fields (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
read_args_or_stdin() (*in module archivebox.misc.jsonl*), 133
read_index_jsonl() (*archivebox.core.models.Snapshot method*), 268
read_ini_config() (*in module archivebox.config.configset*), 332
READ_ONLY_ACTIONS (*in module archivebox.mcp.server*), 296
read_stdin() (*in module archivebox.misc.jsonl*), 133
readonly_fields (*archivebox.api.admin.APITokenAdmin attribute*), 203
readonly_fields (*archivebox.api.admin.CustomWebhookAdmin attribute*), 204
readonly_fields (*archivebox.base_models.admin.BaseModelAdmin attribute*), 354
readonly_fields (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
readonly_fields (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 224
readonly_fields (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
readonly_fields (*archivebox.core.admin_tags.TagAdmin attribute*), 252
readonly_fields (*archivebox.core.admin_users.CustomUserAdmin attribute*), 285
readonly_fields (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
readonly_fields (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
readonly_fields (*archivebox.machine.admin.BinaryAdmin attribute*), 114
readonly_fields (*archivebox.machine.admin.MachineAdmin attribute*), 113
readonly_fields (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 113
readonly_fields (*archivebox.machine.admin.ProcessAdmin attribute*), 114
readonly_fields (*archivebox.personas.admin.PersonaAdmin attribute*), 308
ready() (*archivebox.api.apps.APIConfig method*), 168
ready() (*archivebox.core.apps.CoreConfig method*), 236
reap_foreground_supervisord_process() (*in module archivebox.workers.supervisord_util*), 210
rebuild_models_from_migration_state() (*in module archivebox.misc.db*), 126

reconcile_filesystem_links() (*archivebox.core.models.Snapshot method*), 269
reconcile_known_snapshots() (*in module archivebox.cli.rescan*), 156
reconcile_parent_lifecycle() (*archivebox.core.models.Snapshot method*), 265
reconcile_with_index() (*archivebox.core.models.Snapshot method*), 267
reconcile_with_index_json() (*archivebox.core.models.Snapshot method*), 267
reconfigure() (*archivebox._ReconfigurableStream method*), 379
recover_orchestrator_state() (*in module archivebox.core.recovery_util*), 242
recrawl() (*archivebox.crawls.admin.CrawlAdmin method*), 319
redact_env() (*in module archivebox.machine.env_util*), 112
redact_sensitive_config() (*in module archivebox.config.common*), 338
redo_failed_view() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
refresh_snapshot_output_sizes() (*archivebox.core.models.ArchiveResult static method*), 275
regex (*archivebox.base_models.admin.HexUUIDConverter attribute*), 353
regex_escape() (*archivebox.crawls.admin.CrawlAdminForm static method*), 317
region (*archivebox.machine.models.NetworkInterface attribute*), 98
register (*in module archivebox.core.templatetags.core_tags*), 217
register_admin() (*in module archivebox.api.admin*), 204
register_admin() (*in module archivebox.api.apps*), 168
register_admin() (*in module archivebox.core*), 287
register_admin() (*in module archivebox.core.admin*), 287
register_admin() (*in module archivebox.core.admin_archiveresults*), 226
register_admin() (*in module archivebox.core.admin_tags*), 253
register_admin() (*in module archivebox.core.admin_users*), 286
register_admin() (*in module archivebox.crawls*), 321
register_admin() (*in module archivebox.crawls.admin*), 320
register_admin() (*in module archivebox.machine.admin*), 115
register_admin() (*in module archivebox.machine.apps*), 93
register_admin() (*in module archivebox.personas.admin*), 308
register_admin() (*in module archivebox.workers*), 215
register_admin() (*in module archivebox.workers.admin*), 215
register_admin_site() (*in module archivebox.core.admin_site*), 254
register_delete_signal() (*archivebox.base_models.models.ModelWithOutputDir class method*), 352
register_delete_webhook_field_loader() (*in module archivebox.api.webhooks*), 172
register_sonic_daemon_event_handler() (*in module archivebox.search.sonic_daemon*), 288
register_urls() (*in module archivebox.api.v1_api*), 173
reindex_snapshots() (*in module archivebox.cli.archivebox_update*), 146
reject_shared_filesystem_sqlite_wal() (*archivebox.config.common.DatabaseConfig method*), 342
remaining_snapshot_capacity() (*archivebox.crawls.models.Crawl method*), 314
remaining_url_capacity() (*archivebox.crawls.models.Crawl method*), 314
remove() (*archivebox.core.models.SnapshotManager method*), 262
remove() (*in module archivebox.cli.archivebox_remove*), 138
remove_legacy_archive_symlink() (*archivebox.core.models.Snapshot method*), 269
remove_tag_ids() (*archivebox.core.models.Snapshot method*), 264
remove_tags() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 285
RemoveCommandSchema (*class in archivebox.api.v1_cli*), 202
rename_tag() (*in module archivebox.api.v1_core*), 190
rename_tag() (*in module archivebox.core.tag_util*), 279
rename_url (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
renamed_commands (*archivebox.cli.ArchiveBoxGroup attribute*), 167
render() (*archivebox.base_models.admin.KeyValueWidget method*), 354
render() (*archivebox.core.widgets.InlineTagEditorWidget method*), 227
render() (*archivebox.core.widgets.TagEditorWidget method*), 227
render() (*archivebox.core.widgets.URLFiltersWidget method*), 227
render_archiveresults_list() (*in module archive-*

box.core.admin_archiveresults), 223
render_binary_detail_description() (*in module archivebox.config.views*), 330
render_change_form() (*archivebox.base_models.admin.BaseModelAdmin method*), 355
render_change_form() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
render_change_form() (*archivebox.core.admin_tags.TagAdmin method*), 252
render_change_form() (*archivebox.personas.admin.PersonaAdmin method*), 308
render_code_block() (*in module archivebox.plugins.views*), 376
render_code_tag_list() (*in module archivebox.plugins.views*), 376
render_config_properties_html() (*in module archivebox.plugins.views*), 376
render_highlighted_json_block() (*in module archivebox.plugins.views*), 376
render_hook_links_html() (*in module archivebox.plugins.views*), 376
render_link_tag_list() (*in module archivebox.plugins.views*), 376
render_live_index() (*archivebox.core.views.SnapshotView static method*), 245
render_plugin_card_document() (*in module archivebox.core.templatetags.core_tags*), 220
render_plugin_full_response() (*in module archivebox.plugins.discovery*), 374
render_plugin_metadata_html() (*in module archivebox.plugins.views*), 376
render_property_links() (*in module archivebox.plugins.views*), 376
render_snapshots_list() (*in module archivebox.crawls.admin*), 316
renew_lease() (*archivebox.services.snapshot_service.SnapshotService method*), 365
REPLAY_AUTH_SALT (*in module archivebox.core.views*), 244
REPLAY_COOKIE_PREFIX (*in module archivebox.core.views*), 244
REPLAY_GRANT_MAX_AGE (*in module archivebox.core.views*), 244
REPO_ROOT (*in module archivebox.workers.models*), 213
request_delete() (*archivebox.core.models.SnapshotQuerySet method*), 261
require_dependency_install_memory() (*in module archivebox.workers.supervisord_util*), 208
require_server_worker_memory() (*in module archivebox.workers.supervisord_util*), 208
rescan_snapshots() (*in module archivebox.cli.rescan*), 156
RESERVED_ARCHIVE_DIR_NAMES (*archivebox.config.constants.ConstantsDict attribute*), 329
resnapshot_snapshot() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
RESOLUTION (*archivebox.config.common.ArchivingConfig attribute*), 343
resolve_archive_path() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolve_archive_size() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolve_archiveresults() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolve_browser_import_source() (*in module archivebox.personas.importers*), 301
resolve_browser_profile_source() (*in module archivebox.personas.importers*), 301
resolve_config() (*archivebox.api.v1_crawls.CrawlSchema static method*), 169
resolve_config() (*archivebox.api.v1_personas.PersonaSchema static method*), 175
resolve_created_by_id() (*archivebox.api.v1_core.MinimalArchiveResultSchema static method*), 179
resolve_created_by_id() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolve_created_by_id() (*archivebox.api.v1_core.TagSchema static method*), 186
resolve_created_by_id() (*archivebox.api.v1_crawls.CrawlSchema static method*), 169
resolve_created_by_id() (*archivebox.api.v1_personas.PersonaSchema static method*), 174
resolve_created_by_username() (*archivebox.api.v1_core.MinimalArchiveResultSchema static method*), 179
resolve_created_by_username() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolve_created_by_username() (*archive-*

box.api.v1_core.TagSchema static method), 186
resolve_created_by_username() (*archivebox.api.v1_crawls.CrawlSchema static method*), 169
resolve_created_by_username() (*archivebox.api.v1_personas.PersonaSchema static method*), 175
resolve_custom_import_source() (*in module archivebox.personas.importers*), 301
resolve_delete_after_config_value() (*in module archivebox.config.common*), 338
resolve_env_binary() (*in module archivebox.workers.supervisord_util*), 208
resolve_is_valid() (*archivebox.api.v1_machine.BinarySchema static method*), 197
resolve_machine_hostname() (*archivebox.api.v1_machine.BinarySchema static method*), 197
resolve_num_archiveresults() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolve_num_snapshots() (*archivebox.api.v1_core.TagSchema static method*), 187
resolve_output_files() (*archivebox.api.v1_core.MinimalArchiveResultSchema static method*), 179
resolve_output_mimetypes() (*archivebox.api.v1_core.MinimalArchiveResultSchema static method*), 179
resolve_output_size() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolve_persona() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolve_persona() (*archivebox.crawls.models.Crawl method*), 314
resolve_runtime_paths() (*archivebox.config.common.ArchiveBoxBaseConfig method*), 345
resolve_snapshot_id() (*archivebox.api.v1_core.ArchiveResultSchema static method*), 179
resolve_snapshot_tags() (*archivebox.api.v1_core.ArchiveResultSchema static method*), 180
resolve_snapshot_timestamp() (*archivebox.api.v1_core.ArchiveResultSchema static method*), 179
resolve_snapshot_url() (*archivebox.api.v1_core.ArchiveResultSchema static method*), 179
resolve_snapshots() (*archivebox.api.v1_core.TagSchema static method*), 187
resolve_snapshots() (*archivebox.api.v1_crawls.CrawlSchema static method*), 169
resolve_source_browser_binary() (*in module archivebox.personas.importers*), 301
resolve_tags() (*archivebox.api.v1_core.SnapshotSchema static method*), 183
resolved_title (*archivebox.core.models.Snapshot property*), 269
resource_admission (*in module archivebox.services.resource_admission*), 361
RESOURCE_RECHECK_SECONDS (*in module archivebox.services.resource_admission*), 360
ResourceAdmission (*class in archivebox.services.resource_admission*), 360
response_add() (*archivebox.core.admin_tags.TagAdmin method*), 252
response_change() (*archivebox.core.admin_tags.TagAdmin method*), 252
restore_paused_scheduler_marker() (*archivebox.core.models.Snapshot method*), 265
result (*archivebox.api.v1_cli.CLICommandResponseSchema attribute*), 199
result_format (*archivebox.api.v1_cli.CLICommandResponseSchema attribute*), 199
result_id() (*archivebox.core.admin_archiveresults.ArchiveResultInline method*), 224
result_list() (*in module archivebox.core.templatetags.core_tags*), 219
resume (*archivebox.api.v1_cli.UpdateCommandSchema attribute*), 201
resume() (*archivebox.core.models.Snapshot method*), 265
resume() (*archivebox.crawls.models.Crawl method*), 313
resume() (*archivebox.workers.models.ModelWithQueue method*), 214
resume_selected_crawls() (*archivebox.crawls.admin.CrawlAdmin method*), 319
retry_at (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
retry_at (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
retry_at (*archivebox.api.v1_core.SnapshotUpdateSchema*

attribute), 183
retry_at (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
retry_at (*archivebox.api.v1_crawls.CrawlUpdateSchema attribute*), 169
retry_at (*archivebox.core.models.ArchiveResult attribute*), 273
retry_at (*archivebox.core.models.Snapshot attribute*), 262
retry_at (*archivebox.crawls.models.Crawl attribute*), 311
retry_at (*archivebox.machine.models.Binary attribute*), 100
retry_at (*archivebox.machine.models.Process attribute*), 105
retry_at (*archivebox.workers.models.ModelWithQueue attribute*), 213
RETRY_AT (*archivebox.workers.models.ModelWithQueue property*), 214
retry_at_field_name (*archivebox.core.models.Snapshot attribute*), 263
retry_at_field_name (*archivebox.crawls.models.Crawl attribute*), 311
RETRY_AT_MAX (*in module archivebox.workers.models*), 213
retry_failed_archiveresults() (*archivebox.core.models.Snapshot method*), 270
retry_sqlite_locks() (*in module archivebox.misc.db*), 126
RetryAtField() (*archivebox.workers.models.ModelWithQueue class method*), 214
REVERSE_PROXY_USER_HEADER (*archivebox.config.common.ServerConfig attribute*), 341
REVERSE_PROXY_WHITELIST (*archivebox.config.common.ServerConfig attribute*), 341
ReverseProxyAuthMiddleware (*class in archivebox.core.middleware*), 277
RISKY_REPLAY_EXTENSIONS (*in module archivebox.misc.serve_static*), 130
RISKY_REPLAY_MARKERS (*in module archivebox.misc.serve_static*), 130
RISKY_REPLAY_MIMETYPES (*in module archivebox.misc.serve_static*), 130
ROBOTS_TXT_FILENAME (*archivebox.config.constants.ConstantsDict attribute*), 328
root (*archivebox.machine.models.Process property*), 107
root_data_dir_handoff_paths() (*in module archivebox.config.permissions*), 325
ROOT_HANDOFF_NAMES (*in module archivebox.config.permissions*), 325
root_parent_can_grant_group_traversal() (*in module archivebox.config.permissions*), 325
root_should_handoff_data_dir() (*in module archivebox.config.permissions*), 325
ROOT_URLCONF (*in module archivebox.core.settings*), 229
router (*in module archivebox.api.v1_auth*), 192
router (*in module archivebox.api.v1_cli*), 199
router (*in module archivebox.api.v1_core*), 177
router (*in module archivebox.api.v1_crawls*), 168
router (*in module archivebox.api.v1_machine*), 195
router (*in module archivebox.api.v1_personas*), 173
rprint() (*in module archivebox.config.common*), 339
run() (*archivebox.machine.models.Binary method*), 101
run() (*archivebox.services.runner.CrawlRunner method*), 363
run_all (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 201
run_binary() (*in module archivebox.services.runner*), 363
run_crawl() (*archivebox.services.runner.CrawlRunner method*), 363
run_crawl() (*in module archivebox.services.runner*), 363
run_db_analyze_batch() (*in module archivebox.misc.db*), 126
run_due_binary() (*in module archivebox.services.runner*), 364
run_due_crawl() (*in module archivebox.services.runner*), 364
run_due_snapshot() (*in module archivebox.services.runner*), 364
run_install() (*in module archivebox.services.runner*), 364
run_mcp_server() (*in module archivebox.mcp.server*), 298
run_pending_crawls() (*in module archivebox.services.runner*), 364
run_plugins() (*in module archivebox.cli.archivebox_extract*), 155
run_runner() (*in module archivebox.cli.archivebox_run*), 143
run_runner_worker() (*in module archivebox.workers.supervisord_util*), 210
run_scheduled_maintenance() (*in module archivebox.cli.archivebox_update*), 146
run_snapshot() (*archivebox.services.runner.CrawlRunner method*), 363
run_snapshot_maintenance() (*in module archivebox.services.runner*), 363
run_snapshot_worker() (*in module archivebox.cli.archivebox_run*), 143
run_stdio_server() (*archivebox.mcp.server.MCPServer method*), 298

run_watchdog() (*in module archivebox.workers.supervisord_parent_watchdog*), 212
RUNNABLE_STATES (*archivebox.core.models.Snapshot attribute*), 263
RUNNABLE_STATES (*archivebox.crawls.models.Crawl attribute*), 312
RUNNER_ACTIVE_WORKER_TYPE (*in module archivebox.core.takeover_util*), 241
RUNNER_DAEMON_ENV (*in module archivebox.cli.archivebox_run*), 142
RUNNER_GATE_WORKER_TYPES (*in module archivebox.core.takeover_util*), 241
RUNNER_ONCE_WORKER() (*in module archivebox.workers.supervisord_util*), 209
RUNNER_WAITING_WORKER_TYPE (*in module archivebox.core.takeover_util*), 241
RUNNER_WATCH_WORKER() (*in module archivebox.workers.supervisord_util*), 209
RUNNER_WORKER() (*in module archivebox.workers.supervisord_util*), 209
RUNNING (*archivebox.machine.models.Process.StatusChoices attribute*), 103
RUNNING_AS_GID (*archivebox.config.constants.ConstantsDict attribute*), 327
RUNNING_AS_GID (*in module archivebox.config.permissions*), 324
RUNNING_AS_UID (*archivebox.config.constants.ConstantsDict attribute*), 327
RUNNING_AS_UID (*in module archivebox.config.permissions*), 324
RUNSERVER_WORKER() (*in module archivebox.workers.supervisord_util*), 209
runtime_components_for_worker_names() (*in module archivebox.workers.supervisord_util*), 210
runtime_derived_config_keys() (*archivebox.config.common.ArchiveBoxBaseConfig class method*), 345
runtime_dir (*archivebox.machine.models.Process property*), 108
runtime_downloads_dir_for_crawl() (*archivebox.personas.models.Persona method*), 307
runtime_downloads_dir_for_snapshot() (*archivebox.personas.models.Persona method*), 307
runtime_profile_dir_for_crawl() (*archivebox.personas.models.Persona method*), 306
runtime_profile_dir_for_snapshot() (*archivebox.personas.models.Persona method*), 307
runtime_root_for_crawl() (*archivebox.personas.models.Persona method*), 306
runtime_root_for_snapshot() (*archivebox.personas.models.Persona method*), 307
runtime_stack_owner() (*in module archivebox.core.takeover_util*), 241

S

safe_update() (*archivebox.core.models.ArchiveResult method*), 274
safe_update() (*archivebox.machine.models.Process method*), 106
safe_update() (*archivebox.workers.models.ModelWithQueue method*), 214
sanitize_extracted_url() (*in module archivebox.misc.util*), 137
sanitize_html_text() (*in module archivebox.misc.util*), 136
save() (*archivebox.base_models.models.ModelWithConfig method*), 351
save() (*archivebox.base_models.models.ModelWithDeleteAfter method*), 351
save() (*archivebox.core.admin_snapshots.SnapshotAdminForm method*), 282
save() (*archivebox.core.models.ArchiveResult method*), 274
save() (*archivebox.core.models.Snapshot method*), 265
save() (*archivebox.core.models.Tag method*), 260
save() (*archivebox.crawls.admin.CrawlAdminForm method*), 317
save() (*archivebox.crawls.models.Crawl method*), 313
save() (*archivebox.crawls.models.CrawlSchedule method*), 310
save() (*archivebox.machine.models.Machine method*), 98
save() (*archivebox.personas.models.Persona method*), 306
save() (*archivebox.workers.models.ModelWithQueue method*), 214
save_model() (*archivebox.base_models.admin.ConfigEditorMixin method*), 354
save_model() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
save_model() (*archivebox.personas.admin.PersonaAdmin method*), 308
save_on_top (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 225
save_on_top (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
save_related() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 282

save_tags() (*archivebox.core.models.Snapshot method*), 269
schedule (*archivebox.core.forms.AddLinkForm attribute*), 235
schedule (*archivebox.crawls.models.Crawl attribute*), 311
schedule (*archivebox.crawls.models.CrawlSchedule attribute*), 310
schedule() (*in module archivebox.cli.archivebox_schedule*), 141
SCHEDULE_ALIASES (*in module archivebox.crawls.schedule_util*), 309
schedule_child_snapshots_for_pause() (*archivebox.crawls.models.Crawl method*), 313
schedule_child_snapshots_for_sealing() (*archivebox.crawls.models.Crawl method*), 313
schedule_delete_cleanup() (*archivebox.base_models.models.ModelWithOutputDir method*), 352
schedule_delete_cleanup() (*archivebox.core.models.ArchiveResult method*), 274
schedule_id (*archivebox.crawls.models.Crawl attribute*), 312
schedule_plugin_run() (*archivebox.core.models.Snapshot method*), 265
schedule_str() (*archivebox.crawls.admin.CrawlAdmin method*), 319
ScheduleCommandSchema (*class in archivebox.api.v1_cli*), 201
scheme (*in module archivebox.misc.util*), 136
scheme() (*archivebox.core.models.Snapshot method*), 270
scope_for_key() (*archivebox.config.common.ArchiveBoxBaseConfig class method*), 345
seal() (*archivebox.core.models.Snapshot method*), 265
seal() (*archivebox.crawls.models.Crawl method*), 315
seal_selected_crawls() (*archivebox.crawls.admin.CrawlAdmin method*), 319
seal_snapshot_due_to_limit() (*archivebox.services.runner.CrawlRunner method*), 363
SEALED (*archivebox.workers.models.DefaultStatusChoices attribute*), 213
search (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 201
search (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
search (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
SEARCH (*archivebox.machine.models.Process.TypeChoices attribute*), 103
search() (*archivebox.core.models.SnapshotQuerySet method*), 261
search_backend_command_env() (*in module archivebox.search.backends*), 289
SEARCH_BACKEND_ENGINE (*archivebox.config.common.SearchBackendConfig attribute*), 344
search_fields (*archivebox.api.admin.APITokenAdmin attribute*), 203
search_fields (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
search_fields (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
search_fields (*archivebox.core.admin_tags.TagAdmin attribute*), 252
search_fields (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
search_fields (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
search_fields (*archivebox.machine.admin.BinaryAdmin attribute*), 114
search_fields (*archivebox.machine.admin.MachineAdmin attribute*), 113
search_fields (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 113
search_fields (*archivebox.machine.admin.ProcessAdmin attribute*), 114
search_fields (*archivebox.personas.admin.PersonaAdmin attribute*), 307
search_mode (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
SEARCH_MODES (*in module archivebox.search.config*), 293
SEARCH_RESULT_CACHE_TTL (*in module archivebox.search.views*), 290
search_stream_view() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
search_tags() (*in module archivebox.api.v1_core*), 189
SearchBackendConfig (*class in archivebox.config.common*), 344
SearchConfig (*class in archivebox.search.apps*), 288
SearchResultsAdminMixin (*class in archive-*

box.search.admin), 294
SearchResultsChangeList (*class in archivebox.search.admin*), 293
SECRET_KEY (*archivebox.config.common.ServerConfig attribute*), 340
SECRET_KEY (*in module archivebox.core.settings*), 231
SECURE_BROWSER_XSS_FILTER (*in module archivebox.core.settings*), 231
SECURE_CONTENT_TYPE_NOSNIFF (*in module archivebox.core.settings*), 231
SECURE_REFERRER_POLICY (*in module archivebox.core.settings*), 231
seekable() (*archivebox.misc.serve_static._StreamingQueueWriter method*), 129
select_archivebox_user() (*in module archivebox.config.permissions*), 324
SENSITIVE_CONFIG_VALUE_REDACTED (*in module archivebox.config.common*), 338
SENSITIVE_ENV_KEY_PARTS (*in module archivebox.machine.env_util*), 112
SENSITIVE_QUERY_PARAM_RE (*in module archivebox.misc.monkey_patches*), 134
serve_plugin_replay_asset() (*in module archivebox.plugins.discovery*), 374
serve_static() (*in module archivebox.misc.serve_static*), 131
serve_static_with_byterange_support() (*in module archivebox.misc.serve_static*), 131
SERVER (*archivebox.machine.models.Process.TypeChoices attribute*), 103
server() (*in module archivebox.cli.archivebox_server*), 162
SERVER_SECURITY_MODE (*archivebox.config.common.ServerConfig attribute*), 341
SERVER_SECURITY_MODES (*archivebox.config.common.ServerConfig attribute*), 340
SERVER_WORKER() (*in module archivebox.workers.supervisord_util*), 209
ServerConfig (*class in archivebox.config.common*), 340
ServerSecurityModeMiddleware() (*in module archivebox.core.middleware*), 277
SESSION_COOKIE_AGE (*in module archivebox.core.settings*), 232
SESSION_COOKIE_DOMAIN (*in module archivebox.core.settings*), 231
SESSION_COOKIE_HTTPONLY (*in module archivebox.core.settings*), 231
SESSION_COOKIE_NAME (*in module archivebox.core.settings*), 231
SESSION_COOKIE_SECURE (*in module archivebox.core.settings*), 231
SESSION_ENGINE (*in module archivebox.core.settings*), 232
SESSION_EXPIRE_AT_BROWSER_CLOSE (*in module archivebox.core.settings*), 232
SESSION_SAVE_EVERY_REQUEST (*in module archivebox.core.settings*), 232
set_crawl_permissions() (*archivebox.crawls.admin.CrawlAdmin method*), 319
set_delete_at_from_config() (*archivebox.base_models.models.ModelWithDeleteAfter method*), 352
set_permissions_view() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
set_permissions_view() (*archivebox.crawls.admin.CrawlAdmin method*), 319
set_snapshot_permissions() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 283
set_url_filters() (*archivebox.crawls.models.Crawl method*), 313
settings (*archivebox.api.v1_personas.PersonaSyncSchema attribute*), 174
settings_customise_sources() (*archivebox.config.configset.BaseConfigSet class method*), 332
SETTINGS_LOGGING (*in module archivebox.core.settings_logging*), 238
setup_commands (*archivebox.cli.ArchiveBoxGroup attribute*), 167
setup_django() (*in module archivebox.config.django*), 336
sha256 (*archivebox.api.v1_machine.BinarySchema attribute*), 197
sha256 (*archivebox.machine.models.Binary attribute*), 100
shares_pid_namespace (*archivebox.machine.models.Process property*), 108
shell() (*in module archivebox.cli.archivebox_shell*), 148
SHELL_ENV_KEY_RE (*in module archivebox.machine.env_util*), 112
SHELL_PLUS (*in module archivebox.core.settings*), 232
SHELL_PLUS_PRINT_SQL (*in module archivebox.core.settings*), 232
shell_to_mcp_tool() (*in module archivebox.mcp.server*), 297
ShellConfig (*class in archivebox.config.common*), 339
SHORT_DATETIME_FORMAT (*in module archivebox.core.settings*), 232
short_id() (*archivebox.crawls.admin.CrawlAdmin method*), 319
should_annotate_snapshot_counts() (*archive-*

box.crawls.admin.CrawlAdmin method), 318
SHOULD_NEUTER_RISKY_REPLAY (*archivebox.config.common.ServerConfig property*), 341
show (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 201
show_change_link (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 224
show_full_result_count (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 225
show_full_result_count (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 283
show_full_result_count (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
SHOW_PROGRESS (*archivebox.config.common.ShellConfig attribute*), 339
show_search_mode_selector (*archivebox.base_models.admin.BaseModelAdmin attribute*), 354
show_search_mode_selector (*archivebox.search.admin.SearchResultsAdminMixin attribute*), 294
ShutdownSignalState (*class in archivebox.core.shutdown_util*), 250
signal_name (*archivebox.core.shutdown_util.ShutdownSignalState attribute*), 250
SIGNAL_WEBHOOKS (*in module archivebox.core.settings*), 233
SIGNAL_WEBHOOKS_CUSTOM_MODEL (*in module archivebox.core.settings*), 232
site_header (*archivebox.core.admin_site.ArchiveBoxAdmin attribute*), 254
site_title (*archivebox.core.admin_site.ArchiveBoxAdmin attribute*), 254
size() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
size_with_stats() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
SKIPPED (*archivebox.core.models.ArchiveResult.StatusChoices attribute*), 272
slots_for_headroom() (*archivebox.services.resource_admission.ResourceAdmission static method*), 360
slug (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
slug (*archivebox.core.models.Tag property*), 260
snapshot (*archivebox.core.models.ArchiveResult attribute*), 273
snapshot (*archivebox.core.models.SnapshotTag attribute*), 261
Snapshot (*class in archivebox.core.models*), 262
Snapshot.Meta (*class in archivebox.core.models*), 264
snapshot_base_url() (*in module archivebox.core.templatetags.core_tags*), 220
snapshot_count_badge() (*archivebox.core.admin_users.CustomUserAdmin method*), 286
snapshot_count_column() (*archivebox.core.admin_users.CustomUserAdmin method*), 286
snapshot_count_expr() (*archivebox.core.models.ArchiveResult class method*), 272
snapshot_count_subquery() (*archivebox.core.models.ArchiveResult class method*), 272
snapshot_detail_url() (*in module archivebox.core.templatetags.core_tags*), 220
snapshot_dir() (*archivebox.core.models.ArchiveResult method*), 275
snapshot_files_url() (*in module archivebox.core.templatetags.core_tags*), 220
snapshot_filter_kwargs() (*in module archivebox.api.v1_cli*), 202
snapshot_filter_options() (*in module archivebox.cli.archivebox_snapshot*), 160
SNAPSHOT_FILTER_TYPE_CHOICES (*in module archivebox.cli.archivebox_snapshot*), 160
SNAPSHOT_FIRST_VALUES (*archivebox.core.admin_snapshots.SnapshotResultHealthListFilter attribute*), 281
snapshot_half_count_subquery() (*archivebox.core.models.ArchiveResult class method*), 272
snapshot_id (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
snapshot_id (*archivebox.api.v1_core.ArchiveResultSchema attribute*), 179
snapshot_id (*archivebox.api.v1_core.SnapshotDeleteResponseSchema attribute*), 184
snapshot_id (*archivebox.api.v1_core.TagSnapshotRequestSchema attribute*), 189
snapshot_id (*archivebox.core.models.ArchiveResult attribute*), 274
snapshot_ids (*archivebox.api.v1_cli.AddCommandSchema attribute*),

200
snapshot_ids_with_majority_status() (archivebox.core.models.ArchiveResult class method), 273
snapshot_index_row() (in module archivebox.core.templatetags.core_tags), 220
snapshot_info() (archivebox.core.admin_archiveresults.ArchiveResultAdmin method), 225
snapshot_limit() (archivebox.services.resource_admission.ResourceAdmission method), 360
snapshot_link() (archivebox.machine.admin.ProcessAdmin method), 115
SNAPSHOT_LIST_CHUNK_SIZE (in module archivebox.cli.archivebox_snapshot), 160
snapshot_max_size (archivebox.api.v1_cli.AddCommandSchema attribute), 200
SNAPSHOT_MAX_SIZE (archivebox.config.common.ArchivingConfig attribute), 343
snapshot_max_size (archivebox.core.forms.AddLinkForm attribute), 234
snapshot_metadata_search_waves() (in module archivebox.search.query), 292
snapshot_output_options() (in module archivebox.cli.archivebox_snapshot), 160
SNAPSHOT_PERMISSION_META (in module archivebox.core.admin_snapshots), 280
snapshot_preview_url() (in module archivebox.core.templatetags.core_tags), 220
snapshot_result_file_url() (in module archivebox.core.templatetags.core_tags), 220
snapshot_role_output_url() (in module archivebox.core.templatetags.core_tags), 220
snapshot_rss_badge() (archivebox.core.admin_users.CustomUserAdmin method), 286
snapshot_search_stream_response() (in module archivebox.search.views), 291
snapshot_set (archivebox.core.models.Tag attribute), 260
snapshot_set (archivebox.crawls.models.Crawl attribute), 312
snapshot_set() (archivebox.core.admin_users.CustomUserAdmin method), 286
snapshot_status_values() (in module archivebox.core.snapshot_status), 236
snapshot_summary() (archivebox.core.admin_snapshots.SnapshotAdmin method), 284
snapshot_tag (archivebox.api.v1_core.ArchiveResultFilterSchema attribute), 180
snapshot_tags (archivebox.api.v1_core.ArchiveResultSchema attribute), 179
snapshot_thumbnail() (in module archivebox.core.templatetags.core_tags), 220
snapshot_thumbnail_result() (in module archivebox.plugins.output_groups), 371
snapshot_timestamp (archivebox.api.v1_core.ArchiveResultSchema attribute), 179
snapshot_url (archivebox.api.v1_core.ArchiveResultFilterSchema attribute), 180
snapshot_url (archivebox.api.v1_core.ArchiveResultSchema attribute), 179
snapshot_url() (in module archivebox.core.templatetags.core_tags), 220
snapshot_url_text() (in module archivebox.core.templatetags.core_tags), 217
snapshot_zip_url() (in module archivebox.core.templatetags.core_tags), 220
SnapshotActionForm (class in archivebox.core.admin_snapshots), 280
SnapshotAdmin (class in archivebox.core.admin_snapshots), 282
SnapshotAdminForm (class in archivebox.core.admin_snapshots), 282
SnapshotAdminForm.Meta (class in archivebox.core.admin_snapshots), 282
SnapshotArchiveStateListFilter (class in archivebox.core.admin_snapshots), 281
SnapshotChangeList (class in archivebox.core.admin_snapshots), 281
SnapshotCreateSchema (class in archivebox.api.v1_core), 183
SnapshotDeleteResponseSchema (class in archivebox.api.v1_core), 184
SnapshotFilterCommandSchema (class in archivebox.api.v1_cli), 200
SnapshotFilterSchema (class in archivebox.api.v1_core), 184
SnapshotHostView (class in archivebox.core.views), 246
SnapshotManager (class in archivebox.core.models), 261
SnapshotMigrationError, 259
SnapshotPathView (class in archivebox.core.views), 245
SnapshotPermissionsListFilter (class in archive-

box.core.admin_snapshots), 280
SnapshotQuerySet (*class in archivebox.core.models*), 261
SnapshotReplayAuthView (*class in archivebox.core.views*), 244
SnapshotReplayView (*class in archivebox.core.views*), 246
SnapshotResultHealthListFilter (*class in archivebox.core.admin_snapshots*), 281
snapshots (*archivebox.api.v1_core.TagSchema attribute*), 186
snapshots (*archivebox.api.v1_core.TagSearchCardSchema attribute*), 188
snapshots() (*archivebox.core.admin_tags.TagAdmin method*), 252
snapshots() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
snapshots_changelist() (*archivebox.crawls.admin.CrawlAdmin method*), 319
SNAPSHOTS_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
SNAPSHOTS_PER_PAGE (*archivebox.config.common.ServerConfig attribute*), 341
SnapshotSchema (*class in archivebox.api.v1_core*), 182
SnapshotService (*class in archivebox.services.snapshot_service*), 365
SnapshotSizeListFilter (*class in archivebox.core.admin_snapshots*), 281
SnapshotStatusListFilter (*class in archivebox.core.admin_snapshots*), 280
SnapshotTag (*class in archivebox.core.models*), 260
SnapshotTag.Meta (*class in archivebox.core.models*), 261
SnapshotUpdateSchema (*class in archivebox.api.v1_core*), 183
SnapshotView (*class in archivebox.core.views*), 245
sort (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 201
sort (*archivebox.api.v1_core.TagSearchResponseSchema attribute*), 188
sort_fields (*archivebox.api.admin.APITokenAdmin attribute*), 203
sort_fields (*archivebox.api.admin.CustomWebhookAdmin attribute*), 204
sort_fields (*archivebox.core.admin_archiveresults.ArchiveResultAdmin attribute*), 224
sort_fields (*archivebox.core.admin_archiveresults.ArchiveResultInline attribute*), 224
sort_fields (*archivebox.core.admin_snapshots.SnapshotAdmin attribute*), 282
sort_fields (*archivebox.core.admin_users.CustomUserAdmin attribute*), 285
sort_fields (*archivebox.crawls.admin.CrawlAdmin attribute*), 318
sort_fields (*archivebox.crawls.admin.CrawlScheduleAdmin attribute*), 320
sort_fields (*archivebox.machine.admin.BinaryAdmin attribute*), 114
sort_fields (*archivebox.machine.admin.MachineAdmin attribute*), 113
sort_fields (*archivebox.machine.admin.NetworkInterfaceAdmin attribute*), 113
sort_fields (*archivebox.machine.admin.ProcessAdmin attribute*), 114
source (*archivebox.personas.importers.PersonaImportResult attribute*), 300
source_name (*archivebox.personas.importers.PersonaImportSource attribute*), 300
SOURCES_DIR (*archivebox.config.constants.ConstantsDict attribute*), 328
SOURCES_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
SPECIAL_FILTER_ARG_KEYS (*archivebox.core.models.SnapshotQuerySet attribute*), 261
split() (*in module archivebox.core.templatetags.core_tags*), 219
split_comma_separated_urls() (*in module archivebox.misc.util*), 137
split_filter_patterns() (*archivebox.crawls.models.Crawl static method*), 313
split_host_port() (*in module archivebox.core.routes_util*), 256
SQL_INDEX_FILENAME (*archivebox.config.constants.ConstantsDict attribute*), 328
SQLITE_BUSY_TIMEOUT (*archivebox.config.common.DatabaseConfig attribute*), 342
SQLITE_CONNECTION_OPTIONS (*in module archivebox.core.settings*), 230
SQLITE_JOURNAL_MODE (*archive-*

box.config.common.DatabaseConfig attribute), 342
SQLITE_JOURNAL_MODE (*in module archivebox.core.settings*), 230
sqlite_lock_error() (*in module archivebox.misc.db*), 126
sqlite_lock_holders() (*in module archivebox.misc.db*), 126
SQLITE_LOCK_RETRY_INTERVAL (*archivebox.config.common.DatabaseConfig attribute*), 342
SQLITE_LOCK_RETRY_TIMEOUT (*archivebox.config.common.DatabaseConfig attribute*), 342
SQLITE_MMAP_SIZE (*archivebox.config.common.DatabaseConfig attribute*), 342
SQLITE_MMAP_SIZE (*in module archivebox.core.settings*), 230
SQLiteCursorWrapper (*class in archivebox.core.sqlite_backend.base*), 222
standby_until_foreground_runner_needed() (*in module archivebox.core.takeover_util*), 241
standby_until_runtime_stack_needed() (*in module archivebox.core.takeover_util*), 241
start_new_supervisord_process() (*in module archivebox.workers.supervisord_util*), 210
start_paused (*archivebox.core.forms.AddLinkForm attribute*), 235
start_processing() (*archivebox.core.models.Snapshot method*), 265
start_server_workers() (*in module archivebox.workers.supervisord_util*), 211
START_TIME_TOLERANCE (*in module archivebox.machine.models*), 95
start_ts (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 179
start_ts (*archivebox.core.models.ArchiveResult attribute*), 273
start_worker() (*in module archivebox.workers.supervisord_util*), 210
STARTED (*archivebox.core.models.ArchiveResult.StatusChoices attribute*), 271
STARTED (*archivebox.workers.models.DefaultStatusChoices attribute*), 212
started_at (*archivebox.machine.models.Process attribute*), 104
STATE (*archivebox.workers.models.ModelWithQueue property*), 214
state_field_name (*archivebox.core.models.Snapshot attribute*), 263
state_field_name (*archivebox.crawls.models.Crawl attribute*), 311
static_archive_path (*archivebox.core.models.Snapshot property*), 271
STATIC_DIR (*archivebox.config.constants.ConstantsDict attribute*), 327
STATIC_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
static_export_root_url() (*in module archivebox.core.templatetags.core_tags*), 220
STATIC_URL (*in module archivebox.core.settings*), 230
STATICFILE_EXTENSIONS (*archivebox.config.constants.ConstantsDict attribute*), 329
STATICFILES_DIRS (*in module archivebox.core.settings*), 230
stats (*archivebox.api.v1_machine.MachineSchema attribute*), 196
stats (*archivebox.machine.models.Machine attribute*), 97
status (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 200
status (*archivebox.api.v1_core.ArchiveResultFilterSchema attribute*), 180
status (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
status (*archivebox.api.v1_core.SnapshotCreateSchema attribute*), 184
status (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
status (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
status (*archivebox.api.v1_core.SnapshotUpdateSchema attribute*), 183
status (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
status (*archivebox.api.v1_crawls.CrawlUpdateSchema attribute*), 169
status (*archivebox.api.v1_machine.BinaryFilterSchema attribute*), 198
status (*archivebox.api.v1_machine.BinarySchema attribute*), 197
status (*archivebox.core.models.ArchiveResult attribute*), 273
status (*archivebox.core.models.Snapshot attribute*), 263
status (*archivebox.crawls.models.Crawl attribute*), 311
status (*archivebox.machine.models.Binary attribute*), 100
status (*archivebox.machine.models.Process attribute*), 105
status (*archivebox.workers.models.ModelWithQueue attribute*), 213
status() (*in module archivebox.cli.archivebox_status*), 158
status_badge() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin*

method), 226
status_badge() (*archivebox.machine.admin.ProcessAdmin method*), 115
status_counts() (*archivebox.core.models.ArchiveResult class method*), 273
status_counts() (*archivebox.workers.models.ModelWithQueue class method*), 214
status_info() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
status_with_progress() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
status_with_stop_reason() (*archivebox.crawls.admin.CrawlAdmin method*), 319
StatusChoices (*archivebox.core.models.Snapshot attribute*), 263
StatusChoices (*archivebox.crawls.models.Crawl attribute*), 312
StatusChoices (*archivebox.workers.models.ModelWithQueue attribute*), 213
StatusField() (*archivebox.workers.models.ModelWithQueue class method*), 214
stderr (*archivebox.api.v1_cli.CLICommandResponseSchema attribute*), 199
stderr (*archivebox.machine.models.Process attribute*), 104
STDERR (*in module archivebox.cli*), 166
STDERR (*in module archivebox.config.django*), 336
STDERR (*in module archivebox.misc.logging*), 121
stderr() (*in module archivebox.misc.logging*), 121
stderr_display() (*archivebox.machine.admin.ProcessAdmin method*), 115
stderr_file (*archivebox.machine.models.Process property*), 108
stdout (*archivebox.api.v1_cli.CLICommandResponseSchema attribute*), 199
stdout (*archivebox.machine.models.Process attribute*), 104
stdout() (*in module archivebox.misc.logging*), 121
stdout_display() (*archivebox.machine.admin.ProcessAdmin method*), 115
stdout_file (*archivebox.machine.models.Process property*), 108
stop_all() (*archivebox.machine.models.Process class method*), 111
stop_existing_supervisord_process() (*in module archivebox.workers.supervisord_util*), 210
stop_own_supervisord_process() (*in module archivebox.workers.supervisord_util*), 210
stop_reason() (*archivebox.crawls.models.Crawl method*), 314
stop_reason_display() (*archivebox.crawls.admin.CrawlAdmin method*), 319
stop_reason_for_crawl() (*archivebox.crawls.admin.CrawlAdmin method*), 319
stop_snapshot_tasks() (*archivebox.services.runner.CrawlRunner method*), 363
stop_stale_sonic_processes() (*in module archivebox.workers.supervisord_util*), 211
stop_worker() (*in module archivebox.workers.supervisord_util*), 211
storage_captured (*archivebox.personas.importers.PersonaImportResult attribute*), 300
StorageConfig (*class in archivebox.config.common*), 340
STORAGES (*in module archivebox.core.settings*), 231
stringify_env_value() (*in module archivebox.machine.env_util*), 112
strip_role_subdomain() (*in module archivebox.core.routes_util*), 256
StripANSIColorCodesFilter (*class in archivebox.core.settings_logging*), 238
subpath_prefix() (*archivebox.crawls.admin.CrawlAdminForm static method*), 317
SUCCEEDED (*archivebox.core.models.ArchiveResult.StatusChoices attribute*), 272
success (*archivebox.api.v1_cli.CLICommandResponseSchema attribute*), 199
success (*archivebox.api.v1_core.SnapshotDeleteResponseSchema attribute*), 184
success (*archivebox.api.v1_core.TagCreateResponseSchema attribute*), 187
success (*archivebox.api.v1_core.TagDeleteResponseSchema attribute*), 189
success (*archivebox.api.v1_core.TagSnapshotResponseSchema attribute*), 189
success (*archivebox.api.v1_core.TagUpdateResponseSchema attribute*), 189
success (*archivebox.api.v1_crawls.CrawlDeleteResponseSchema attribute*), 170
success (*archivebox.api.v1_personas.PersonaSyncResponseSchema attribute*), 175
SUDO_GID (*in module archivebox.config.permissions*), 324

SUDO_UID (*in module archivebox.config.permissions*), 324
SudoPermission() (*in module archivebox.config.permissions*), 325
SUPERVISORD (*archivebox.machine.models.Process.TypeChoices attribute*), 103
SUPERVISORD_PARENT_WATCHDOG_WORKER() (*in module archivebox.workers.supervisord_util*), 209
SUPERVISORD_SOCKET_FILENAME (*in module archivebox.config.paths*), 334
SupervisordConnectionCache (*class in archivebox.workers.supervisord_util*), 210
swap_source_identifiers() (*in module archivebox.services.resource_admission*), 360
symlink_to_lib_bin() (*archivebox.machine.models.Binary method*), 102
symlink_to_lib_bin_after_commit() (*archivebox.machine.models.Binary method*), 102
sync_machine_and_file() (*in module archivebox.config.collection*), 323
sync_persona() (*in module archivebox.api.v1_personas*), 175
sync_supervisord_workers() (*in module archivebox.workers.supervisord_util*), 210
system_warnings_banner() (*in module archivebox.core.templatetags.core_tags*), 219

T

tag (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
tag (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 202
tag (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 201
tag (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
tag (*archivebox.core.forms.AddLinkForm attribute*), 234
tag (*archivebox.core.models.SnapshotTag attribute*), 261
Tag (*class in archivebox.core.models*), 259
Tag.Meta (*class in archivebox.core.models*), 260
TAG_HAS_SNAPSHOTS_CHOICES (*in module archivebox.core.tag_util*), 278
tag_id (*archivebox.api.v1_core.TagCreateResponseSchema attribute*), 187
tag_id (*archivebox.api.v1_core.TagDeleteResponseSchema attribute*), 189
tag_id (*archivebox.api.v1_core.TagSnapshotRequestSchema attribute*), 189
tag_id (*archivebox.api.v1_core.TagSnapshotResponseSchema attribute*), 189
tag_id (*archivebox.api.v1_core.TagUpdateResponseSchema attribute*), 189
tag_name (*archivebox.api.v1_core.TagCreateResponseSchema attribute*), 187
tag_name (*archivebox.api.v1_core.TagSnapshotRequestSchema attribute*), 189
tag_name (*archivebox.api.v1_core.TagSnapshotResponseSchema attribute*), 189
tag_name (*archivebox.api.v1_core.TagUpdateResponseSchema attribute*), 189
TAG_SEPARATOR_PATTERN (*archivebox.config.common.GeneralConfig attribute*), 340
tag_set() (*archivebox.core.admin_users.CustomUserAdmin method*), 286
TAG_SNAPSHOT_PREVIEW_LIMIT (*in module archivebox.core.tag_util*), 278
tag_snapshots_export() (*in module archivebox.api.v1_core*), 190
TAG_SORT_CHOICES (*in module archivebox.core.tag_util*), 278
tag_urls_export() (*in module archivebox.api.v1_core*), 190
TagAdmin (*class in archivebox.core.admin_tags*), 252
TagAdminForm (*class in archivebox.core.admin_tags*), 251
TagAdminForm.Meta (*class in archivebox.core.admin_tags*), 251
TagAutocompleteSchema (*class in archivebox.api.v1_core*), 187
TagCreateResponseSchema (*class in archivebox.api.v1_core*), 187
TagCreateSchema (*class in archivebox.api.v1_core*), 187
TagDeleteResponseSchema (*class in archivebox.api.v1_core*), 189
TagEditorWidget (*class in archivebox.core.widgets*), 226
TagEditorWidget.Media (*class in archivebox.core.widgets*), 226
TagField (*class in archivebox.core.forms*), 235
TagInline (*class in archivebox.core.admin_tags*), 251
TagNameListFilter (*class in archivebox.core.admin_snapshots*), 280
tags (*archivebox.api.v1_core.SnapshotCreateSchema attribute*), 184
tags (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
tags (*archivebox.api.v1_core.SnapshotUpdateSchema attribute*), 183
tags (*archivebox.api.v1_core.TagAutocompleteSchema attribute*), 187
tags (*archivebox.api.v1_core.TagSearchResponseSchema attribute*), 188
tags (*archivebox.api.v1_crawls.CrawlCreateSchema attribute*), 170

tags (*archivebox.api.v1_crawls.CrawlUpdateSchema attribute*), 170
tags (*archivebox.core.models.Snapshot attribute*), 263
tags_add_to_snapshot() (*in module archivebox.api.v1_core*), 190
tags_autocomplete() (*in module archivebox.api.v1_core*), 190
tags_badges() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
tags_create() (*in module archivebox.api.v1_core*), 190
tags_editor (*archivebox.core.admin_snapshots.SnapshotAdminForm attribute*), 282
tags_editor (*archivebox.crawls.admin.CrawlAdminForm attribute*), 317
tags_inline() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
tags_inline() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
tags_remove_from_snapshot() (*in module archivebox.api.v1_core*), 190
tags_str (*archivebox.api.v1_crawls.CrawlCreateSchema attribute*), 170
tags_str (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
tags_str (*archivebox.api.v1_crawls.CrawlUpdateSchema attribute*), 170
tags_str (*archivebox.crawls.models.Crawl attribute*), 311
tags_str() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
tags_str() (*archivebox.core.models.Snapshot method*), 268
TagSchema (*class in archivebox.api.v1_core*), 186
TagSearchCardSchema (*class in archivebox.api.v1_core*), 188
TagSearchResponseSchema (*class in archivebox.api.v1_core*), 188
TagSearchSnapshotSchema (*class in archivebox.api.v1_core*), 187
TagService (*class in archivebox.services.tag_service*), 366
TagSnapshotRequestSchema (*class in archivebox.api.v1_core*), 189
TagSnapshotResponseSchema (*class in archivebox.api.v1_core*), 189
TagUpdateResponseSchema (*class in archivebox.api.v1_core*), 189
TagUpdateSchema (*class in archivebox.api.v1_core*), 188
TagWidget (*class in archivebox.core.forms*), 235
tail_multiple_worker_logs() (*in module archivebox.workers.supervisord_util*), 211
tail_stderr() (*archivebox.machine.models.Process method*), 109
tail_stdout() (*archivebox.machine.models.Process method*), 108
tell() (*archivebox.misc.serve_static._StreamingQueueWriter method*), 129
template (*archivebox.crawls.models.CrawlSchedule attribute*), 310
TEMPLATE_DIRS (*in module archivebox.core.settings*), 230
template_name (*archivebox.base_models.admin.KeyValueWidget attribute*), 353
template_name (*archivebox.core.views.AddView attribute*), 247
template_name (*archivebox.core.views.PublicIndexView attribute*), 247
template_name (*archivebox.core.widgets.TagEditorWidget attribute*), 226
template_name (*archivebox.core.widgets.URLFiltersWidget attribute*), 227
template_str() (*archivebox.crawls.admin.CrawlScheduleAdmin method*), 320
TEMPLATES (*in module archivebox.core.settings*), 230
TEMPLATES_DIR (*archivebox.config.constants.ConstantsDict attribute*), 327
TEMPLATES_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
TEMPLATES_DIR_NAME (*in module archivebox.core.settings*), 230
TERM_WIDTH (*archivebox.config.common.ShellConfig property*), 339
terminate() (*archivebox.machine.models.Process method*), 110
test_func() (*archivebox.core.views.AddView method*), 247
text_plugins (*archivebox.core.forms.AddLinkForm attribute*), 234
TIME_ZONE (*in module archivebox.core.settings*), 232
TimedProgress (*class in archivebox.misc.logging_util*), 122
timeout (*archivebox.api.v1_cli.RemoveCommandSchema attribute*), 202
TIMEOUT (*archivebox.config.common.ArchivingConfig attribute*), 343

timeout (*archivebox.core.forms.AddLinkForm attribute*), 234
timeout (*archivebox.core.models.ArchiveResult property*), 276
timeout (*archivebox.machine.models.Process attribute*), 104
timeout (*archivebox.plugins.discovery.PluginSpecialConfig attribute*), 373
TIMEOUT_INPUT_PATTERN (*in module archivebox.plugins.forms*), 368
timestamp (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
timestamp (*archivebox.api.v1_core.SnapshotSchema attribute*), 183
timestamp (*archivebox.core.models.Snapshot attribute*), 262
timezone (*archivebox.api.v1_personas.PersonaBrowserSettingsSchema attribute*), 174
TIMEZONE (*archivebox.config.constants.ConstantsDict attribute*), 329
TimezoneMiddleware() (*in module archivebox.core.middleware*), 277
title (*archivebox.api.v1_core.SnapshotCreateSchema attribute*), 184
title (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
title (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
title (*archivebox.api.v1_core.TagSearchSnapshotSchema attribute*), 187
title (*archivebox.core.admin_snapshots.SnapshotArchiveStateListFilter attribute*), 281
title (*archivebox.core.admin_snapshots.SnapshotPermissionsListFilter attribute*), 280
title (*archivebox.core.admin_snapshots.SnapshotResultHtmlListFilter attribute*), 281
title (*archivebox.core.admin_snapshots.SnapshotSizeListFilter attribute*), 281
title (*archivebox.core.admin_snapshots.SnapshotStatusListFilter attribute*), 281
title (*archivebox.core.admin_snapshots.TagNameListFilter attribute*), 280
title (*archivebox.core.models.Snapshot attribute*), 262
title (*archivebox.crawls.admin.MaxDepthListFilter attribute*), 316
title_str() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
title_stripped (*archivebox.core.models.Snapshot property*), 269
TMP_DIR (*archivebox.config.common.StorageConfig attribute*), 340
tmp_dir_candidates() (*in module archivebox.config.paths*), 335
TMP_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 328
tmp_dir_socket_path_is_short_enough() (*in module archivebox.config.paths*), 335
to_csv() (*archivebox.core.models.Snapshot method*), 271
to_csv() (*archivebox.core.models.SnapshotQuerySet method*), 261
to_dict() (*archivebox.core.models.Snapshot method*), 271
to_html() (*archivebox.core.models.SnapshotQuerySet method*), 261
to_json() (*archivebox.core.models.ArchiveResult method*), 274
to_json() (*archivebox.core.models.Snapshot method*), 269
to_json() (*archivebox.core.models.SnapshotQuerySet method*), 261
to_json() (*archivebox.core.models.Tag method*), 260
to_json() (*archivebox.crawls.models.Crawl method*), 313
to_json() (*archivebox.machine.models.Binary method*), 101
to_json() (*archivebox.machine.models.Machine method*), 97
to_json() (*archivebox.machine.models.Process method*), 106
to_json() (*in module archivebox.misc.util*), 138
to_json_str() (*archivebox.core.models.Snapshot method*), 271
to_python() (*archivebox.base_models.admin.HexUUIDConverter method*), 353
to_python() (*archivebox.crawls.admin.URLFiltersField method*), 317
to_python() (*archivebox.uuid_compat.CompactUUIDField method*), 378
to_url() (*archivebox.base_models.admin.HexUUIDConverter method*), 353
token (*archivebox.api.models.APIToken attribute*), 191
token (*archivebox.api.v1_auth.TokenAuthSchema attribute*), 192
token_from_request() (*in module archivebox.api.auth*), 194
token_redacted (*archivebox.api.models.APIToken property*), 191
TokenAuthSchema (*class in archivebox.api.v1_auth*), 192
toml_section_header (*archivebox.config.common.ArchivingConfig attribute*), 342

toml_section_header (*archivebox.config.common.DatabaseConfig attribute*), 342
toml_section_header (*archivebox.config.common.GeneralConfig attribute*), 340
toml_section_header (*archivebox.config.common.SearchBackendConfig attribute*), 344
toml_section_header (*archivebox.config.common.ServerConfig attribute*), 340
toml_section_header (*archivebox.config.common.ShellConfig attribute*), 339
toml_section_header (*archivebox.config.common.StorageConfig attribute*), 340
toml_section_header (*archivebox.config.ldap.LDAPConfig attribute*), 347
tool_annotations() (*in module archivebox.mcp.server*), 297
tool_output_schema() (*in module archivebox.mcp.server*), 297
total_items (*archivebox.api.v1_core.CustomPagination.Output attribute*), 177
total_pages (*archivebox.api.v1_core.CustomPagination.Output attribute*), 178
transaction_on_commit_task_handler() (*in module archivebox.api.webhooks*), 172
TRANSFORMED_HTML_PREVIEW_STYLE (*in module archivebox.misc.serve_static*), 128
truncate_overlong_charfields() (*in module archivebox.misc.db*), 125
truncate_url() (*in module archivebox.misc.logging_util*), 123
ts_to_date_str() (*in module archivebox.misc.util*), 136
TYPE (*archivebox.api.v1_core.ArchiveResultSchema attribute*), 179
TYPE (*archivebox.api.v1_core.MinimalArchiveResultSchema attribute*), 178
TYPE (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
TYPE (*archivebox.api.v1_core.TagSchema attribute*), 186
TYPE (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
TYPE (*archivebox.api.v1_machine.BinarySchema attribute*), 197
TYPE (*archivebox.api.v1_machine.MachineSchema attribute*), 195
TYPE (*archivebox.api.v1_personas.PersonaSchema attribute*), 174
type (*archivebox.base_models.admin.ConfigOption attribute*), 353
TYPE_ARCHIVERESULT (*in module archivebox.misc.jsonl*), 133
TYPE_BINARY (*in module archivebox.misc.jsonl*), 133
TYPE_BINARYREQUEST (*in module archivebox.misc.jsonl*), 133
TYPE_CRAWL (*in module archivebox.misc.jsonl*), 133
TYPE_MACHINE (*in module archivebox.misc.jsonl*), 133
TYPE_PROCESS (*in module archivebox.misc.jsonl*), 133
TYPE_SNAPSHOT (*in module archivebox.misc.jsonl*), 133
TYPE_TAG (*in module archivebox.misc.jsonl*), 133

U

UngroupedSubquery (*class in archivebox.core.models*), 259
unique_together (*archivebox.core.models.SnapshotTag.Meta attribute*), 261
unique_together (*archivebox.machine.models.Binary.Meta attribute*), 101
unique_together (*archivebox.machine.models.NetworkInterface.Meta attribute*), 99
unknown_if_blank() (*in module archivebox.machine.detect*), 94
UNORDERED_OUTPUTS (*in module archivebox.plugins.output_groups*), 371
update (*archivebox.api.v1_cli.AddCommandSchema attribute*), 200
update (*archivebox.api.v1_cli.ScheduleCommandSchema attribute*), 202
UPDATE (*archivebox.machine.models.Process.TypeChoices attribute*), 103
update() (*archivebox.config.configset.BaseConfigSet method*), 333
update() (*archivebox.misc.logging.AttrDict method*), 121
update() (*in module archivebox.cli.archivebox_update*), 146
update_and_requeue() (*archivebox.core.models.Snapshot method*), 264
update_and_requeue() (*archivebox.machine.models.Process method*), 106
update_and_requeue() (*archivebox.workers.models.ModelWithQueue method*), 214
update_archiveresults() (*in module archivebox.cli.archivebox_archiveresult*), 154
update_binaries() (*in module archivebox.cli.archivebox_binary*), 144

update_child_snapshot_permissions() (*archivebox.crawls.models.Crawl method*), 313
update_cmd() (*in module archivebox.cli.archivebox_archiveresult*), 155
update_cmd() (*in module archivebox.cli.archivebox_binary*), 145
update_cmd() (*in module archivebox.cli.archivebox_crawl*), 140
update_cmd() (*in module archivebox.cli.archivebox_persona*), 150
update_cmd() (*in module archivebox.cli.archivebox_snapshot*), 161
update_cmd() (*in module archivebox.cli.archivebox_tag*), 165
update_crawl_permissions() (*archivebox.crawls.admin.CrawlAdmin method*), 319
update_crawls() (*in module archivebox.cli.archivebox_crawl*), 140
update_output_metadata_from_filesystem() (*archivebox.core.models.ArchiveResult method*), 275
update_personas() (*in module archivebox.cli.archivebox_persona*), 150
update_snapshot_permissions() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
update_snapshots() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
update_snapshots() (*in module archivebox.cli.archivebox_snapshot*), 160
update_tags() (*in module archivebox.cli.archivebox_tag*), 165
UpdateCommandSchema (*class in archivebox.api.v1_cli*), 201
url (*archivebox.api.v1_core.SnapshotCreateSchema attribute*), 183
url (*archivebox.api.v1_core.SnapshotFilterSchema attribute*), 185
url (*archivebox.api.v1_core.SnapshotSchema attribute*), 182
url (*archivebox.api.v1_core.TagSearchSnapshotSchema attribute*), 187
url (*archivebox.core.admin_snapshots.SnapshotAdminForm attribute*), 282
url (*archivebox.core.forms.AddLinkForm attribute*), 234
url (*archivebox.core.models.Snapshot attribute*), 262
url (*archivebox.machine.models.Process attribute*), 105
url() (*archivebox.core.models.ArchiveResult method*), 275
url__icontains (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 201
url__istartswith (*archivebox.api.v1_cli.SnapshotFilterCommandSchema attribute*), 201
URL_ALLOWLIST (*archivebox.config.common.ArchivingConfig attribute*), 343
URL_ALLOWLIST_PTN (*archivebox.config.common.ArchivingConfig property*), 344
URL_DENYLIST (*archivebox.config.common.ArchivingConfig attribute*), 343
URL_DENYLIST_PTN (*archivebox.config.common.ArchivingConfig property*), 344
url_favicon() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
url_filters (*archivebox.core.forms.AddLinkForm attribute*), 234
url_filters (*archivebox.crawls.admin.CrawlAdminForm attribute*), 317
url_hash() (*archivebox.core.models.Snapshot method*), 270
url_passes_compiled_filters() (*archivebox.crawls.models.Crawl method*), 313
url_passes_filters() (*archivebox.crawls.models.Crawl method*), 313
url_path() (*archivebox.core.models.Snapshot method*), 269
URL_PREFIX_SEARCH_LIMIT (*in module archivebox.search.views*), 290
url_prefix_upper_bound() (*in module archivebox.search.views*), 290
url_replace() (*in module archivebox.core.templatetags.core_tags*), 220
url_str() (*archivebox.core.admin_snapshots.SnapshotAdmin method*), 284
urldecode (*in module archivebox.misc.util*), 136
urlencode (*in module archivebox.misc.util*), 136
URLFiltersField (*class in archivebox.crawls.admin*), 316
URLFiltersWidget (*class in archivebox.core.widgets*), 227
urlpatterns (*in module archivebox.api.urls*), 171
urlpatterns (*in module archivebox.core.urls*), 248
urls (*archivebox.api.v1_cli.AddCommandSchema attribute*), 199
urls (*archivebox.api.v1_crawls.CrawlCreateSchema attribute*), 170
urls (*archivebox.api.v1_crawls.CrawlSchema attribute*), 169
urls (*archivebox.crawls.models.Crawl attribute*), 311

urls (*in module archivebox.api.v1_api*), 173
urls_editor() (*archivebox.crawls.admin.CrawlAdmin method*), 320
urls_preview() (*archivebox.crawls.admin.CrawlAdmin method*), 319
USE_COLOR (*archivebox.config.common.ShellConfig attribute*), 339
USE_I18N (*in module archivebox.core.settings*), 232
USE_TZ (*in module archivebox.core.settings*), 232
USER (*in module archivebox.config.permissions*), 324
user_agent (*archivebox.api.v1_personas.PersonaBrowserSettingsSchema attribute*), 174
USER_AGENT (*archivebox.config.common.ArchivingConfig attribute*), 343
user_agent_imported (*archivebox.personas.importers.PersonaImportResult attribute*), 300
user_data_dir (*archivebox.personas.importers.PersonaImportSource attribute*), 300
USER_PLUGINS_DIR (*archivebox.config.constants.ConstantsDict attribute*), 328
USER_PLUGINS_DIR (*in module archivebox.plugins.discovery*), 373
username (*archivebox.api.v1_auth.PasswordAuthSchema attribute*), 192
USERS_DIR (*archivebox.config.constants.ConstantsDict attribute*), 326
USERS_DIR (*in module archivebox.config.paths*), 334
USERS_DIR_NAME (*archivebox.config.constants.ConstantsDict attribute*), 327
USES_SUBDOMAIN_ROUTING (*archivebox.config.common.ServerConfig property*), 341
uuid7() (*in module archivebox.uuid_compat*), 378

V

VALID_TYPES (*in module archivebox.misc.jsonl*), 133
validate_check_ssl_validity() (*archivebox.config.common.ArchivingConfig method*), 343
validate_delete_after() (*archivebox.config.common.ArchivingConfig class method*), 344
validate_ldap_config() (*archivebox.config.ldap.LDAPConfig method*), 347
validate_name() (*archivebox.personas.models.Persona method*), 306
validate_output_paths_for_delete() (*archivebox.base_models.models.ModelWithOutputDir class method*), 352
validate_permissions() (*archivebox.config.common.ArchivingConfig class method*), 344
validate_persona_name() (*in module archivebox.cli.archivebox_persona*), 149
validate_persona_name() (*in module archivebox.personas.importers*), 301
validate_schedule() (*in module archivebox.crawls.schedule_util*), 309
validate_server_security_mode() (*archivebox.config.common.ServerConfig method*), 341
validate_url() (*in module archivebox.misc.util*), 137
validate_url_for_archiving() (*archivebox.core.models.Snapshot method*), 265
validate_url_length() (*in module archivebox.misc.util*), 137
value_from_datadict() (*archivebox.base_models.admin.KeyValueWidget method*), 354
value_from_datadict() (*archivebox.core.widgets.URLFiltersWidget method*), 227
values() (*archivebox.config.configset.BaseConfigSet method*), 333
verbose_name (*archivebox.api.models.APIToken.Meta attribute*), 191
verbose_name (*archivebox.api.models.OutboundWebhook.Meta attribute*), 191
verbose_name (*archivebox.core.admin_archiveresults.ArchiveResultAdmin.Meta attribute*), 225
verbose_name (*archivebox.core.models.ArchiveResult.Meta attribute*), 274
verbose_name (*archivebox.core.models.Snapshot.Meta attribute*), 264
verbose_name (*archivebox.core.models.Tag.Meta attribute*), 260
verbose_name (*archivebox.crawls.models.Crawl.Meta attribute*), 312
verbose_name (*archivebox.crawls.models.CrawlSchedule.Meta attribute*), 310
verbose_name (*archivebox.ldap.apps.LDAPConfig attribute*), 118
verbose_name (*archivebox.machine.apps.MachineConfig attribute*), 93
verbose_name (*archivebox.machine.models.Binary.Meta attribute*), 101
verbose_name (*archive-*

box.machine.models.Process.Meta attribute), 105
verbose_name (*archivebox.mcp.apps.MCPConfig attribute*), 294
verbose_name (*archivebox.plugins.apps.PluginsConfig attribute*), 370
verbose_name (*archivebox.search.apps.SearchConfig attribute*), 288
verbose_name_plural (*archivebox.api.models.APIToken.Meta attribute*), 191
verbose_name_plural (*archivebox.core.admin_archiveresults.ArchiveResultAdmin.Meta attribute*), 225
verbose_name_plural (*archivebox.core.models.ArchiveResult.Meta attribute*), 274
verbose_name_plural (*archivebox.core.models.Snapshot.Meta attribute*), 264
verbose_name_plural (*archivebox.core.models.Tag.Meta attribute*), 260
verbose_name_plural (*archivebox.crawls.models.Crawl.Meta attribute*), 312
verbose_name_plural (*archivebox.crawls.models.CrawlSchedule.Meta attribute*), 310
verbose_name_plural (*archivebox.machine.models.Binary.Meta attribute*), 101
verbose_name_plural (*archivebox.machine.models.Process.Meta attribute*), 105
version (*archivebox.api.v1_machine.BinaryFilterSchema attribute*), 198
version (*archivebox.api.v1_machine.BinarySchema attribute*), 197
VERSION (*archivebox.config.constants.ConstantsDict attribute*), 326
version (*archivebox.machine.models.Binary attribute*), 100
VERSION (*in module archivebox.config.version*), 348
version() (*archivebox.core.admin_archiveresults.ArchiveResultInline method*), 224
version() (*in module archivebox.cli.archivebox_version*), 151
viewport_device_scale_factor (*archivebox.api.v1_personas.PersonaBrowserSettingsSchema attribute*), 174
viewport_size (*archivebox.api.v1_personas.PersonaBrowserSettingsSchema attribute*), 174
VOLATILE_PROFILE_COPY_PATTERNS (*in module archivebox.personas.importers*), 299
VOLATILE_PROFILE_DIR_NAMES (*in module archivebox.personas.models*), 305
VOLATILE_PROFILE_FILE_NAMES (*in module archivebox.personas.models*), 305

W

wait() (*archivebox.machine.models.Process method*), 109
wait_for_snapshot_tasks() (*archivebox.services.runner.CrawlRunner method*), 363
wait_for_supervisord_ready() (*in module archivebox.workers.supervisord_util*), 210
wait_popen_and_kill_children() (*in module archivebox.core.shutdown_util*), 250
wait_psutil_and_kill_children() (*in module archivebox.core.shutdown_util*), 250
warn_if_invalid() (*archivebox.config.common.ArchivingConfig method*), 343
warn_on_save_outside_runner (*archivebox.machine.models.Binary attribute*), 101
warn_on_save_outside_runner (*archivebox.workers.models.ModelWithQueue attribute*), 213
warning_error_handler() (*in module archivebox.api.webhooks*), 172
warnings (*archivebox.personas.importers.PersonaImportResult attribute*), 301
watch_for_cancelled_crawl() (*archivebox.services.runner.CrawlRunner method*), 362
web_base_url() (*in module archivebox.core.templatetags.core_tags*), 220
WebAddView (*class in archivebox.core.views*), 248
widget (*archivebox.core.forms.TagField attribute*), 235
widget (*archivebox.crawls.admin.URLFiltersField attribute*), 316
widgets (*archivebox.core.admin_tags.TagAdminForm.Meta attribute*), 252
widgets (*archivebox.crawls.admin.CrawlAdminForm.Meta attribute*), 317
with_headers (*archivebox.api.v1_cli.ListCommandSchema attribute*), 202
without_fragment (*in module archivebox.misc.util*), 136
without_path (*in module archivebox.misc.util*), 136
without_query (*in module archivebox.misc.util*), 136
without_scheme (*in module archivebox.misc.util*), 136
WORKER (*archivebox.machine.models.Process.TypeChoices attribute*), 103

worker_detail_view() (*in module archivebox.config.views*), 331
worker_list_view() (*in module archivebox.config.views*), 331
worker_runtime_component() (*in module archivebox.workers.supervisord_util*), 210
worker_type (*archivebox.machine.models.Process attribute*), 104
WORKERS_DIR_NAME (*in module archivebox.workers.supervisord_util*), 208
WorkersConfig (*class in archivebox.workers.apps*), 205
writable() (*archivebox.misc.serve_static._StreamingQueueWriter method*), 129
write() (*archivebox.misc.serve_static._StreamingQueueWriter method*), 129
write_config_file() (*in module archivebox.config.collection*), 323
write_entry() (*archivebox.misc.monkey_patches.ModifiedAccessLogGenerator method*), 134
write_html_details() (*archivebox.core.models.Snapshot method*), 271
write_index_json() (*archivebox.core.models.Snapshot method*), 268
write_index_jsonl() (*archivebox.core.models.Snapshot method*), 268
write_json_details() (*archivebox.core.models.Snapshot method*), 271
write_record() (*in module archivebox.misc.jsonl*), 133
write_records() (*in module archivebox.misc.jsonl*), 133
WSGI_APPLICATION (*in module archivebox.core.settings*), 229

Y

year (*archivebox.api.v1_core.TagSearchResponseSchema attribute*), 188

Z

zip_link() (*archivebox.core.admin_archiveresults.ArchiveResultAdmin method*), 225
ZRAM_DEVICE_NAME (*in module archivebox.services.resource_admission*), 360
zram_swap_identifiers() (*in module archivebox.services.resource_admission*), 360