


Vidlička youtube-dl s dalšími funkcemi a opravami


 github.com/yt-dlp/yt-dlp





YT-DLP *A youtube-dl fork with additional features and fixes*

DOWNLOAD **V2023.07.06**

 **PYPI**

 **DONATE**

 **152 USERS**

 **1820 ONLINE**

SUPPORTED SITES

UNLICENSE

TESTS **PASSING**

COMMITTS **22 / MONTH**

LAST THURSDAY

yt-dlp is a youtube-dl fork based on the now inactive youtube-dlc. The main focus of this project is adding new features and patches while also keeping up to date with the original project

- EXTRACTOR ARGUMENTS
- EMBEDDING YT-DLP
Embedding examples
- DEPRECATED OPTIONS

- [WIKI](#)
[FAQ](#)

NEW FEATURES

- Forked from [yt-dlc@f9401f2](#) and merged with [youtube-dl@42f2d4](#) ([exceptions](#))
- **[SponsorBlock Integration](#)**: You can mark/remove sponsor sections in YouTube videos by utilizing the [SponsorBlock](#) API
- **[Format Sorting](#)**: The default format sorting options have been changed so that higher resolution and better codecs will be now preferred instead of simply using larger bitrate. Furthermore, you can now specify the sort order using `-S`. This allows for much easier format selection than what is possible by simply using `--format` ([examples](#))
- **Merged with [animelover1984/youtube-dl](#)**: You get most of the features and improvements from [animelover1984/youtube-dl](#) including `--write-comments`, [BiliBiliSearch](#), [BilibiliChannel](#), Embedding thumbnail in mp4/ogg/opus, playlist infojson etc. Note that NicoNico livestreams are not available. See [#31](#) for details.

- **YouTube improvements:**
 - Supports Clips, Stories (`ytstories:<channel UCID>`), Search (including filters)*, YouTube Music Search, Channel-specific search, Search prefixes (`ytsearch:`, `ytsearchdate:`)*, Mixes, and Feeds (`:ytfav`, `:ytwatchlater`, `:ytsubs`, `:ythistory`, `:ytrec`, `:ytnotif`)
 - Fix for n-sig based throttling *
 - Supports some (but not all) age-gated content without cookies
 - Download livestreams from the start using `--live-from-start` (*experimental*)
 - `255kbps` audio is extracted (if available) from YouTube Music when premium cookies are given
 - Channel URLs download all uploads of the channel, including shorts and live
- **Cookies from browser:** Cookies can be automatically extracted from all major web browsers using `--cookies-from-browser BROWSER[+KEYRING][:PROFILE][:CONTAINER]`
- **Download time range:** Videos can be downloaded partially based on either timestamps or chapters using `--download-sections`
- **Split video by chapters:** Videos can be split into multiple files based on chapters using `--split-chapters`
- **Multi-threaded fragment downloads:** Download multiple fragments of m3u8/mpd videos in parallel. Use `--concurrent-fragments (-N)` option to set the number of threads used
- **Aria2c with HLS/DASH:** You can use `aria2c` as the external downloader for DASH(mpd) and HLS(m3u8) formats
- **New and fixed extractors:** Many new extractors have been added and a lot of existing ones have been fixed. See the [changelog](#) or the [list of supported sites](#)

- **New MSOs:** Philo, Spectrum, SlingTV, Cablevision, RCN etc.
- **Subtitle extraction from manifests:** Subtitles can be extracted from streaming media manifests. See [commit/be6202f](#) for details
- **Multiple paths and output templates:** You can give different [output templates](#) and download paths for different types of files. You can also set a temporary path where intermediary files are downloaded to using `--paths (-P)`
- **Portable Configuration:** Configuration files are automatically loaded from the home and root directories. See [CONFIGURATION](#) for details
- **Output template improvements:** Output templates can now have date-time formatting, numeric offsets, object traversal etc. See [output template](#) for details. Even more advanced operations can also be done with the help of `--parse-metadata` and `--replace-in-metadata`
- **Other new options:** Many new options have been added such as `--alias`, `--print`, `--concat-playlist`, `--wait-for-video`, `--retry-sleep`, `--sleep-requests`, `--convert-thumbnails`, `--force-download-archive`, `--force-overwrites`, `--break-match-filter` etc
- **Improvements:** Regex and other operators in `--format/--match-filter`, multiple `--postprocessor-args` and `--downloader-args`, faster archive checking, more [format selection options](#), merge multi-video/audio, multiple `--config-locations`, `--exec` at different stages, etc
- **Plugins:** Extractors and PostProcessors can be loaded from an external file. See [plugins](#) for details
- **Self updater:** The releases can be updated using `yt-dlp -U`, and downgraded using `--update-to` if required

- **Nightly builds:** Automated nightly builds can be used with `--update-to nightly`

See changelog or commits for the full list of changes

Features marked with a * have been back-ported to youtube-dl

Differences in default behavior

Some of yt-dlp's default options are different from that of youtube-dl and youtube-dlc:

- yt-dlp supports only Python 3.7+, and *may* remove support for more versions as they become EOL; while youtube-dl still supports Python 2.6+ and 3.2+
- The options `--auto-number (-A)`, `--title (-t)` and `--literal (-l)`, no longer work. See removed options for details
- `avconv` is not supported as an alternative to `ffmpeg`
- yt-dlp stores config files in slightly different locations to youtube-dl. See CONFIGURATION for a list of correct locations
- The default output template is `%(title)s [%(id)s].%(ext)s`. There is no real reason for this change. This was changed before yt-dlp was ever made public and now there are no plans to change it back to `%(title)s-%(id)s.%(ext)s`. Instead, you may use `--compat-options filename`
- The default format sorting is different from youtube-dl and prefers higher resolution and better codecs rather than higher bitrates. You can use the `--format-sort` option to change this to any order you prefer, or use `--compat-options format-sort` to use youtube-dl's sorting order
- The default format selector is `bv*+ba/b`. This means that if a combined video + audio format that is better than the best video-only format is found, the former will be preferred. Use `-f bv+ba/b` or `--compat-options format-spec` to revert this

- Unlike youtube-dl, yt-dlp does not allow merging multiple audio/video streams into one file by default (since this conflicts with the use of `-f bv*+ba`). If needed, this feature must be enabled using `--audio-multistreams` and `--video-multistreams`. You can also use `--compat-options multistreams` to enable both
- `--no-abort-on-error` is enabled by default. Use `--abort-on-error` or `--compat-options abort-on-error` to abort on errors instead
- When writing metadata files such as thumbnails, description or infojson, the same information (if available) is also written for playlists. Use `--no-write-playlist-metafiles` or `--compat-options no-playlist-metafiles` to not write these files
- `--add-metadata` attaches the `infojson` to `mkv` files in addition to writing the metadata when used with `--write-info-json`. Use `--no-embed-info-json` or `--compat-options no-attach-info-json` to revert this
- Some metadata are embedded into different fields when using `--add-metadata` as compared to youtube-dl. Most notably, `comment` field contains the `webpage_url` and `synopsis` contains the `description`. You can use `--parse-metadata` to modify this to your liking or use `--compat-options embed-metadata` to revert this
- `playlist_index` behaves differently when used with options like `--playlist-reverse` and `--playlist-items`. See [#302](#) for details. You can use `--compat-options playlist-index` if you want to keep the earlier behavior
- The output of `-F` is listed in a new format. Use `--compat-options list-formats` to revert this
- Live chats (if available) are considered as subtitles. Use `--sublangs all, -live_chat` to download all subtitles except live chat. You can also use `--compat-options no-live-chat` to prevent any live chat/danmaku from downloading

- YouTube channel URLs download all uploads of the channel. To download only the videos in a specific tab, pass the tab's URL. If the channel does not show the requested tab, an error will be raised. Also, `/live` URLs raise an error if there are no live videos instead of silently downloading the entire channel. You may use `--compat-options no-youtube-channel-redirect` to revert all these redirections
- Unavailable videos are also listed for YouTube playlists. Use `--compat-options no-youtube-unavailable-videos` to remove this
- The upload dates extracted from YouTube are in UTC when available. Use `--compat-options no-youtube-prefer-utc-upload-date` to prefer the non-UTC upload date.
- If `ffmpeg` is used as the downloader, the downloading and merging of formats happen in a single step when possible. Use `--compat-options no-direct-merge` to revert this
- Thumbnail embedding in `mp4` is done with `mutagen` if possible. Use `--compat-options embed-thumbnail-atomicparsley` to force the use of `AtomicParsley` instead
- Some internal metadata such as filenames are removed by default from the `infojson`. Use `--no-clean-infojson` or `--compat-options no-clean-infojson` to revert this
- When `--embed-subs` and `--write-subs` are used together, the subtitles are written to disk and also embedded in the media file. You can use just `--embed-subs` to embed the subs and automatically delete the separate file. See [#630 \(comment\)](#) for more info. `--compat-options no-keep-subs` can be used to revert this
- `certifi` will be used for SSL root certificates, if installed. If you want to use system certificates (e.g. self-signed), use `--compat-options no-certifi`

- Dezinfekce neplatných znaků v názvech souborů yt-dlp je jiná/chytřejší než u youtube-dl. Můžete použít `--compat-options filename-sanitization` k návratu k chování youtube-dl
- yt-dlp se pokusí analyzovat výstupy externího stahování do standardního výstupu, pokud je to možné (aktuálně implementováno: [aria2e](#)). Můžete použít `--compat-options no-external-downloader-progress` k získání výstupu stahování tak, jak je
- Verze yt-dlp mezi 2021.09.01 a 2023.01.02 se vztahují `--match-filter` na vnořené seznamy skladeb. Toto byl neúmyslný vedlejší účinek [8f18ac](#) a je opraven v [d7b460](#) . Použijte `--compat-options playlist-match-filter` k tomu, abyste to vrátili

Pro snadné použití je k dispozici několik dalších možností kompatibility:

- `--compat-options all`: Použijte všechny možnosti kompatibility (NEPOUŽÍVEJTE)
- `--compat-options youtube-dl`: Stejný jako `--compat-options all, -multistreams, -playlist-match-filter`
- `--compat-options youtube-dlc`: Stejný jako `--compat-options all, -no-live-chat, -no-youtube-channel-redirect, -playlist-match-filter`
- `--compat-options 2021`: Stejný jako `--compat-options 2022, no-certifi, filename-sanitization, no-youtube-prefer-utc-upload-date`
- `--compat-options 2022`: Stejný jako `--compat-options playlist-match-filter, no-external-downloader-progress`. Použijte toto k povolení všech budoucích možností kompatibility

INSTALACE

LINUX/BSD

MACOS

PYPI

SOURCE TAR

OTHER

ALL VERSIONS

Yt-dlp můžete nainstalovat pomocí binárních souborů , pip nebo pomocí správce balíčků třetí strany. Podrobné pokyny najdete na wiki

AKTUALIZACE

Můžete použít `yt-dlp -U` k aktualizaci, pokud používáte binární soubory vydání

Pokud jste nainstalovali pomocí pip , jednoduše znovu spusťte stejný příkaz, který byl použit k instalaci programu

Další správci balíčků třetích stran naleznete na wiki nebo v jejich dokumentaci

V současné době existují dva kanály vydání pro binární soubory `stable` a `nightly`. `stable` je výchozí kanál a mnoho jeho změn bylo testováno uživateli nočního kanálu. Kanál `nightly` má vydání vytvořená po každém odeslání do hlavní větve a bude mít nejnovější opravy a doplňky, ale také má větší riziko regresí. Jsou k dispozici ve vlastním repo .

Při použití `--update/ -U` binární soubor aktualizuje pouze na svůj aktuální kanál. `--update-to CHANNEL` lze použít k přepnutí na jiný kanál, když je k dispozici novější verze. `--update-to [CHANNEL@]TAG` lze také použít k upgradu nebo downgradu na konkrétní značky z kanálu.

Můžete také použít `--update-to <repository>(<owner>/<repository>)` k aktualizaci na kanál ve zcela jiném úložišti. Buďte opatrní s tím, na jaké úložiště aktualizujete, pro binární soubory z různých úložišť se neprovádí žádné ověření.

Příklad použití:

- `yt-dlp --update-to nightly` změnit `nightly` kanál a aktualizovat na jeho nejnovější verzi
- `yt-dlp --update-to stable@2023.02.17` upgrade/downgrade pro vydání na `stable` značku kanálu `2023.02.17`
- `yt-dlp --update-to 2023.01.06` upgrade/downgrade na tag, `2023.01.06` pokud na aktuálním kanálu existuje
- `yt-dlp --update-to example/yt-dlp@2023.03.01` upgrade/downgrade na vydání z `example/yt-dlp` úložiště, tag `2023.03.01`

UVOLNĚNÍ SOUBORŮ

Doporučeno

Soubor	Popis
<code>yt-dlp</code>	<u>Binární soubor zipimport</u> nezávislý na platformě . Vyžaduje Python (doporučeno pro Linux/BSD)
<code>yt-dlp.exe</code>	Windows (Win7 SP1+) samostatný binární x64 (doporučeno pro Windows)
<code>yt-dlp_macos</code>	Universal MacOS (10.15+) samostatný spustitelný soubor (doporučeno pro MacOS)

Alternativy

Soubor	Popis
<code>yt-dlp_x86.exe</code>	Windows (Vista SP2+) samostatný x86 (32bitový) binární soubor
<code>yt-dlp_min.exe</code>	Windows (Win7 SP1+) samostatný binární x64 postavený s <code>py2exe</code> (<u>nedoporučuje se</u>)
<code>yt-dlp_linux</code>	Samostatný Linux x64 binární

Soubor	Popis
yt-dlp_linux.zip	Rozbalený spustitelný soubor Linux (bez automatické aktualizace)
yt-dlp_linux_armv7l	Samostatný Linux armv7l (32bitový) binární
yt-dlp_linux_aarch64	Samostatný Linux aarch64 (64bitový) binární
yt-dlp_win.zip	Rozbalený spustitelný soubor Windows (bez automatické aktualizace)
yt-dlp_macos.zip	Rozbalený spustitelný soubor MacOS (10.15+) (bez automatické aktualizace)
yt-dlp_macos_legacy	MacOS (10.9+) samostatný spustitelný x64

Různé

Soubor	Popis
yt-dlp.tar.gz	Zdrojový tarball
SHA2-512SUMS	Součty SHA512 ve stylu GNU
SHA2-512SUMS.sig	Soubor podpisu GPG pro částky SHA512
SHA2-256SUMS	GNU-style SHA256 sums
SHA2-256SUMS.sig	GPG signature file for SHA256 sums

The public key that can be used to verify the GPG signatures is [available here](#) Example usage:

```
curl -L https://github.com/yt-dlp/yt-dlp/raw/master/public.key
| gpg --import
gpg --verify SHA2-256SUMS.sig SHA2-256SUMS
gpg --verify SHA2-512SUMS.sig SHA2-512SUMS
```

Note: The manpages, shell completion (autocomplete) files etc. are available inside the [source tarball](#)

DEPENDENCIES

Python versions 3.7+ (CPython and PyPy) are supported. Other versions and implementations may or may not work correctly.

While all the other dependencies are optional, `ffmpeg` and `ffprobe` are highly recommended

Strongly recommended

ffmpeg and **ffprobe** - Required for merging separate video and audio files as well as for various post-processing tasks. License depends on the build

There are bugs in ffmpeg that causes various issues when used alongside yt-dlp. Since ffmpeg is such an important dependency, we provide custom builds with patches for some of these issues at yt-dlp/FFmpeg-Builds. See the readme for details on the specific issues solved by these builds

Important: What you need is ffmpeg *binary*, **NOT** the python package of the same name

Networking

- **certifi*** - Provides Mozilla's root certificate bundle. Licensed under MPLv2
- **brotili*** or **brotilicffi** - Brotli content encoding support. Both licensed under MIT ¹₂
- **websockets*** - For downloading over websocket. Licensed under BSD-3-Clause

Metadata

- **mutagen*** - For `--embed-thumbnail` in certain formats. Licensed under GPLv2+
- **AtomicParsley** - For `--embed-thumbnail` in `mp4/m4a` files when `mutagen/ffmpeg` cannot. Licensed under GPLv2+
- **xattr**, **pyxattr** or **setfattr** - For writing xattr metadata (`--xattr`) on **Linux**. Licensed under MIT, LGPL2.1 and GPLv2+ respectively

Misc

- **pycryptodomex*** - For decrypting AES-128 HLS streams and various other data. Licensed under BSD-2-Clause
- **phantomjs** - Used in extractors where javascript needs to be run. Licensed under BSD-3-Clause
- **secretstorage** - For `--cookies-from-browser` to access the **Gnome** keyring while decrypting cookies of **Chromium**-based browsers on **Linux**. Licensed under BSD-3-Clause
- Any external downloader that you want to use with `--downloader`

Deprecated

- **avconv** and **avprobe** - Now **deprecated** alternative to ffmpeg. License depends on the build
- **sponskrub** - For using the now **deprecated** sponskrub options. Licensed under GPLv3+
- **rtmpdump** - For downloading `rtmp` streams. ffmpeg can be used instead with `--downloader ffmpeg`. Licensed under GPLv2+
- **mplayer** or **mpv** - For downloading `rtsp/mms` streams. ffmpeg can be used instead with `--downloader ffmpeg`. Licensed under GPLv2+

To use or redistribute the dependencies, you must agree to their respective licensing terms.

The standalone release binaries are built with the Python interpreter and the packages marked with * included.

If you do not have the necessary dependencies for a task you are attempting, yt-dlp will warn you. All the currently available dependencies are visible at the top of the `--verbose` output

COMPILE

Standalone PyInstaller Builds

To build the standalone executable, you must have Python and `pyinstaller` (plus any of yt-dlp's optional dependencies if needed). Once you have all the necessary dependencies installed, simply run `pyinst.py`. The executable will be built for the same architecture (x86/ARM, 32/64 bit) as the Python used.

```
python3 -m pip install -U pyinstaller -r requirements.txt
python3 devscripts/make_lazy_extractors.py
python3 pyinst.py
```

On some systems, you may need to use `py` or `python` instead of `python3`.

`pyinst.py` accepts any arguments that can be passed to `pyinstaller`, such as `--onefile/-F` or `--onedir/-D`, which is further documented here.

Note: Pyinstaller versions below 4.4 do not support Python installed from the Windows store without using a virtual environment.

Important: Running `pyinstaller` directly **without** using `pyinst.py` is **not** officially supported. This may or may not work correctly.

Platform-independent Binary (UNIX)

You will need the build tools `python` (3.7+), `zip`, `make` (GNU), `pandoc`* and `pytest`*.

After installing these, simply run `make`.

You can also run `make yt-dlp` instead to compile only the binary without updating any of the additional files. (The build tools marked with * are not needed for this)

Standalone Py2Exe Builds (Windows)

While we provide the option to build with `py2exe`, it is recommended to build using PyInstaller instead since the py2exe builds **cannot contain `pycryptodomex/certifi` and needs VC++14** on the target

computer to run.

If you wish to build it anyway, install Python and py2exe, and then simply run `setup.py py2exe`

```
py -m pip install -U py2exe -r requirements.txt
py devscripts/make_lazy_extractors.py
py setup.py py2exe
```

Related scripts

- [devscripts/update-version.py](#) - Update the version number based on current date.
- [devscripts/set-variant.py](#) - Set the build variant of the executable.
- [devscripts/make_changelog.py](#) - Create a markdown changelog using short commit messages and update `CONTRIBUTORS` file.
- [devscripts/make_lazy_extractors.py](#) - Create lazy extractors. Running this before building the binaries (any variant) will improve their startup performance. Set the environment variable `YTDLP_NO_LAZY_EXTRACTORS=1` if you wish to forcefully disable lazy extractor loading.

Note: See their `--help` for more info.

Forking the project

If you fork the project on GitHub, you can run your fork's [build workflow](#) to automatically build the selected version(s) as artifacts. Alternatively, you can run the [release workflow](#) or enable the [nightly workflow](#) to create full (pre-)releases.

USAGE AND OPTIONS

```
yt-dlp [OPTIONS] [--] URL [URL...]
```

`Ctrl+F` is your friend :D

General Options:

-h, --help	Print this help text and exit
--version	Print program version and exit
-U, --update	Update this program to the
latest version	
--no-update	Do not check for updates
(default)	
--update-to [CHANNEL]@[TAG]	Upgrade/downgrade to a specific
version.	
well. CHANNEL	CHANNEL can be a repository as
"latest"	and TAG default to "stable" and
"UPDATE" for	respectively if omitted; See
stable, nightly	details. Supported channels:
-i, --ignore-errors	Ignore download and
postprocessing errors.	
successful	The download will be considered
fails	even if the postprocessing
--no-abort-on-error	Continue with next video on
download errors;	e.g. to skip unavailable videos
in a	playlist (default)
--abort-on-error	Abort downloading of further
videos if an	error occurs (Alias: --no-
ignore-errors)	Display the current user-agent
--dump-user-agent	
and exit	List all supported extractors
--list-extractors	
and exit	Output descriptions of all
--extractor-descriptions	
supported	extractors and exit
--use-extractors NAMES	Extractor names to use
separated by commas.	
"all", "default"	You can also use regexes,
	and "end" (end URL matching);

e.g. `--ies`

the name

`--ies`

extractors for

(Alias: `--ies`)

`--default-search PREFIX`
URLs. E.g.

two videos from

term "python".

`dlp` guess

warning when

an error. The

repairs broken

this is not

`--ignore-config`
configuration files

locations.

this option

configuration

not loaded.

`--no-config-locations`
configuration files

configuration

`config-locations`

"holodex.*,end,youtube". Prefix

with a "-" to exclude it, e.g.

default,-generic. Use `--list-`

a list of extractor names.

Use this prefix for unqualified

"gvsearch2:python" downloads

google videos for the search

Use the value "auto" to let yt-

("auto_warning" to emit a

guessing). "error" just throws

default value "fixup_error"

URLs, but emits an error if

possible instead of searching

Don't load any more

except those given by `--config-`

For backward compatibility, if

is found inside the system

file, the user configuration is

(Alias: `--no-config`)

Do not load any custom

(default). When given inside a

file, ignore all previous `--`

<p>--config-locations PATH configuration file;</p> <p>or its</p> <p>stdin). Can be</p> <p>other</p> <p>--flat-playlist playlist,</p> <p>--no-flat-playlist playlist</p> <p>--live-from-start start.</p> <p>YouTube</p> <p>--no-live-from-start current time</p> <p>--wait-for-video MIN[-MAX] become</p> <p>number of</p> <p>between retries</p> <p>--no-wait-for-video streams (default)</p> <p>--mark-watched --simulate)</p> <p>--no-mark-watched (default)</p> <p>--color [STREAM:]POLICY output,</p> <p>STREAM (stdout or</p> <p>to. Can be one</p>	<p>defined in the current file Location of the main</p> <p>either the path to the config containing directory ("- " for</p> <p>used multiple times and inside configuration files</p> <p>Do not extract the videos of a only list them</p> <p>Fully extract the videos of a (default)</p> <p>Download livestreams from the Currently only supported for</p> <p>(Experimental)</p> <p>Download livestreams from the (default)</p> <p>Wait for scheduled streams to available. Pass the minimum</p> <p>seconds (or range) to wait</p> <p>Do not wait for scheduled</p> <p>Mark videos watched (even with</p> <p>Do not mark videos watched</p> <p>Whether to emit color codes in optionally prefixed by the</p> <p>stderr) to apply the setting of "always", "auto" (default),</p>
--	--

"never", or
terminal
multiple times
--compat-options OPTS
compatibility
some of the
"Differences in
--alias ALIASES OPTIONS
string. Unless
", it is
are parsed
formatting
get-audio, -X
format {0}"
and "-X" that
expands to
format ARG0".
in the --help
trigger more
defining
measure, each
maximum of 100
multiple times

"no_color" (use non color
sequences). Can be used
Options that can help keep
with youtube-dl or youtube-dlc
configurations by reverting
changes made in yt-dlp. See
default behavior" for details
Create aliases for an option
an alias starts with a dash "-
prefixed with "--". Arguments
according to the Python string
mini-language. E.g. --alias
"-S=aext:{0},abr -x --audio-
creates options "--get-audio"
takes an argument (ARG0) and
"-S=aext:ARG0,abr -x --audio-
All defined aliases are listed
output. Alias options can
aliases; so be careful to avoid
recursive options. As a safety
alias may be triggered a
times. This option can be used

Network Options:

<code>--proxy URL</code> HTTP/HTTPS/SOCKS proxy. To proper scheme, <code>socks5://user:pass@127.0.0.1:1080/</code> <code>proxy "")</code> for	Use the specified enable SOCKS proxy, specify a e.g. Pass in an empty string (<code>-- direct connection</code>
<code>--socket-timeout SECONDS</code> in seconds	Time to wait before giving up,
<code>--source-address IP</code> to	Client-side IP address to bind
<code>-4, --force-ipv4</code>	Make all connections via IPv4
<code>-6, --force-ipv6</code>	Make all connections via IPv6
<code>--enable-file-urls</code> disabled by	Enable file:// URLs. This is default for security reasons.

Geo-restriction:

<code>--geo-verification-proxy</code> URL address for default proxy if the option is not present) is used for the actual downloading	Use this proxy to verify the IP address for some geo-restricted sites. The specified by <code>--proxy</code> (or none, if the option is not present) is used for the actual downloading
<code>--xff</code> VALUE HTTP header to restriction. One of be useful), notation, or a code	How to fake X-Forwarded-For try bypassing geographic "default" (only when known to "never", an IP block in CIDR two-letter ISO 3166-2 country

Video Selection:

<p><code>-I, --playlist-items ITEM_SPEC</code> Comma separated playlist_index of the items</p> <p>range using</p> <p>backward</p> <p>also supported.</p> <p>from the right</p> <p>in reverse</p> <p>used on a</p> <p>download the items</p> <p><code>--min-filesize SIZE</code> Abort download if filesize is smaller than</p> <p><code>--max-filesize SIZE</code> Abort download if filesize is larger than</p> <p><code>--date DATE</code> Download only videos uploaded on this date.</p> <p>in the format</p> <p><code>N[day week month year][-</code></p> <p>downloads only</p> <p>two weeks ago</p> <p><code>--datebefore DATE</code> Download only videos uploaded on or before</p> <p>accepted is the</p> <p><code>--dateafter DATE</code> Download only videos uploaded on or after</p> <p>accepted is the</p>	<p>]"[START]:[STOP][:STEP]". For compatibility, START-STOP is</p> <p>Use negative indices to count and negative STEP to download order. E.g. <code>"-I 1:3,7,-5::2"</code></p> <p>playlist of size 15 will</p> <p>at index 1,2,3,7,11,13,15</p> <p>SIZE, e.g. 50k or 44.6M</p> <p>SIZE, e.g. 50k or 44.6M</p> <p>The date can be "YYYYMMDD" or</p> <p><code>[now today yesterday][-</code></p> <p>E.g. <code>"--date today-2weeks"</code></p> <p>videos uploaded on the same day</p> <p>this date. The date formats</p> <p>same as <code>--date</code></p> <p>this date. The date formats</p> <p>same as <code>--date</code></p>
---	--

`--match-filters FILTER`
`"OUTPUT TEMPLATE"`

number or a

defined in

also simply

field is

if the field

check multiple

"&" or

multiple times,

one of the

`match-filter`

`"like_count>?100 &`

`dogs\b'"` matches

OR those that

(or the like

also has a

phrase "cats &

`filter -"` to

download each

`--no-match-filters`
(default)

`--break-match-filters FILTER`
stops the

Generic video filter. Any

field can be compared with a

string using the operators

"Filtering Formats". You can

specify a field to match if the

present, use "`!field`" to check

is not present, and "&" to

conditions. Use a "\" to escape

quotes if needed. If used

the filter matches if atleast

conditions are met. E.g. `--`

`!is_live --match-filter`

`description~='(?i)\bcats \&`

only videos that are not live

have a like count more than 100

field is not available) and

description that contains the

dogs" (caseless). Use "`--match-`

interactively ask whether to

video

Do not use any `--match-filter`

Same as "`--match-filters`" but

is rejected	download process when a video
--no-break-match-filters filters (default)	Do not use any --break-match-
--no-playlist URL refers	Download only the video, if the
	to a video and a playlist
--yes-playlist URL refers to	Download the playlist, if the
	a video and a playlist
--age-limit YEARS for the given	Download only videos suitable
	age
--download-archive FILE in the	Download only videos not listed
	archive file. Record the IDs of
all	downloaded videos in it
--no-download-archive (default)	Do not use archive file
--max-downloads NUMBER files	Abort after downloading NUMBER
--break-on-existing encountering	Stop the download process when
	a file that is in the archive
--break-per-input break-on-existing,	Alters --max-downloads, --
	--break-match-filter, and
autonumber to	reset per input URL
--no-break-per-input options	--break-on-existing and similar
queue	terminates the entire download
--skip-playlist-after-errors N until the rest of	Number of allowed failures
	the playlist is skipped

Download Options:

<p>-N, --concurrent-fragments N dash/hlsnative concurrently</p> <p>-r, --limit-rate RATE per second,</p> <p>--throttled-rate RATE per second</p> <p>assumed and the</p> <p>e.g. 100K</p> <p>-R, --retries RETRIES 10), or</p> <p>--file-access-retries RETRIES file access</p> <p>"infinite"</p> <p>--fragment-retries RETRIES fragment (default is</p> <p>hlsnative and ISM)</p> <p>--retry-sleep [TYPE:]EXPR in seconds</p> <p>type of retry</p> <p>file_access,</p> <p>to. EXPR can</p> <p>linear=START[:END[:STEP=1]] or</p> <p>option can be</p> <p>sleep for the</p> <p>retry-sleep</p> <p>fragment:exp=1:20</p>	<p>Number of fragments of a video that should be downloaded (default is 1)</p> <p>Maximum download rate in bytes e.g. 50K or 4.2M</p> <p>Minimum download rate in bytes below which throttling is video data is re-extracted,</p> <p>Number of retries (default is "infinite"</p> <p>Number of times to retry on error (default is 3), or</p> <p>Number of retries for a 10), or "infinite" (DASH,</p> <p>Time to sleep between retries (optionally) prefixed by the (http (default), fragment, extractor) to apply the sleep be a number, exp=START[:END[:BASE=2]]]. This used multiple times to set the different retry types, e.g. -- linear=1::2 --retry-sleep</p>
--	--

<p>--skip-unavailable-fragments DASH, (default) unavailable-fragments) --abort-on-unavailable-fragments unavailable fragments) --keep-fragments disk after --no-keep-fragments after (default) --buffer-size SIZE 1024 or 16K --resize-buffer automatically resized buffer-size --no-resize-buffer buffer size --http-chunk-size SIZE HTTP 10M (default bypassing a webserver --playlist-random random order --lazy-playlist as they are n_entries,</p>	<p>Skip unavailable fragments for hlsnative and ISM downloads (Alias: --no-abort-on- unavailable-fragments) Abort download if a fragment is unavailable (Alias: --no-skip-unavailable- fragments) Keep downloaded fragments on disk after downloading is finished Delete downloaded fragments after downloading is finished Size of download buffer, e.g. 1024 or 16K (default is 1024) The buffer size is automatically resized from an initial value of -- buffer-size (default) Do not automatically adjust the buffer size Size of a chunk for chunk-based HTTP downloading, e.g. 10485760 or 10M (default is disabled). May be useful for bypassing bandwidth throttling imposed by a webserver (experimental) Download playlist videos in random order Process entries in the playlist as they are received. This disables n_entries,</p>
---	---

playlist-reverse	--playlist-random and --
--no-lazy-playlist only after	Process videos in the playlist
(default)	the entire playlist is parsed
--xattr-set-filesize ytdl.filesize with	Set file xattribute
--hls-use-mpegts HLS videos;	expected file size
the video	Use the mpegts container for
the chance	allowing some players to play
is	while downloading, and reducing
default for	of file corruption if download
--no-hls-use-mpegts for HLS	interrupted. This is enabled by
not downloading	live streams
--download-sections REGEX match the	Do not use the mpegts container
prefix denotes	videos. This is default when
Negative	live streams
the end.	Download only chapters that
download between	regular expression. A "*"
extracted	time-range instead of chapter.
This option can	timestamps are calculated from
download multiple	"*from-url" can be used to
sections	the "start_time" and "end_time"
	from the URL. Needs ffmpeg.
	be used multiple times to
	sections, e.g. --download-

sections "intro"
--downloader [PROTO:]NAME
downloader to
the protocols
rtmp, mms) to
native,
ffmpeg, httpie,
multiple times
for different
aria2c
will use
and the
downloads
--downloader-args NAME:ARGS
external
downloader name and
colon ":". For
to different
as
use this
different
downloaders (Alias:

"*10:15-inf" --download-
Name or path of the external
use (optionally) prefixed by
(http, ftp, m3u8, dash, rstp,
use it for. Currently supports
aria2c, avconv, axel, curl,
wget. You can use this option
to set different downloaders
protocols. E.g. --downloader
--downloader "dash,m3u8:native"
aria2c for http/ftp downloads,
native downloader for dash/m3u8
(Alias: --external-downloader)
Give these arguments to the
downloader. Specify the
the arguments separated by a
ffmpeg, arguments can be passed
positions using the same syntax
--postprocessor-args. You can
option multiple times to give
arguments to different
--external-downloader-args)

Filesystem Options:

<p><code>-a, --batch-file FILE</code> download ("-") for starting considered as</p> <p><code>--no-batch-file</code> file (default)</p> <p><code>-P, --paths [TYPES:]PATH</code> should be</p> <p>file and the</p> <p>All the same</p> <p>supported.</p> <p>provide "home"</p> <p>intermediary</p> <p>the temp path</p> <p>moved over to</p> <p>finished.</p> <p>output is an</p> <p><code>-o, --output [TYPES:]TEMPLATE</code> "OUTPUT"</p> <p><code>--output-na-placeholder TEXT</code> fields in</p> <p>"NA")</p> <p><code>--restrict-filenames</code> ASCII characters,</p> <p>filenames</p> <p><code>--no-restrict-filenames</code> and spaces in</p>	<p>File containing URLs to stdin), one URL per line. Lines with "#", ";" or "]" are comments and ignored</p> <p>Do not read URLs from batch</p> <p>The paths where the files downloaded. Specify the type of path separated by a colon ":".</p> <p>TYPES as <code>--output</code> are</p> <p>Additionally, you can also (default) and "temp" paths. All files are first downloaded to and then the final files are the home path after download is</p> <p>This option is ignored if <code>--</code> absolute path</p> <p>Output filename template; see TEMPLATE" for details</p> <p>Placeholder for unavailable "OUTPUT TEMPLATE" (default: Restrict filenames to only and avoid "&" and spaces in</p> <p>Allow Unicode characters, "&"</p>
---	---

--windows-filenames compatible	filenames (default) Force filenames to be Windows-
--no-windows-filenames compatible only if	Make filenames Windows-
--trim-filenames LENGTH (excluding	using Windows (default) Limit the filename length
number of	extension) to the specified
	characters
-w, --no-overwrites	Do not overwrite any files
--force-overwrites metadata files. This	Overwrite all video and
	option includes --no-continue
--no-force-overwrites overwrite	Do not overwrite the video, but
	related files (default)
-c, --continue files/fragments	Resume partially downloaded
	(default)
--no-continue downloaded	Do not resume partially
fragmented,	fragments. If the file is not
file	restart download of the entire
--part writing directly	Use .part files instead of
	into output file (default)
--no-part directly into	Do not use .part files - write
	output file
--mtime set the file	Use the Last-modified header to
	modification time (default)
--no-mtime header to set	Do not use the Last-modified
	the file modification time
--write-description .description file	Write video description to a
--no-write-description (default)	Do not write video description

<p>--write-info-json .info.json file</p> <p>information)</p> <p>--no-write-info-json (default)</p> <p>--write-playlist-metafiles addition to the</p> <p>write-info-json, (default)</p> <p>--no-write-playlist-metafiles when using</p> <p>description etc.</p> <p>--clean-info-json such as</p> <p>(default)</p> <p>--no-clean-info-json infojson</p> <p>--write-comments placed in the</p> <p>fetches even</p> <p>extraction is</p> <p>get-comments)</p> <p>--no-write-comments unless the</p> <p>(Alias:</p> <p>--load-info-json FILE information</p> <p>info-json" option)</p> <p>--cookies FILE cookies from</p> <p>--no-cookies</p>	<p>Write video metadata to a (this may contain personal</p> <p>Do not write video metadata</p> <p>Write playlist metadata in video metadata when using -- --write-description etc.</p> <p>Do not write playlist metadata --write-info-json, --write-</p> <p>Remove some internal metadata filenames from the infojson</p> <p>Write all fields to the</p> <p>Retrieve video comments to be infojson. The comments are without this option if the known to be quick (Alias: --</p> <p>Do not retrieve video comments extraction is known to be quick</p> <p>--no-get-comments) JSON file containing the video (created with the "--write-</p> <p>Netscape formatted file to read and dump cookie jar in</p> <p>Do not read/dump cookies</p>
--	---

from/to file (default)

--cookies-from-browser BROWSER[+KEYRING][:PROFILE][::CONTAINER] The name of the browser to load cookies from. Currently supported browsers are: brave, chrome, chromium, edge, firefox, opera, safari, vivaldi. Optionally, the KEYRING used for decrypting Chromium cookies on Linux, the name/path of the PROFILE to load cookies from, and the CONTAINER name (if Firefox) ("none" for no container) can be given with their respective separators. By default, all containers of the most recently accessed profile are used.

are: basictext, Currently supported keyrings
kwallet5, kwallet6, gnomekeyring, kwallet,

--no-cookies-from-browser Do not load cookies from browser (default)

--cache-dir DIR Location in the filesystem where yt-dlp can store some downloaded information (such as client ids and signatures) permanently. By default \${XDG_CACHE_HOME}/yt-dlp

--no-cache-dir Disable filesystem caching

--rm-cache-dir Delete all filesystem cache files

Thumbnail Options:

<code>--write-thumbnail</code>	Write thumbnail image to disk
<code>--no-write-thumbnail</code> disk (default)	Do not write thumbnail image to disk
<code>--write-all-thumbnails</code> formats to disk	Write all thumbnail image formats to disk
<code>--list-thumbnails</code> each video.	List available thumbnails of each video.
is used	Simulate unless <code>--no-simulate</code> is used

Internet Shortcut Options:

<code>--write-link</code> file, depending on the current platform (<code>.url</code> , <code>.webloc</code> or <code>.desktop</code>). The URL may be cached by the OS	Write an internet shortcut on the current platform (<code>.url</code> , <code>.webloc</code> or <code>.desktop</code>). The URL may be cached by the OS
<code>--write-url-link</code> shortcut. The file path	Write a <code>.url</code> Windows internet shortcut. The OS caches the URL based on the file path
<code>--write-webloc-link</code> shortcut	Write a <code>.webloc</code> macOS internet shortcut
<code>--write-desktop-link</code> shortcut	Write a <code>.desktop</code> Linux internet shortcut

Verbosity and Simulation Options:

<p>-q, --quiet with --verbose,</p> <p>--no-quiet (Default)</p> <p>--no-warnings</p> <p>-s, --simulate do not write</p> <p>--no-simulate printing/listing</p> <p>--ignore-no-formats-error error. Useful for videos are download</p> <p>--no-ignore-no-formats-error downloadable video</p> <p>--skip-download write all download)</p> <p>-O, --print [WHEN:]TEMPLATE to print to with when to Supported as that of video).</p> <p>simulate unless of WHEN are multiple times</p> <p>--print-to-file [WHEN:]TEMPLATE FILE</p>	<p>Activate quiet mode. If used print the log to stderr Deactivate quiet mode.</p> <p>Ignore warnings</p> <p>Do not download the video and anything to disk Download the video even if options are used</p> <p>Ignore "No video formats" extracting metadata even if the not actually available for (experimental)</p> <p>Throw error when no formats are found (default)</p> <p>Do not download the video but related files (Alias: --no- download)</p> <p>Field name or output template screen, optionally prefixed print it, separated by a ":". values of "WHEN" are the same</p> <p>--use-postprocessor (default: Implies --quiet. Implies -- --no-simulate or later stages used. This option can be used</p>
--	--

file. The same as that syntax as the can be used

`-j, --dump-json`
information for each

simulate is

a

`-J, --dump-single-json`
information for each

Simulate unless

URL refers to

information

`--force-write-archive`
to be written

if `-s` or

used (Alias:

`--newline`
lines

`--no-progress`

`--progress`
quiet mode

`--console-title`
titlebar

`--progress-template [TYPES:]TEMPLATE`

optionally

Append given template to the values of `WHEN` and `TEMPLATE` are of `--print`. `FILE` uses the same output template. This option multiple times

Quiet, but print JSON

video. Simulate unless `--no-`

used. See "OUTPUT TEMPLATE" for

description of available keys

Quiet, but print JSON

url or infojson passed.

`--no-simulate` is used. If the

a playlist, the whole playlist

is dumped in a single line

Force download archive entries

as far as no errors occur, even

another simulation option is

`--force-download-archive`)

Output progress bar as new

Do not print progress bar

Show progress bar, even if in

Display progress in console

Template for progress outputs,

prefixed with one of

"download:" (default),
title),
"postprocess-title:".

accessible under the
attributes are

key. E.g.

template

(progress.eta)s"

-v, --verbose
information

--dump-pages
using base64

verbose)

--write-pages
pages to files

debug problems

--print-traffic
traffic

"download-title:" (the console

"postprocess:", or

The video's fields are

"info" key and the progress

accessible under "progress"

--console-title --progress-

"download-title:%(info.id)s-%

Print various debugging

Print downloaded pages encoded

to debug problems (very

Write downloaded intermediary

in the current directory to

Display sent and read HTTP

Workarounds:

<code>--encoding ENCODING</code> (experimental)	Force the specified encoding
<code>--legacy-server-connect</code> connection to servers	Explicitly allow HTTPS
<code>secure</code>	that do not support RFC 5746
<code>--no-check-certificates</code> validation	renegotiation
<code>--prefer-insecure</code> to retrieve	Suppress HTTPS certificate
(Currently	Use an unencrypted connection
<code>--add-headers FIELD:VALUE</code> and its value,	information about the video
can use this	supported only for YouTube)
<code>--bidi-workaround</code>	Specify a custom HTTP header
Requires <code>bidiv</code>	separated by a colon ":". You
<code>--sleep-requests SECONDS</code> between requests	option multiple times
<code>--sleep-interval SECONDS</code> before each	Work around terminals that lack
time to sleep	bidirectional text support.
<code>sleep-interval</code>	or <code>fribidi</code> executable in <code>PATH</code>
<code>--max-sleep-interval SECONDS</code> sleep. Can only	Number of seconds to sleep
<code>interval</code>	during data extraction
<code>--sleep-subtitles SECONDS</code> before each	Number of seconds to sleep
	download. This is the minimum
	when used along with <code>--max-</code>
	(Alias: <code>--min-sleep-interval</code>)
	Maximum number of seconds to
	be used along with <code>--min-sleep-</code>
	Number of seconds to sleep
	subtitle download

Video Format Options:

<p><code>-f, --format FORMAT SELECTION"</code></p>	<p>Video format code, see "FORMAT for more details</p>
<p><code>-S, --format-sort SORTORDER given, see</code></p>	<p>Sort the formats by the fields "Sorting Formats" for more</p>
<p><code>--format-sort-force</code> to have</p>	<p>Force user specified sort order precedence over all fields, see</p>
<p>"Sorting (Alias: <code>--S-force</code>) <code>--no-format-sort-force</code> over the user</p>	<p>Formats" for more details Some fields have precedence specified sort order (default)</p>
<p><code>--video-multistreams</code> be merged</p>	<p>Allow multiple video streams to into a single file</p>
<p><code>--no-video-multistreams</code> downloaded for each</p>	<p>Only one video stream is output file (default)</p>
<p><code>--audio-multistreams</code> be merged</p>	<p>Allow multiple audio streams to into a single file</p>
<p><code>--no-audio-multistreams</code> downloaded for each</p>	<p>Only one audio stream is output file (default)</p>
<p><code>--prefer-free-formats</code> containers quality. Use with free containers</p>	<p>Prefer video formats with free over non-free ones of same "-S ext" to strictly prefer irrespective of quality</p>
<p><code>--no-prefer-free-formats</code> preference to free</p>	<p>Don't give any special containers (default)</p>
<p><code>--check-formats</code> only from downloadable</p>	<p>Make sure formats are selected those that are actually</p>
<p><code>--check-all-formats</code></p>	<p>Check all formats for whether</p>

they are

--no-check-formats
are actually

-F, --list-formats
video.

is used

--merge-output-format FORMAT
when merging

"mp4/mkv".

required. (currently

mp4, webm)

actually downloadable

Do not check that the formats

downloadable

List available formats of each

Simulate unless --no-simulate

Containers that may be used

formats, separated by "/", e.g.

Ignored if no merge is

supported: avi, flv, mkv, mov,

Subtitle Options:

<code>--write-sub</code>	Write subtitle file
<code>--no-write-sub</code> (default)	Do not write subtitle file
<code>--write-auto-sub</code> subtitle file	Write automatically generated (Alias: <code>--write-automatic-sub</code>)
<code>--no-write-auto-sub</code> subtitles	Do not write auto-generated (default) (Alias: <code>--no-write-</code>
<code>automatic-sub</code>)	
<code>--list-sub</code> each video.	List available subtitles of
	Simulate unless <code>--no-simulate</code>
is used	
<code>--sub-format</code> FORMAT	Subtitle format; accepts
formats preference,	e.g. "srt" or "ass/srt/best"
<code>--sub-lang</code> LANGS	Languages of the subtitles to
download (can	be regex) or "all" separated by
commas, e.g.	<code>--sub-lang "en.*,ja"</code> . You can
prefix the	language code with a "-" to
exclude it from	the requested languages, e.g. -
<code>-sub-lang</code>	<code>all,-live_chat</code> . Use <code>--list-sub</code>
for a list	of available language tags

Authentication Options:

-u, --username USERNAME	Login with this account ID
-p, --password PASSWORD	Account password. If this
option is left	out, yt-dlp will ask
interactively	
-2, --twofactor TWOFACOR	Two-factor authentication code
-n, --netrc	Use .netrc authentication data
--netrc-location PATH	Location of .netrc
authentication data;	
containing directory.	either the path or its
	Defaults to ~/.netrc
--netrc-cmd NETRC_CMD	Command to execute to get the
credentials	for an extractor.
--video-password PASSWORD	Video password (vimeo, youku)
--ap-mso MSO	Adobe Pass multiple-system
operator (TV	
list-mso for	provider) identifier, use --ap-
	a list of available MSOs
--ap-username USERNAME	Multiple-system operator
account login	
--ap-password PASSWORD	Multiple-system operator
account password.	
	If this option is left out, yt-
dlp will ask	interactively
--ap-list-mso	List all supported multiple-
system operators	
--client-certificate CERTFILE	Path to client certificate file
in PEM	
key	format. May include the private
--client-certificate-key KEYFILE	
client	Path to private key file for
	certificate
--client-certificate-password PASSWORD	
private key,	Password for client certificate
and the key	if encrypted. If not provided,

interactively is encrypted, yt-dlp will ask

Post-Processing Options:

`-x, --extract-audio`
only files

`--audio-format` FORMAT
when `-x` is

best (default),
opus, vorbis,

rules using

`--audio-quality` QUALITY
use when

Insert a value
for VBR or a
(default 5)

`--remux-video` FORMAT
container if

avi, flv,
aiff, alac,
vorbis,
not support
will fail.

e.g.

remux aac to m4a,
mkv

`--recode-video` FORMAT
another format if
supported formats

Convert video files to audio-
(requires ffmpeg and ffprobe)
Format to convert the audio to
used. (currently supported:
aac, alac, flac, m4a, mp3,
wav). You can specify multiple
similar syntax as `--remux-video`
Specify ffmpeg audio quality to
converting the audio with `-x`.
between 0 (best) and 10 (worst)
specific bitrate like 128K

Remux the video into another
necessary (currently supported:
gif, mkv, mov, mp4, webm, aac,
flac, m4a, mka, mp3, ogg, opus,
wav). If target container does
the video/audio codec, remuxing
You can specify multiple rules;
"aac>m4a/mov>mp4/mkv" will
mov to mp4 and anything else to

Re-encode the video into
necessary. The syntax and
are the same as `--remux-video`

--postprocessor-args NAME:ARGS
postprocessors.

postprocessor/executable name

a colon ":"

specified

Supported PP are:

SplitChapters,

VideoConvertor,

EmbedThumbnail,

ThumbnailsConvertor,

FixupM3u8,

FixupDuration. The

AtomicParsley,

also specify

arguments to the

being used by

Additionally,

can be

optionally followed

argument before the

e.g. --ppa

You can use

Give these arguments to the

Specify the

and the arguments separated by

to give the argument to the

postprocessor/executable.

Merger, ModifyChapters,

ExtractAudio, VideoRemuxer,

Metadata, EmbedSubtitle,

SubtitlesConvertor,

FixupStretched, FixupM4a,

FixupTimestamp and

supported executables are:

FFmpeg and FFprobe. You can

"PP+EXE:ARGS" to give the

specified executable only when

the specified postprocessor.

for ffmpeg/ffprobe, "_i"/"_o"

appended to the prefix

by a number to pass the

specified input/output file,

"Merger+ffmpeg_i1:-v quiet".

this option multiple times to

give different	arguments to different
postprocessors.	
-k, --keep-video	(Alias: --ppa)
file on disk	Keep the intermediate video
	after post-processing
--no-keep-video	Delete the intermediate video
file after	
	post-processing (default)
--post-overwrites	Overwrite post-processed files
(default)	
--no-post-overwrites	Do not overwrite post-processed
files	
--embed-sub	Embed subtitles in the video
(only for mp4,	
	webm and mkv videos)
--no-embed-sub	Do not embed subtitles
(default)	
--embed-thumbnail	Embed thumbnail in the video as
cover art	
--no-embed-thumbnail	Do not embed thumbnail
(default)	
--embed-metadata	Embed metadata to the video
file. Also	
	embeds chapters/infojson if
present unless	
	--no-embed-chapters/--no-embed-
info-json are	
	used (Alias: --add-metadata)
--no-embed-metadata	Do not add metadata to file
(default)	
	(Alias: --no-add-metadata)
--embed-chapters	Add chapter markers to the
video file	
	(Alias: --add-chapters)
--no-embed-chapters	Do not add chapter markers
(default) (Alias:	
	--no-add-chapters)
--embed-info-json	Embed the infojson as an
attachment to	
	mkv/mka video files
--no-embed-info-json	Do not embed the infojson as an

attachment to the video file

--parse-metadata [WHEN:]FROM:TO Parse additional metadata like title/artist from other fields; see "MODIFYING METADATA" for details. Supported values of "WHEN" are the same as that of --use-postprocessor (default: pre_process)

--replace-in-metadata [WHEN:]FIELDS REGEX REPLACE Replace text in a metadata field using the given regex. This option can be used multiple times. Supported values of "WHEN" are the same as that of --use-postprocessor (default: pre_process)

--xattrs Write metadata to the video file's xattrs (using dublin core and xdg standards)

--concat-playlist POLICY Concatenate videos in a playlist. One of "never", "always", or "multi_video" (default; only when the videos form a single show). All the video files must have same codecs and number of streams to be concatatable. The "pl_video:" prefix can be used with "--paths" and "--output" to set the output filename for the concatenated files. See "OUTPUT TEMPLATE"

for details	Automatically correct known
--fixup POLICY	file. One of never (do
faults of the	emit a warning), detect_or_warn
nothing), warn (only	default; fix file if we can,
(the	otherwise), force (try fixing
warn	already exists)
even if file	Location of the ffmpeg binary;
--ffmpeg-location PATH	path to the binary or its
either the	Execute a command, optionally
containing directory	when to execute it, separated
--exec [WHEN:]CMD	Supported values of "WHEN" are
prefixed with	that of --use-postprocessor
by a ":".	after_move). Same syntax as the
the same as	template can be used to pass
(default:	arguments to the command. If no
output	passed, %(filepath,_filename)q
any field as	to the end of the command. This
fields are	be used multiple times
is appended	Remove any previously defined -
option can	Convert the subtitles to
--no-exec	(currently supported: ass, lrc,
-exec	(Alias: --convert-subtitles)
--convert-subts FORMAT	Convert the thumbnails to
another format	
srt, vtt)	
--convert-thumbnails FORMAT	

another format (currently supported: jpg, png, webp). You can specify multiple rules using similar syntax as `--remux-video`

`--split-chapters` Split video into multiple files based on internal chapters. The `"chapter:"` prefix can be used with `--paths` and `--output` to set the output filename for the split files. See `"OUTPUT TEMPLATE"` for details

`--no-split-chapters` Do not split video based on chapters (default)

`--remove-chapters REGEX` Remove chapters whose title matches the given regular expression. The syntax is the same as `--download-sections`. This option can be used multiple times

`--no-remove-chapters` Do not remove any chapters from the file (default)

`--force-keyframes-at-cuts` Force keyframes at cuts when downloading/splitting/removing sections. This is slow due to needing a re-encode, but the resulting video may have fewer artifacts around the cuts

`--no-force-keyframes-at-cuts` Do not force keyframes around the chapters when cutting/splitting (default)

`--use-postprocessor NAME[:ARGS]` The (case sensitive) name of plugin postprocessors to be enabled,

and
passed to it,
are a
NAME=VALUE.
when the
can be one of
extraction),
passes filter),

(before each
(after each
"after_move"
it's final
(after downloading
video), or
playlist). This option
add different

(optionally) arguments to be
separated by a colon ":". ARGS
semicolon ";" delimited list of
The "when" argument determines
postprocessor is invoked. It
"pre_process" (after video
"after_filter" (after video
"video" (after --format; before
--print/--output), "before_dl"
video download), "post_process"
video download; default),
(after moving video file to
locations), "after_video"
and processing all formats of a
"playlist" (at end of
can be used multiple times to
postprocessors

SponsorBlock Options:

Make chapter entries for, or remove various segments (sponsor, introductions, etc.) from downloaded YouTube videos using the [SponsorBlock API](#)

`--sponsorblock-mark CATS`
create chapters

Available

outro,

interaction,

chapter, all

prefix the

it. See [1]

categories. E.g.

preview

https://wiki.sponsor.ajay.app/w/Segment_Categories

`--sponsorblock-remove CATS`
removed from

commas. If a

mark and remove,

syntax and

same as for

"default"

poi_highlight,

`--sponsorblock-chapter-title TEMPLATE`

title of the

by

available

SponsorBlock categories to

for, separated by commas.

categories are sponsor, intro,

selfpromo, preview, filler,

music_offtopic, poi_highlight,

and default (=all). You can

category with a "-" to exclude

for description of the

`--sponsorblock-mark all,-`

[1]

SponsorBlock categories to be

the video file, separated by

category is present in both

remove takes precedence. The

available categories are the

`--sponsorblock-mark` except that

refers to "all,-filler" and

chapter are not available

An output template for the

SponsorBlock chapters created

`--sponsorblock-mark`. The only

fields are start_time,

end_time, category,

category_names. Defaults

(category_names)1"

--no-sponsorblock
mark and

--sponsorblock-api URL
defaults to

categories, name,

to "[SponsorBlock]: %

Disable both --sponsorblock-

--sponsorblock-remove
SponsorBlock API location,

https://sponsor.ajay.app

Extractor Options:

--extractor-retries RETRIES
extractor errors

Number of retries for known

(default is 3), or "infinite"

--allow-dynamic-mpd
(default)

Process dynamic DASH manifests

(Alias: --no-ignore-dynamic-

mpd)

--ignore-dynamic-mpd
manifests

Do not process dynamic DASH

(Alias: --no-allow-dynamic-mpd)

--hls-split-discontinuity
different formats at

Split HLS playlists to

discontinuities such as ad

breaks

--no-hls-split-discontinuity
different

Do not split HLS playlists to

formats at discontinuities such

as ad breaks

(default)

--extractor-args IE_KEY:ARGS
IE_KEY extractor.

Pass ARGS arguments to the

See "EXTRACTOR ARGUMENTS" for

details. You

can use this option multiple

times to give

arguments for different

extractors

CONFIGURATION

You can configure yt-dlp by placing any supported command line option to a configuration file. The configuration is loaded from the following locations:

1. Main Configuration:

The file given by `--config-location`

2. Portable Configuration: (Recommended for portable installations)

- If using a binary, `yt-dlp.conf` in the same directory as the binary
- If running from source-code, `yt-dlp.conf` in the parent directory of `yt_dlp`

3. Home Configuration:

- `yt-dlp.conf` in the home path given by `-P`
- If `-P` is not given, the current directory is searched

4. User Configuration:

- `${XDG_CONFIG_HOME}/yt-dlp.conf`
- `${XDG_CONFIG_HOME}/yt-dlp/config` (recommended on Linux/macOS)
- `${XDG_CONFIG_HOME}/yt-dlp/config.txt`
- `${APPDATA}/yt-dlp.conf`
- `${APPDATA}/yt-dlp/config` (recommended on Windows)
- `${APPDATA}/yt-dlp/config.txt`
- `~/yt-dlp.conf`
- `~/yt-dlp.conf.txt`
- `~/yt-dlp/config`
- `~/yt-dlp/config.txt`

See also: [Notes about environment variables](#)

5. System Configuration:

- `/etc/yt-dlp.conf`
- `/etc/yt-dlp/config`
- `/etc/yt-dlp/config.txt`

E.g. with the following configuration file yt-dlp will always extract the audio, not copy the mtime, use a proxy and save all videos under `YouTube` directory in your home directory:

```
# Lines starting with # are comments

# Always extract audio
-x

# Do not copy the mtime
--no-mtime

# Use this proxy
--proxy 127.0.0.1:3128

# Save all videos under YouTube directory in your home
directory
-o ~/YouTube/%(title)s.%(ext)s
```

Note: Options in configuration file are just the same options aka switches used in regular command line calls; thus there **must be no whitespace** after `-` or `--`, e.g. `-o` or `--proxy` but not `- o` or `-- proxy`. They must also be quoted when necessary as-if it were a UNIX shell.

You can use `--ignore-config` if you want to disable all configuration files for a particular yt-dlp run. If `--ignore-config` is found inside any configuration file, no further configuration will be loaded. For example, having the option in the portable configuration file prevents loading of home, user, and system configurations. Additionally, (for backward compatibility) if `--ignore-config` is found inside the system configuration file, the user configuration is not loaded.

Configuration file encoding

The configuration files are decoded according to the UTF BOM if present, and in the encoding from system locale otherwise.

If you want your file to be decoded differently, add `# coding: ENCODING` to the beginning of the file (e.g. `# coding: shift-jis`). There must be no characters before that, even spaces or BOM.

Authentication with netrc

You may also want to configure automatic credentials storage for extractors that support authentication (by providing login and password with `--username` and `--password`) in order not to pass credentials as command line arguments on every yt-dlp execution and prevent tracking plain text passwords in the shell command history. You can achieve this using a `.netrc` file on a per-extractor basis. For that you will need to create a `.netrc` file in `--netrc-location` and restrict permissions to read/write by only you:

```
touch ${HOME}/.netrc
chmod a-rwx,u+rw ${HOME}/.netrc
```

After that you can add credentials for an extractor in the following format, where *extractor* is the name of the extractor in lowercase:

```
machine <extractor> login <username> password <password>
```

E.g.

```
machine youtube login myaccount@gmail.com password
my_youtube_password
machine twitch login my_twitch_account_name password
my_twitch_password
```

To activate authentication with the `.netrc` file you should pass `--netrc` to yt-dlp or place it in the configuration file.

The default location of the `.netrc` file is `~` (see below).

As an alternative to using the `.netrc` file, which has the disadvantage of keeping your passwords in a plain text file, you can configure a custom shell command to provide the credentials for an extractor. This is done by providing the `--netrc-cmd` parameter, it shall output the credentials in the netrc format and return `0` on success, other values will be treated as an error. `{}` in the command will be replaced by the name of the extractor to make it possible to select the credentials for the right extractor.

E.g. To use an encrypted `.netrc` file stored as `.authinfo.gpg`

```
yt-dlp --netrc-cmd 'gpg --decrypt ~/.authinfo.gpg'  
https://www.youtube.com/watch?v=BaW_jenozKc
```

Notes about environment variables

- Environment variables are normally specified as `${VARIABLE}/$VARIABLE` on UNIX and `%VARIABLE%` on Windows; but is always shown as `${VARIABLE}` in this documentation
- yt-dlp also allow using UNIX-style variables on Windows for path-like options; e.g. `--output`, `--config-location`
- If unset, `${XDG_CONFIG_HOME}` defaults to `~/.config` and `${XDG_CACHE_HOME}` to `~/.cache`
- On Windows, `~` points to `${HOME}` if present; or, `${USERPROFILE}` or `${HOMEDRIVE}${HOMEPATH}` otherwise
- On Windows, `${USERPROFILE}` generally points to `C:\Users\
<user name>` and `${APPDATA}` to `${USERPROFILE}\AppData\Roaming`

OUTPUT TEMPLATE

The `-o` option is used to indicate a template for the output file names while `-P` option is used to specify the path each type of file should be saved to.

tl;dr: [navigate me to examples.](#)

The simplest usage of `-o` is not to set any template arguments when downloading a single file, like in `yt-dlp -o funny_video.flv "https://some/video"` (hard-coding file extension like this is *not* recommended and could break some post-processing).

It may however also contain special sequences that will be replaced when downloading each video. The special sequences may be formatted according to Python string formatting operations, e.g. `%(NAME)s` or `%(NAME)05d`. To clarify, that is a percent symbol followed by a name in parentheses, followed by formatting operations.

The field names themselves (the part inside the parenthesis) can also have some special formatting:

- 1. Object traversal:** The dictionaries and lists available in metadata can be traversed by using a dot `.` separator; e.g. `%(tags.0)s`, `%(subtitles.en.-1.ext)s`. You can do Python slicing with colon `:`; E.g. `%(id.3:7:-1)s`, `%(formats.:.format_id)s`. Curly braces `{}` can be used to build dictionaries with only specific keys; e.g. `%(formats.:{format_id,height})#j`. An empty field name `%()s` refers to the entire infodict; e.g. `%(.{id,title})s`. Note that all the fields that become available using this method are not listed below. Use `-j` to see such fields
- 2. Addition:** Addition and subtraction of numeric fields can be done using `+` and `-` respectively. E.g. `%(playlist_index+10)03d`, `%(n_entries+1-playlist_index)d`
- 3. Date/time Formatting:** Date/time fields can be formatted according to strftime formatting by specifying it separated from the field name using a `>`. E.g. `%(duration>%H-%M-%S)s`, `%(upload_date>%Y-%m-%d)s`, `%(epoch-3600>%H-%M-%S)s`
- 4. Alternatives:** Alternate fields can be specified separated with a `,`. E.g. `%(release_date>%Y,upload_date>%Y|Unknown)s`

5. **Replacement:** A replacement value can be specified using a `&` separator according to the `str.format` mini-language. If the field is *not* empty, this replacement value will be used instead of the actual field content. This is done after alternate fields are considered; thus the replacement is used if *any* of the alternative fields is *not* empty. E.g. `%(chapters&has chapters|no chapters)s, %(title&TITLE={:>20}|NO TITLE)s`
6. **Default:** A literal default value can be specified for when the field is empty using a `|` separator. This overrides `--output-na-placeholder`. E.g. `%(uploader|Unknown)s`
7. **More Conversions:** In addition to the normal format types `diouxXeEfFgGcrs`, yt-dlp additionally supports converting to `B` = Bytes, `j` = json (flag `#` for pretty-printing, `+` for Unicode), `h` = HTML escaping, `l` = a comma separated list (flag `#` for `\n` newline-separated), `q` = a string quoted for the terminal (flag `#` to split a list into different arguments), `D` = add **D**ecimal suffixes (e.g. 10M) (flag `#` to use 1024 as factor), and `S` = **S**anitize as filename (flag `#` for restricted)
8. **Unicode normalization:** The format type `U` can be used for NFC Unicode normalization. The alternate form flag (`#`) changes the normalization to NFD and the conversion flag `+` can be used for NFKC/NFKD compatibility equivalence normalization. E.g. `%(title)+.100U` is NFKC

To summarize, the general syntax for a field is:

```
%(name[.keys][addition][>strf][,alternate][&replacement]
[|default])[flags][width][.precision][length]type
```

Additionally, you can set different output templates for the various metadata files separately from the general output template by specifying the type of file followed by the template separated by a colon `:`. The different file types supported are `subtitle`, `thumbnail`,

`description`, `annotation` (deprecated), `infojson`, `link`, `pl_thumbnail`, `pl_description`, `pl_infojson`, `chapter`, `pl_video`.
E.g. `-o "%(title)s.%(ext)s" -o "thumbnail:%(title)s\%(title)s.%(ext)s"` will put the thumbnails in a folder with the same name as the video. If any of the templates is empty, that type of file will not be written. E.g. `--write-thumbnail -o "thumbnail:"` will write thumbnails only for playlists and not for video.

Note: Due to post-processing (i.e. merging etc.), the actual output filename might differ. Use `--print-after-move:filepath` to get the name after all post-processing is complete.

The available fields are:

- `id`(string): Video identifier
- `title`(string): Video title
- `fulltitle`(string): Video title ignoring live timestamp and generic title
- `ext`(string): Video filename extension
- `alt_title`(string): A secondary title of the video
- `description`(string): The description of the video
- `display_id`(string): An alternative identifier for the video
- `uploader`(string): Full name of the video uploader
- `license`(string): License name the video is licensed under
- `creator`(string): The creator of the video
- `timestamp`(numeric): UNIX timestamp of the moment the video became available
- `upload_date`(string): Video upload date in UTC (YYYYMMDD)
- `release_timestamp`(numeric): UNIX timestamp of the moment the video was released
- `release_date`(string): The date (YYYYMMDD) when the video was released in UTC
- `modified_timestamp`(numeric): UNIX timestamp of the moment the video was last modified

- modified_date (string): The date (YYYYMMDD) when the video was last modified in UTC
- uploader_id (string): Nickname or id of the video uploader
- channel (string): Full name of the channel the video is uploaded on
- channel_id (string): Id of the channel
- channel_follower_count (numeric): Number of followers of the channel
- channel_is_verified (boolean): Whether the channel is verified on the platform
- location (string): Physical location where the video was filmed
- duration (numeric): Length of the video in seconds
- duration_string (string): Length of the video (HH:mm:ss)
- view_count (numeric): How many users have watched the video on the platform
- concurrent_view_count (numeric): How many users are currently watching the video on the platform.
- like_count (numeric): Number of positive ratings of the video
- dislike_count (numeric): Number of negative ratings of the video
- repost_count (numeric): Number of reposts of the video
- average_rating (numeric): Average rating give by users, the scale used depends on the webpage
- comment_count (numeric): Number of comments on the video (For some extractors, comments are only downloaded at the end, and so this field cannot be used)
- age_limit (numeric): Age restriction for the video (years)
- live_status (string): One of "not live", "is live", "is upcoming", "was live", "post live" (was live, but VOD is not yet processed)
- is_live (boolean): Whether this video is a live stream or a fixed-length video
- was_live (boolean): Whether this video was originally a live stream

- playable_in_embed_(string): Whether this video is allowed to play in embedded players on other sites
- availability_(string): Whether the video is "private", "premium only", "subscriber only", "needs auth", "unlisted" or "public"
- start_time_(numeric): Time in seconds where the reproduction should start, as specified in the URL
- end_time_(numeric): Time in seconds where the reproduction should end, as specified in the URL
- extractor_(string): Name of the extractor
- extractor_key_(string): Key name of the extractor
- epoch_(numeric): Unix epoch of when the information extraction was completed
- autonumber_(numeric): Number that will be increased with each download, starting at --autonumber-start, padded with leading zeros to 5 digits
- video_autonumber_(numeric): Number that will be increased with each video
- n_entries_(numeric): Total number of extracted items in the playlist
- playlist_id_(string): Identifier of the playlist that contains the video
- playlist_title_(string): Name of the playlist that contains the video
- playlist_(string): playlist_id or playlist_title
- playlist_count_(numeric): Total number of items in the playlist. May not be known if entire playlist is not extracted
- playlist_index_(numeric): Index of the video in the playlist padded with leading zeros according the final index
- playlist_autonumber_(numeric): Position of the video in the playlist download queue padded with leading zeros according to the total length of the playlist
- playlist_uploader_(string): Full name of the playlist uploader

- [playlist_uploader_id](#)(string): Nickname or id of the [playlist uploader](#)
- [webpage_url](#)(string): A URL to the video webpage which if given to yt-dlp should allow to get the same result again
- [webpage_url_basename](#)(string): The basename of the webpage URL
- [webpage_url_domain](#)(string): The domain of the webpage URL
- [original_url](#)(string): The URL given by the user (or same as [webpage_url](#) for playlist entries)

All the fields in [Filtering Formats](#) can also be used

Available for the video that belongs to some logical chapter or section:

- [chapter](#) (string): Name or title of the chapter the video belongs to
- [chapter_number](#) (numeric): Number of the chapter the video belongs to
- [chapter_id](#) (string): Id of the chapter the video belongs to

Available for the video that is an episode of some series or programme:

- [series](#) (string): Title of the series or programme the video episode belongs to
- [season](#) (string): Title of the season the video episode belongs to
- [season_number](#) (numeric): Number of the season the video episode belongs to
- [season_id](#) (string): Id of the season the video episode belongs to
- [episode](#) (string): Title of the video episode
- [episode_number](#) (numeric): Number of the video episode within a season
- [episode_id](#) (string): Id of the video episode

Available for the media that is a track or a part of a music album:

- `track` (string): Title of the track
- `track_number` (numeric): Number of the track within an album or a disc
- `track_id` (string): Id of the track
- `artist` (string): Artist(s) of the track
- `genre` (string): Genre(s) of the track
- `album` (string): Title of the album the track belongs to
- `album_type` (string): Type of the album
- `album_artist` (string): List of all artists appeared on the album
- `disc_number` (numeric): Number of the disc or other physical medium the track belongs to
- `release_year` (numeric): Year (YYYY) when the album was released

Available only when using `--download-sections` and for `chapter:` prefix when using `--split-chapters` for videos with internal chapters:

- `section_title` (string): Title of the chapter
- `section_number` (numeric): Number of the chapter within the file
- `section_start` (numeric): Start time of the chapter in seconds
- `section_end` (numeric): End time of the chapter in seconds

Available only when used in `--print:`

- `urls` (string): The URLs of all requested formats, one in each line
- `filename` (string): Name of the video file. Note that the actual filename may differ
- `formats_table` (table): The video format table as printed by `--list-formats`
- `thumbnails_table` (table): The thumbnail format table as printed by `--list-thumbnails`

- `subtitles_table` (table): The subtitle format table as printed by `--list-subs`
- `automatic_captions_table` (table): The automatic subtitle format table as printed by `--list-subs`

Available only after the video is downloaded
(`post_process/after_move`):

`filepath`: Actual path of downloaded video file

Available only in `--sponsorblock-chapter-title`:

- `start_time` (numeric): Start time of the chapter in seconds
- `end_time` (numeric): End time of the chapter in seconds
- `categories` (list): The SponsorBlock categories the chapter belongs to
- `category` (string): The smallest SponsorBlock category the chapter belongs to
- `category_names` (list): Friendly names of the categories
- `name` (string): Friendly name of the smallest category
- `type` (string): The SponsorBlock action type of the chapter

Each aforementioned sequence when referenced in an output template will be replaced by the actual value corresponding to the sequence name. E.g. for `-o %(title)s-%(id)s.%(ext)s` and an mp4 video with title `yt-dlp test video` and id `BaW_jenozKc`, this will result in a `yt-dlp test video-BaW_jenozKc.mp4` file created in the current directory.

Note: Some of the sequences are not guaranteed to be present since they depend on the metadata obtained by a particular extractor. Such sequences will be replaced with placeholder value provided with `--output-na-placeholder` (`NA` by default).

Tip: Look at the `-j` output to identify which fields are available for the particular URL

For numeric sequences you can use numeric related formatting; e.g. `%(view_count)05d` will result in a string with view count padded with zeros up to 5 characters, like in `00042`.

Output templates can also contain arbitrary hierarchical path, e.g. `-o "%(playlist)s/%(playlist_index)s - %(title)s.%(ext)s"` which will result in downloading each video in a directory corresponding to this path template. Any missing directory will be automatically created for you.

To use percent literals in an output template use `%%`. To output to stdout use `-o -`.

The current default template is `%(title)s [%(id)s].%(ext)s`.

In some cases, you don't want special characters such as `中`, spaces, or `&`, such as when transferring the downloaded filename to a Windows system or the filename through an 8bit-unsafe channel. In these cases, add the `--restrict-filenames` flag to get a shorter title.

Output template examples

```
$ yt-dlp --print filename -o "test video.%(ext)s" BaW_jenozKc
test video.webm      # Literal name with correct extension
```

```
$ yt-dlp --print filename -o "%(title)s.%(ext)s" BaW_jenozKc
youtube-dl test video '_ä↔Y.webm      # All kinds of weird
characters
```

```
$ yt-dlp --print filename -o "%(title)s.%(ext)s" BaW_jenozKc --
restrict-filenames
youtube-dl_test_video_.webm      # Restricted file name
```

```
# Download YouTube playlist videos in separate directory
indexed by video order in a playlist
$ yt-dlp -o "%(playlist)s/%(playlist_index)s - %(title)s.%(
ext)s" "https://www.youtube.com/playlist?
list=PLwiyx1dc3P2JR9N8gQaQN_BCv1Slap7re"
```

```
# Download YouTube playlist videos in separate directories
according to their uploaded year
$ yt-dlp -o "%(upload_date>%Y)s/%(title)s.%(ext)s"
"https://www.youtube.com/playlist?
list=PLwiyx1dc3P2JR9N8gQaQN_BCv1Slap7re"
```

```
# Prefix playlist index with " - " separator, but only if it is
available
$ yt-dlp -o "%(playlist_index&{} - |)s%(title)s.%(ext)s"
BaW_jenozKc
"https://www.youtube.com/user/TheLinuxFoundation/playlists"
```

```
# Download all playlists of YouTube channel/user keeping each
playlist in separate directory:
$ yt-dlp -o "%(uploader)s/%(playlist)s/%(playlist_index)s - %
(title)s.%(ext)s"
"https://www.youtube.com/user/TheLinuxFoundation/playlists"
```

```
# Download Udemy course keeping each chapter in separate
directory under MyVideos directory in your home
$ yt-dlp -u user -p password -P "~/MyVideos" -o "%(playlist)s/%
(chapter_number)s - %(chapter)s/%(title)s.%(ext)s"
"https://www.udemy.com/java-tutorial"
```

```
# Download entire series season keeping each series and each
season in separate directory under C:/MyVideos
```

```

$ yt-dlp -P "C:/MyVideos" -o "%(series)s/%(season_number)s - %(
(season)s/%(episode_number)s - %(episode)s.%(ext)s"
"https://videomore.ru/kino_v_detalayah/5_sezon/367617"

# Download video as "C:\MyVideos\uploader\title.ext", subtitles
as "C:\MyVideos\subs\uploader\title.ext"
# and put all temporary files in "C:\MyVideos\tmp"
$ yt-dlp -P "C:/MyVideos" -P "temp:tmp" -P "subtitle:subs" -o
"%(uploader)s/%(title)s.%(ext)s" BaW_jenoz --write-sub

# Download video as "C:\MyVideos\uploader\title.ext" and
subtitles as "C:\MyVideos\uploader\subs\title.ext"
$ yt-dlp -P "C:/MyVideos" -o "%(uploader)s/%(title)s.%(ext)s" -
o "subtitle:%(uploader)s/subs/%(title)s.%(ext)s" BaW_jenozKc --
write-sub

# Stream the video being downloaded to stdout
$ yt-dlp -o - BaW_jenozKc

```

FORMAT SELECTION

By default, yt-dlp tries to download the best available quality if you **don't** pass any options. This is generally equivalent to using **-f bestvideo*+bestaudio/best**. However, if multiple audiostreams is enabled (**--audio-multistreams**), the default format changes to **-f bestvideo+bestaudio/best**. Similarly, if ffmpeg is unavailable, or if you use yt-dlp to stream to **stdout** (**-o -**), the default becomes **-f best/bestvideo+bestaudio**.

Deprecation warning: Latest versions of yt-dlp can stream multiple formats to the stdout simultaneously using ffmpeg. So, in future versions, the default for this will be set to **-f bv*+ba/b** similar to normal downloads. If you want to preserve the **-f b/bv+ba** setting, it is recommended to explicitly specify it in the configuration options.

The general syntax for format selection is **-f FORMAT** (or **--format FORMAT**) where **FORMAT** is a *selector expression*, i.e. an expression that describes format or formats you would like to download.

tl;dr: [navigate me to examples.](#)

The simplest case is requesting a specific format; e.g. with `-f 22` you can download the format with format code equal to 22. You can get the list of available format codes for particular video using `--list-formats` or `-F`. Note that these format codes are extractor specific.

You can also use a file extension (currently `3gp`, `aac`, `flv`, `m4a`, `mp3`, `mp4`, `ogg`, `wav`, `webm` are supported) to download the best quality format of a particular file extension served as a single file, e.g. `-f webm` will download the best quality format with the `webm` extension served as a single file.

You can use `-f -` to interactively provide the format selector *for each video*

You can also use special names to select particular edge case formats:

- `all`: Select **all formats** separately
- `mergeall`: Select and **merge all formats** (Must be used with `--audio-multistreams`, `--video-multistreams` or both)
- `b*`, `best*`: Select the best quality format that **contains either** a video or an audio or both (ie; `vcodec!=none` or `acodec!=none`)
- `b`, `best`: Select the best quality format that **contains both** video and audio. Equivalent to `best*[vcodec!=none][acodec!=none]`
- `bv`, `bestvideo`: Select the best quality **video-only** format. Equivalent to `best*[acodec=none]`
- `bv*`, `bestvideo*`: Select the best quality format that **contains video**. It may also contain audio. Equivalent to `best*[vcodec!=none]`
- `ba`, `bestaudio`: Select the best quality **audio-only** format. Equivalent to `best*[vcodec=none]`

- `ba*`, `bestaudio*`: Select the best quality format that **contains audio**. It may also contain video. Equivalent to `best*` `[acodec!=none]` (Do not use!)
- `w*`, `worst*`: Select the worst quality format that contains either a video or an audio
- `w`, `worst`: Select the worst quality format that contains both video and audio. Equivalent to `worst*` `[vcodec!=none]` `[acodec!=none]`
- `wv`, `worstvideo`: Select the worst quality video-only format. Equivalent to `worst*` `[acodec=none]`
- `wv*`, `worstvideo*`: Select the worst quality format that contains video. It may also contain audio. Equivalent to `worst*` `[vcodec!=none]`
- `wa`, `worstaudio`: Select the worst quality audio-only format. Equivalent to `worst*` `[vcodec=none]`
- `wa*`, `worstaudio*`: Select the worst quality format that contains audio. It may also contain video. Equivalent to `worst*` `[acodec!=none]`

For example, to download the worst quality video-only format you can use `-f worstvideo`. It is however recommended not to use `worst` and related options. When your format selector is `worst`, the format which is worst in all respects is selected. Most of the time, what you actually want is the video with the smallest filesize instead. So it is generally better to use `-S +size` or more rigorously, `-S +size,+br,+res,+fps` instead of `-f worst`. See [Sorting Formats](#) for more details.

You can select the n'th best format of a type by using `best<type>.<n>`. For example, `best.2` will select the 2nd best combined format. Similarly, `bv*.3` will select the 3rd best format that contains a video stream.

If you want to download multiple videos, and they don't have the same formats available, you can specify the order of preference using slashes. Note that formats on the left hand side are preferred; e.g. `-f 22/17/18` will download format 22 if it's available, otherwise it will download format 17 if it's available, otherwise it will download format 18 if it's available, otherwise it will complain that no suitable formats are available for download.

If you want to download several formats of the same video use a comma as a separator, e.g. `-f 22,17,18` will download all these three formats, of course if they are available. Or a more sophisticated example combined with the precedence feature: `-f 136/137/mp4/bestvideo,140/m4a/bestaudio`.

You can merge the video and audio of multiple formats into a single file using `-f <format1>+<format2>+...` (requires ffmpeg installed); e.g. `-f bestvideo+bestaudio` will download the best video-only format, the best audio-only format and mux them together with ffmpeg.

Deprecation warning: Since the *below* described behavior is complex and counter-intuitive, this will be removed and multistreams will be enabled by default in the future. A new operator will be instead added to limit formats to single audio/video

Unless `--video-multistreams` is used, all formats with a video stream except the first one are ignored. Similarly, unless `--audio-multistreams` is used, all formats with an audio stream except the first one are ignored. E.g. `-f bestvideo+best+bestaudio --video-multistreams --audio-multistreams` will download and merge all 3 given formats. The resulting file will have 2 video streams and 2 audio streams. But `-f bestvideo+best+bestaudio --no-video-multistreams` will download and merge only `bestvideo` and `bestaudio`. `best` is ignored since another format containing a video stream (`bestvideo`) has already been selected. The order of the

formats is therefore important. `-f best+bestaudio --no-audio-multistreams` will download only `best` while `-f bestaudio+best --no-audio-multistreams` will ignore `best` and download only `bestaudio`.

Filtering Formats

You can also filter the video formats by putting a condition in brackets, as in `-f "best[height=720]"` (or `-f "[filesize>10M]"` since filters without a selector are interpreted as `best`).

The following numeric meta fields can be used with comparisons `<`, `<=`, `>`, `>=`, `=` (equals), `!=` (not equals):

- `filesize`: The number of bytes, if known in advance
- `filesize_approx`: An estimate for the number of bytes
- `width`: Width of the video, if known
- `height`: Height of the video, if known
- `aspect_ratio`: Aspect ratio of the video, if known
- `tbr`: Average bitrate of audio and video in KBit/s
- `abr`: Average audio bitrate in KBit/s
- `vbr`: Average video bitrate in KBit/s
- `asr`: Audio sampling rate in Hertz
- `fps`: Frame rate
- `audio_channels`: The number of audio channels
- `stretched_ratio`: `width:height` of the video's pixels, if not square

Also filtering work for comparisons `=` (equals), `^=` (starts with), `$=` (ends with), `*=` (contains), `~=` (matches regex) and following string meta fields:

- `url`: Video URL
- `ext`: File extension
- `acodec`: Name of the audio codec in use
- `vcodec`: Name of the video codec in use

- **container**: Name of the container format
- **protocol**: The protocol that will be used for the actual download, lower-case (`http`, `https`, `rtsp`, `rtmp`, `rtmpe`, `mms`, `f4m`, `ism`, `http_dash_segments`, `m3u8`, or `m3u8_native`)
- **language**: Language code
- **dynamic_range**: The dynamic range of the video
- **format_id**: A short description of the format
- **format**: A human-readable description of the format
- **format_note**: Additional info about the format
- **resolution**: Textual description of width and height

Any string comparison may be prefixed with negation `!` in order to produce an opposite comparison, e.g. `!*=` (does not contain). The comparand of a string comparison needs to be quoted with either double or single quotes if it contains spaces or special characters other than `._-`.

Note: None of the aforementioned meta fields are guaranteed to be present since this solely depends on the metadata obtained by particular extractor, i.e. the metadata offered by the website. Any other field made available by the extractor can also be used for filtering.

Formats for which the value is not known are excluded unless you put a question mark (`?`) after the operator. You can combine format filters, so `-f "bv[height<=?720][tbr>500]"` selects up to 720p videos (or videos where the height is not known) with a bitrate of at least 500 KBit/s. You can also use the filters with `all` to download all formats that satisfy the filter, e.g. `-f "all[vcodec=none]"` selects all audio-only formats.

Format selectors can also be grouped using parentheses; e.g. `-f "(mp4,webm)[height<480]"` will download the best pre-merged mp4 and webm formats with a height lower than 480.

Sorting Formats

You can change the criteria for being considered the **best** by using **-S** (**--format-sort**). The general format for this is **--format-sort field1,field2....**

The available fields are:

- **hasvid**: Gives priority to formats that have a video stream
- **hasaud**: Gives priority to formats that have an audio stream
- **ie_pref**: The format preference
- **lang**: The language preference
- **quality**: The quality of the format
- **source**: The preference of the source
- **proto**: Protocol used for download (**https/https > http/ftp > m3u8_native/m3u8 > http_dash_segments > websocket_frag > mms/rtsp > f4f/f4m**)
- **vcodec**: Video Codec (**av01 > vp9.2 > vp9 > h265 > h264 > vp8 > h263 > theora > other**)
- **acodec**: Audio Codec (**flac/alac > wav/aiff > opus > vorbis > aac > mp4a > mp3 > ac4 > eac3 > ac3 > dts > other**)
- **codec**: Equivalent to **vcodec,acodec**
- **vext**: Video Extension (**mp4 > mov > webm > flv > other**). If **--prefer-free-formats** is used, **webm** is preferred.
- **aext**: Audio Extension (**m4a > aac > mp3 > ogg > opus > webm > other**). If **--prefer-free-formats** is used, the order changes to **ogg > opus > webm > mp3 > m4a > aac**
- **ext**: Equivalent to **vext,aext**
- **filesize**: Exact filesize, if known in advance
- **fs_approx**: Approximate filesize
- **size**: Exact filesize if available, otherwise approximate filesize
- **height**: Height of video
- **width**: Width of video
- **res**: Video resolution, calculated as the smallest dimension.
- **fps**: Framerate of video

- `hdr`: The dynamic range of the video (`DV > HDR12 > HDR10+ > HDR10 > HLG > SDR`)
- `channels`: The number of audio channels
- `tbr`: Total average bitrate in KBit/s
- `vbr`: Average video bitrate in KBit/s
- `abr`: Average audio bitrate in KBit/s
- `br`: Average bitrate in KBit/s, `tbr/vbr/abr`
- `asr`: Audio sample rate in Hz

Deprecation warning: Many of these fields have (currently undocumented) aliases, that may be removed in a future version. It is recommended to use only the documented field names.

All fields, unless specified otherwise, are sorted in descending order. To reverse this, prefix the field with a `+`. E.g. `+res` prefers format with the smallest resolution. Additionally, you can suffix a preferred value for the fields, separated by a `:`. E.g. `res:720` prefers larger videos, but no larger than 720p and the smallest video if there are no videos less than 720p. For `codec` and `ext`, you can provide two preferred values, the first for video and the second for audio. E.g.

`+codec:avc:m4a` (equivalent to `+vcodec:avc,+acodec:m4a`) sets the video codec preference to `h264 > h265 > vp9 > vp9.2 > av01 > vp8 > h263 > theora` and audio codec preference to `mp4a > aac > vorbis > opus > mp3 > ac3 > dts`. You can also make the sorting prefer the nearest values to the provided by using `~` as the delimiter. E.g. `filesize~1G` prefers the format with filesize closest to 1 GiB.

The fields `hasvid` and `ie_pref` are always given highest priority in sorting, irrespective of the user-defined order. This behaviour can be changed by using `--format-sort-force`. Apart from these, the default order used is:

`lang, quality, res, fps, hdr:12, vcodec:vp9.2, channels, acodec, size, br, asr, proto, ext, hasaud, source, id`. The extractors may override this default order, but they cannot override the user-provided order.

Note that the default has `vcodec:vp9.2`; i.e. `av1` is not preferred. Similarly, the default for `hdr:12`; i.e. dolby vision is not preferred. These choices are made since DV and AV1 formats are not yet fully compatible with most devices. This may be changed in the future as more devices become capable of smoothly playing back these formats.

If your format selector is `worst`, the last item is selected after sorting. This means it will select the format that is worst in all respects. Most of the time, what you actually want is the video with the smallest filesize instead. So it is generally better to use `-f best -S +size, +br, +res, +fps`.

Tip: You can use the `-v -F` to see how the formats have been sorted (worst to best).

Format Selection examples

```
# Download and merge the best video-only format and the best
audio-only format,
# or download the best combined format if video-only format is
not available
$ yt-dlp -f "bv+ba/b"
```

```
# Download best format that contains video,
# and if it doesn't already have an audio stream, merge it with
best audio-only format
$ yt-dlp -f "bv*+ba/b"
```

```
# Same as above
$ yt-dlp
```

```
# Download the best video-only format and the best audio-only
format without merging them
# For this case, an output template should be used since
# by default, bestvideo and bestaudio will have the same file
name.
$ yt-dlp -f "bv,ba" -o "%(title)s.f%(format_id)s.%(ext)s"
```

```
# Download and merge the best format that has a video stream,
# and all audio-only formats into one file
$ yt-dlp -f "bv*+mergeall[vcodec=none]" --audio-multistreams
```

```
# Download and merge the best format that has a video stream,
# and the best 2 audio-only formats into one file
$ yt-dlp -f "bv*+ba+ba.2" --audio-multistreams
```

```
# The following examples show the old method (without -S) of
format selection
# and how to use -S to achieve a similar but (generally) better
result
```

```
# Download the worst video available (old method)
$ yt-dlp -f "wv*+wa/w"
```

```
# Download the best video available but with the smallest
resolution
$ yt-dlp -S "+res"
```

```
# Download the smallest video available
```

```
$ yt-dlp -S "+size,+br"
```

```
# Download the best mp4 video available, or the best video if  
no mp4 available
```

```
$ yt-dlp -f "bv*[ext=mp4]+ba[ext=m4a]/b[ext=mp4] / bv*+ba/b"
```

```
# Download the best video with the best extension
```

```
# (For video, mp4 > mov > webm > flv. For audio, m4a > aac >  
mp3 ...)
```

```
$ yt-dlp -S "ext"
```

```
# Download the best video available but no better than 480p,  
# or the worst video if there is no video under 480p
```

```
$ yt-dlp -f "bv*[height<=480]+ba/b[height<=480] / wv*+ba/w"
```

```
# Download the best video available with the largest height but  
no better than 480p,
```

```
# or the best video with the smallest resolution if there is no  
video under 480p
```

```
$ yt-dlp -S "height:480"
```

```
# Download the best video available with the largest resolution  
but no better than 480p,
```

```
# or the best video with the smallest resolution if there is no  
video under 480p
```

```
# Resolution is determined by using the smallest dimension.
```

```
# So this works correctly for vertical videos as well
```

```
$ yt-dlp -S "res:480"
```

```
# Download the best video (that also has audio) but no bigger  
than 50 MB,
```

```
# or the worst video (that also has audio) if there is no video  
under 50 MB
```

```
$ yt-dlp -f "b[filesize<50M] / w"
```

```
# Download largest video (that also has audio) but no bigger  
than 50 MB,
```

```

# or the smallest video (that also has audio) if there is no
video under 50 MB
$ yt-dlp -f "b" -S "filesize:50M"

# Download best video (that also has audio) that is closest in
size to 50 MB
$ yt-dlp -f "b" -S "filesize~50M"

# Download best video available via direct link over HTTP/HTTPS
protocol,
# or the best video available via any protocol if there is no
such video
$ yt-dlp -f "(bv*+ba/b)[protocol^=http][protocol!*=dash] /
(bv*+ba/b)"

# Download best video available via the best protocol
# (https/ftps > http/ftp > m3u8_native > m3u8 >
http_dash_segments ...)
$ yt-dlp -S "proto"

# Download the best video with either h264 or h265 codec,
# or the best video if there is no such video
$ yt-dlp -f "(bv*[vcodec~='^((he|a)vc|h26[45])']+ba) /
(bv*+ba/b)"

# Download the best video with best codec no better than h264,
# or the best video with worst codec if there is no such video
$ yt-dlp -S "codec:h264"

# Download the best video with worst codec no worse than h264,
# or the best video with best codec if there is no such video
$ yt-dlp -S "+codec:h264"

# More complex examples

# Download the best video no better than 720p preferring
framerate greater than 30,

```

```
# or the worst video (still preferring framerate greater than
30) if there is no such video
$ yt-dlp -f "((bv*[fps>30]/bv*)[height<=720]/(wv*[fps>30]/wv*))
+ ba / (b[fps>30]/b)[height<=720]/(w[fps>30]/w)"

# Download the video with the largest resolution no better than
720p,
# or the video with the smallest resolution available if there
is no such video,
# preferring larger framerate for formats with the same
resolution
$ yt-dlp -S "res:720,fps"

# Download the video with smallest resolution no worse than
480p,
# or the video with the largest resolution available if there
is no such video,
# preferring better codec and then larger total bitrate for the
same resolution
$ yt-dlp -S "+res:480,codec,br"
```

MODIFYING METADATA

The metadata obtained by the extractors can be modified by using `-parse-metadata` and `--replace-in-metadata`

`--replace-in-metadata FIELDS REGEX REPLACE` is used to replace text in any metadata field using python regular expression.

Backreferences can be used in the replace string for advanced use.

The general syntax of `--parse-metadata FROM:TO` is to give the name of a field or an output template to extract data from, and the format to interpret it as, separated by a colon `:`. Either a python regular expression with named capture groups, a single field name, or a similar syntax to the output template (only `%(field)s` formatting is supported) can be used for `TO`. The option can be used multiple times to parse and modify various fields.

Note that these options preserve their relative order, allowing replacements to be made in parsed fields and viceversa. Also, any field thus created can be used in the output template and will also affect the media file's metadata added when using `--embed-metadata`.

This option also has a few special uses:

- You can download an additional URL based on the metadata of the currently downloaded video. To do this, set the field `additional_urls` to the URL that you want to download. E.g. `--parse-metadata "description:(?P<additional_urls>https?://www\.vimeo\.com/\d+)"` will download the first vimeo video found in the description
- You can use this to change the metadata that is embedded in the media file. To do this, set the value of the corresponding field with a `meta_` prefix. For example, any value you set to `meta_description` field will be added to the `description` field in the file - you can use this to set a different "description" and "synopsis". To modify the metadata of individual streams, use the `meta<n>_` prefix (e.g. `meta1_language`). Any value set to the `meta_` field will overwrite all default values.

Note: Metadata modification happens before format selection, post-extraction and other post-processing operations. Some fields may be added or changed during these steps, overriding your changes.

For reference, these are the fields yt-dlp adds by default to the file metadata:

Metadata fields	From
<code>title</code>	<code>track</code> or <code>title</code>
<code>date</code>	<code>upload_date</code>
<code>description, synopsis</code>	<code>description</code>
<code>purl, comment</code>	<code>webpage_url</code>

Metadata fields	From
track	track_number
artist	artist, creator, uploader or uploader_id
genre	genre
album	album
album_artist	album_artist
disc	disc_number
show	series
season_number	season_number
episode_id	episode or episode_id
episode_sort	episode_number
language of each stream	the format's language

Note: The file format may not support some of these fields

Modifying metadata examples

```

# Interpret the title as "Artist - Title"
$ yt-dlp --parse-metadata "title:%(artist)s - %(title)s"

# Regex example
$ yt-dlp --parse-metadata "description:Artist - (?P<artist>.+)"

# Set title as "Series name S01E05"
$ yt-dlp --parse-metadata "%(series)s S%(season_number)02dE%(episode_number)02d:%(title)s"

# Prioritize uploader as the "artist" field in video metadata
$ yt-dlp --parse-metadata "%(uploader|)s:%(meta_artist)s" --embed-metadata

# Set "comment" field in video metadata using description instead of webpage_url,
# handling multiple lines correctly
$ yt-dlp --parse-metadata "description:(?s)(?P<meta_comment>.+)" --embed-metadata

# Do not set any "synopsis" in the video metadata
$ yt-dlp --parse-metadata ":(?P<meta_synopsis>)"

# Remove "formats" field from the infojson by setting it to an empty string
$ yt-dlp --parse-metadata "video::(?P<formats>)" --write-info-json

# Replace all spaces and "_" in title and uploader with a ` - `
$ yt-dlp --replace-in-metadata "title,uploader" "[ _]" "-"

```

EXTRACTOR ARGUMENTS

Some extractors accept additional arguments which can be passed using `--extractor-args KEY:ARGS`. `ARGS` is a ; (semicolon) separated string of `ARG=VAL1,VAL2`. E.g. `--extractor-args "youtube:player-client=android_embedded,web;include_live_dash" --extractor-args "funimation:version=uncut"`

Note: In CLI, `ARG` can use `-` instead of `_`; e.g. `youtube:player-client` becomes `youtube:player_client`

The following extractors use this feature:

youtube

- `lang`: Prefer translated metadata (`title`, `description` etc) of this language code (case-sensitive). By default, the video primary language metadata is preferred, with a fallback to `en` translated. See `youtube.py` for list of supported content language codes
- `skip`: One or more of `hls`, `dash` or `translated_subs` to skip extraction of the m3u8 manifests, dash manifests and auto-translated subtitles respectively
- `player_client`: Clients to extract video data from. The main clients are `web`, `android` and `ios` with variants `_music`, `_embedded`, `_embedscreen`, `_creator` (e.g. `web_embedded`); and `mweb` and `tv_embedded` (agegate bypass) with no variants. By default, `ios`, `android`, `web` is used, but `tv_embedded` and `creator` variants are added as required for age-gated videos. Similarly, the music variants are added for `music.youtube.com` urls. You can use `all` to use all the clients, and `default` for the default clients.
- `player_skip`: Skip some network requests that are generally needed for robust extraction. One or more of `configs` (skip client configs), `webpage` (skip initial webpage), `js` (skip js player). While these options can help reduce the number of requests needed or avoid some rate-limiting, they could cause some issues. See [#860](#) for more details
- `player_params`: YouTube player parameters to use for player requests. Will overwrite any default ones set by yt-dlp.
- `comment_sort`: `top` or `new` (default) - choose comment sorting mode (on YouTube's side)

- **max_comments**: Limit the amount of comments to gather. Comma-separated list of integers representing **max_comments**, **max_parents**, **max_replies**, **max_replies_per_thread**. Default is **all, all, all, all**
E.g. **all, all, 1000, 10** will get a maximum of 1000 replies total, with up to 10 replies per thread. **1000, all, 100** will get a maximum of 1000 comments, with a maximum of 100 replies total
- **formats**: Change the types of formats to return. **dashy** (convert HTTP to DASH), **duplicate** (identical content but different URLs or protocol; includes **dashy**), **incomplete** (cannot be downloaded completely - live dash and post-live m3u8)
- **innertube_host**: Innertube API host to use for all API requests; e.g. **studio.youtube.com**, **youtubei.googleapis.com**. Note that cookies exported from one subdomain will not work on others
- **innertube_key**: Innertube API key to use for all API requests

youtubetab (YouTube playlists, channels, feeds, etc.)

- **skip**: One or more of **webpage** (skip initial webpage download), **authcheck** (allow the download of playlists requiring authentication when no initial webpage is downloaded. This may cause unwanted behavior, see [#1122](#) for more details)
- **approximate_date**: Extract approximate **upload_date** and **timestamp** in flat-playlist. This may cause date-based filters to be slightly off

generic

- **fragment_query**: Passthrough any query in mpd/m3u8 manifest URLs to their fragments if no value is provided, or else apply the query string given as **fragment_query=VALUE**. Does not apply to ffmpeg
- **variant_query**: Passthrough the master m3u8 URL query to its variant playlist URLs if no value is provided, or else apply the query string given as **variant_query=VALUE**

- `hls_key`: An HLS AES-128 key URI or key (as hex), and optionally the IV (as hex), in the form of `(URI|KEY)[, IV]`; e.g. `generic:hls_key=ABCDEF1234567980,0xFEDCBA0987654321`. Passing any of these values will force usage of the native HLS downloader and override the corresponding values found in the m3u8 playlist
- `is_live`: Bypass live HLS detection and manually set `live_status` - a value of `false` will set `not_live`, any other value (or no value) will set `is_live`

funimation

- `language`: Audio languages to extract, e.g. `funimation:language=english,japanese`
- `version`: The video version to extract - `uncut` or `simulcast`

crunchyrollbeta (Crunchyroll)

- `format`: Which stream type(s) to extract (default: `adaptive_hls`). Potentially useful values include `adaptive_hls`, `adaptive_dash`, `vo_adaptive_hls`, `vo_adaptive_dash`, `download_hls`, `download_dash`, `multitrack_adaptive_hls_v2`
- `hardsub`: Preference order for which hardsub versions to extract, or `all` (default: `None` = no hardsubs), e.g. `crunchyrollbeta:hardsub=en-US, None`

vikichannel

`video_types`: Types of videos to download - one or more of `episodes`, `movies`, `clips`, `trailers`

niconico

`segment_duration`: Segment duration in milliseconds for HLS-DMC formats. Use it at your own risk since this feature **may result in your account termination**.

youtubewebarchive

`check_all`: Try to check more at the cost of more requests. One or more of `thumbnails`, `captures`

gamejolt

`comment_sort`: `hot` (default), `you` (cookies needed), `top`, `new` - choose comment sorting mode (on GameJolt's side)

hotstar

- `res`: resolution to ignore - one or more of `sd`, `hd`, `fhd`
- `vcodec`: vcodec to ignore - one or more of `h264`, `h265`, `dvh265`
- `dr`: dynamic range to ignore - one or more of `sdr`, `hdr10`, `dv`

tiktok

- `api_hostname`: Hostname to use for mobile API requests, e.g. `api-h2.tiktokv.com`
- `app_version`: App version to call mobile APIs with - should be set along with `manifest_app_version`, e.g. `20.2.1`
- `manifest_app_version`: Numeric app version to call mobile APIs with, e.g. `221`

rokfinchannel

`tab`: Which tab to download - one of `new`, `top`, `videos`, `podcasts`, `streams`, `stacks`

twitter

`legacy_api`: Force usage of the legacy Twitter API instead of the GraphQL API for tweet extraction. Has no effect if login cookies are passed

stacommunity, wrestleuniverse

`device_id`: UUID value assigned by the website and used to enforce device limits for paid livestream content. Can be found in browser local storage

twitch

`client_id`: Client ID value to be sent with GraphQL requests, e.g. `twitch:client_id=kimne78kx3ncx6brgo4mv6wki5h1ko`

`nhkradirulive` (NHK らじる★らじる LIVE)

`area`: Which regional variation to extract. Valid areas are: `sapporo`, `sendai`, `tokyo`, `nagoya`, `osaka`, `hiroshima`, `matsuyama`, `fukuoka`. Defaults to `tokyo`

Note: These options may be changed/removed in the future without concern for backward compatibility

PLUGINS

Note that **all** plugins are imported even if not invoked, and that **there are no checks** performed on plugin code. **Use plugins at your own risk and only if you trust the code!**

Plugins can be of `<type>s` `extractor` or `postprocessor`.

- Extractor plugins do not need to be enabled from the CLI and are automatically invoked when the input URL is suitable for it.
- Extractor plugins take priority over builtin extractors.
- Postprocessor plugins can be invoked using `--use-postprocessor NAME`.

Plugins are loaded from the namespace packages

`yt_dlp_plugins.extractor` and `yt_dlp_plugins.postprocessor`.

In other words, the file structure on the disk looks something like:

```
yt_dlp_plugins/  
  extractor/  
    myplugin.py  
  postprocessor/  
    myplugin.py
```

yt-dlp looks for these `yt_dlp_plugins` namespace folders in many locations (see below) and loads in plugins from **all** of them.

See the [wiki](#) for some known [plugins](#)

Installing Plugins

Plugins can be installed using various methods and locations.

1. **Configuration directories:** Plugin packages (containing a `yt_dlp_plugins` namespace folder) can be dropped into the following standard configuration locations:

- **User Plugins**

- `${XDG_CONFIG_HOME}/yt-dlp/plugins/<package name>/yt_dlp_plugins/` (recommended on Linux/macOS)
- `${XDG_CONFIG_HOME}/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
- `${APPDATA}/yt-dlp/plugins/<package name>/yt_dlp_plugins/` (recommended on Windows)
- `${APPDATA}/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
- `~/.yt-dlp/plugins/<package name>/yt_dlp_plugins/`
- `~/yt-dlp-plugins/<package name>/yt_dlp_plugins/`

- **System Plugins**

- `/etc/yt-dlp/plugins/<package name>/yt_dlp_plugins/`
- `/etc/yt-dlp-plugins/<package name>/yt_dlp_plugins/`

2. **Executable location:** Plugin packages can similarly be installed in a `yt-dlp-plugins` directory under the executable location (recommended for portable installations):

- Binary: where `<root-dir>/yt-dlp.exe`, `<root-dir>/yt-dlp-plugins/<package name>/yt_dlp_plugins/`
- Source: where `<root-dir>/yt_dlp/__main__.py`, `<root-dir>/yt-dlp-plugins/<package name>/yt_dlp_plugins/`

3. pip and other locations in `PYTHONPATH`

- Plugin packages can be installed and managed using `pip`. See [yt-dlp-sample-plugins](#) for an example.

Note: plugin files between plugin packages installed with pip must have unique filenames.

- Any path in `PYTHONPATH` is searched in for the `yt_dlp_plugins` namespace folder.

Note: This does not apply for Pyinstaller/py2exe builds.

`.zip`, `.egg` and `.whl` archives containing a `yt_dlp_plugins` namespace folder in their root are also supported as plugin packages.

e.g. `${XDG_CONFIG_HOME}/yt-dlp/plugins/mypluginpkg.zip`
where `mypluginpkg.zip` contains
`yt_dlp_plugins/<type>/myplugin.py`

Run `yt-dlp` with `--verbose` to check if the plugin has been loaded.

Developing Plugins

See the [yt-dlp-sample-plugins](#) repo for a template plugin package and the [Plugin Development](#) section of the wiki for a plugin development guide.

All public classes with a name ending in `IE/PP` are imported from each file for extractors and postprocessors respectively. This respects underscore prefix (e.g. `_MyBasePluginIE` is private) and `__all__`.

Modules can similarly be excluded by prefixing the module name with an underscore (e.g. `_myplugin.py`).

To replace an existing extractor with a subclass of one, set the `plugin_name` class keyword argument (e.g. `class MyPluginIE(ABuiltInIE, plugin_name='myplugin')` will replace `ABuiltInIE` with `MyPluginIE`). Since the extractor replaces the parent, you should exclude the subclass extractor from being imported separately by making it private using one of the methods described above.

If you are a plugin author, add `yt-dlp-plugins` as a topic to your repository for discoverability.

See the [Developer Instructions](#) on how to write and test an extractor.

EMBEDDING YT-DLP

yt-dlp makes the best effort to be a good command-line program, and thus should be callable from any programming language.

Your program should avoid parsing the normal stdout since they may change in future versions. Instead they should use options such as `-J`, `--print`, `--progress-template`, `--exec` etc to create console output that you can reliably reproduce and parse.

From a Python program, you can embed yt-dlp in a more powerful fashion, like this:

```
from yt_dlp import YoutubeDL

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']
with YoutubeDL() as ydl:
    ydl.download(URLS)
```

Most likely, you'll want to use various options. For a list of options available, have a look at `yt_dlp/YoutubeDL.py` or `help(yt_dlp.YoutubeDL)` in a Python shell. If you are already

familiar with the CLI, you can use `devscripts/cli_to_api.py` to translate any CLI switches to `YoutubeDL` params.

Tip: If you are porting your code from `youtube-dl` to `yt-dlp`, one important point to look out for is that we do not guarantee the return value of `YoutubeDL.extract_info` to be json serializable, or even be a dictionary. It will be dictionary-like, but if you want to ensure it is a serializable dictionary, pass it through `YoutubeDL.sanitize_info` as shown in the example below

Embedding examples

Extracting information

```
import json
import yt_dlp

URL = 'https://www.youtube.com/watch?v=BaW_jenozKc'

# i See help(yt_dlp.YoutubeDL) for a list of available options
and public functions
ydl_opts = {}
with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    info = ydl.extract_info(URL, download=False)

    # i ydl.sanitize_info makes the info json-serializable
    print(json.dumps(ydl.sanitize_info(info)))
```

Download using an info-json

```
import yt_dlp

INFO_FILE = 'path/to/video.info.json'

with yt_dlp.YoutubeDL() as ydl:
    error_code = ydl.download_with_info_file(INFO_FILE)

print('Some videos failed to download' if error_code
      else 'All videos successfully downloaded')
```

Extract audio

```
import yt_dlp

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']

ydl_opts = {
    'format': 'm4a/bestaudio/best',
    # i See help(yt_dlp.postprocessor) for a list of available
    Postprocessors and their arguments
    'postprocessors': [{ # Extract audio using ffmpeg
        'key': 'FFmpegExtractAudio',
        'preferredcodec': 'm4a',
    }]
}

with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    error_code = ydl.download(URLS)
```

Filter videos

```
import yt_dlp

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']

def longer_than_a_minute(info, *, incomplete):
    """Download only videos longer than a minute (or with
    unknown duration)"""
    duration = info.get('duration')
    if duration and duration < 60:
        return 'The video is too short'

ydl_opts = {
    'match_filter': longer_than_a_minute,
}

with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    error_code = ydl.download(URLS)
```

Adding logger and progress hook

```

import yt_dlp

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']

class MyLogger:
    def debug(self, msg):
        # For compatibility with youtube-dl, both debug and
        # info are passed into debug
        # You can distinguish them by the prefix '[debug] '
        if msg.startswith('[debug] '):
            pass
        else:
            self.info(msg)

    def info(self, msg):
        pass

    def warning(self, msg):
        pass

    def error(self, msg):
        print(msg)

# i See "progress_hooks" in help(yt_dlp.YoutubeDL)
def my_hook(d):
    if d['status'] == 'finished':
        print('Done downloading, now post-processing ...')

ydl_opts = {
    'logger': MyLogger(),
    'progress_hooks': [my_hook],
}

with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    ydl.download(URLS)

```

Add a custom PostProcessor

```
import yt_dlp

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']

# i See help(yt_dlp.postprocessor.PostProcessor)
class MyCustomPP(yt_dlp.postprocessor.PostProcessor):
    def run(self, info):
        self.to_screen('Doing stuff')
        return [], info

with yt_dlp.YoutubeDL() as ydl:
    # i "when" can take any value in
    yt_dlp.utils.POSTPROCESS_WHEN
    ydl.add_post_processor(MyCustomPP(), when='pre_process')
    ydl.download(URLS)
```

Use a custom format selector

```

import yt_dlp

URLS = ['https://www.youtube.com/watch?v=BaW_jenozKc']

def format_selector(ctx):
    """ Select the best video and the best audio that won't
    result in an mkv.
    NOTE: This is just an example and does not handle all cases
    """

    # formats are already sorted worst to best
    formats = ctx.get('formats')[::-1]

    # acodec='none' means there is no audio
    best_video = next(f for f in formats
                      if f['vcodec'] != 'none' and f['acodec']
== 'none')

    # find compatible audio extension
    audio_ext = {'mp4': 'm4a', 'webm': 'webm'}
[best_video['ext']]
    # vcodec='none' means there is no video
    best_audio = next(f for f in formats if (
        f['acodec'] != 'none' and f['vcodec'] == 'none' and
f['ext'] == audio_ext))

    # These are the minimum required fields for a merged format
    yield {
        'format_id': f'{best_video["format_id"]}'
+
{best_audio["format_id"]}',
        'ext': best_video['ext'],
        'requested_formats': [best_video, best_audio],
        # Must be + separated list of protocols
        'protocol': f'{best_video["protocol"]}'
+
{best_audio["protocol"]}'
    }

ydl_opts = {
    'format': format_selector,
}

```



```
with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    ydl.download(URLS)
```

DEPRECATED OPTIONS

These are all the deprecated options and the current alternative to achieve the same effect

Almost redundant options

While these options are almost the same as their new counterparts, there are some differences that prevents them being redundant

```
-j, --dump-json          --print "%()j"
-F, --list-formats      --print formats_table
--list-thumbnails      --print thumbnails_table --
print playlist:thumbnails_table
--list-subtitles        --print
automatic_captions_table --print subtitles_table
```

Redundant options

While these options are redundant, they are still expected to be used due to their ease of use

<code>--get-description</code>	<code>--print description</code>
<code>--get-duration</code>	<code>--print duration_string</code>
<code>--get-filename</code>	<code>--print filename</code>
<code>--get-format</code>	<code>--print format</code>
<code>--get-id</code>	<code>--print id</code>
<code>--get-thumbnail</code>	<code>--print thumbnail</code>
<code>-e, --get-title</code>	<code>--print title</code>
<code>-g, --get-url</code>	<code>--print urls</code>
<code>--match-title REGEX i)REGEX"</code>	<code>--match-filter "title =~ (? i)REGEX"</code>
<code>--reject-title REGEX i)REGEX"</code>	<code>--match-filter "title !~=? i)REGEX"</code>
<code>--min-views COUNT COUNT"</code>	<code>--match-filter "view_count >=? COUNT"</code>
<code>--max-views COUNT COUNT"</code>	<code>--match-filter "view_count <=? COUNT"</code>
<code>--break-on-reject</code>	Use <code>--break-match-filter</code>
<code>--user-agent UA</code>	<code>--add-header "User-Agent:UA"</code>
<code>--referer URL</code>	<code>--add-header "Referer:URL"</code>
<code>--playlist-start NUMBER</code>	<code>-I NUMBER:</code>
<code>--playlist-end NUMBER</code>	<code>-I :NUMBER</code>
<code>--playlist-reverse</code>	<code>-I ::-1</code>
<code>--no-playlist-reverse</code>	Default
<code>--no-colors</code>	<code>--color no_color</code>

Not recommended

While these options still work, their use is not recommended since there are other alternatives to achieve the same

--force-generic-extractor	--ies generic,default
--exec-before-download CMD	--exec "before_dl:CMD"
--no-exec-before-download	--no-exec
--all-formats	-f all
--all-subsubs	--sub-langs all --write-subsubs
--print-json	-j --no-simulate
--autonumber-size NUMBER (autonumber)03d	Use string formatting, e.g. %
--autonumber-start NUMBER like %(autonumber+NUMBER)s	Use internal field formatting
--id	-o "%(id)s.%(ext)s"
--metadata-from-title FORMAT (title)s:FORMAT"	--parse-metadata "%
--hls-prefer-native	--downloader "m3u8:native"
--hls-prefer-ffmpeg	--downloader "m3u8:ffmpeg"
--list-formats-old (Alias: --no-list-formats-as-table)	--compat-options list-formats
--list-formats-as-table [Default] (Alias: --no-list-formats-old)	--compat-options -list-formats
--youtube-skip-dash-manifest "youtube:skip=dash" (Alias: --no-youtube-include-dash-manifest)	--extractor-args
--youtube-skip-hls-manifest "youtube:skip=hls" (Alias: --no-youtube-include-hls-manifest)	--extractor-args
--youtube-include-dash-manifest skip-dash-manifest)	Default (Alias: --no-youtube-
--youtube-include-hls-manifest skip-hls-manifest)	Default (Alias: --no-youtube-
--geo-bypass	--xff "default"
--no-geo-bypass	--xff "never"
--geo-bypass-country CODE	--xff CODE
--geo-bypass-ip-block IP_BLOCK	--xff IP_BLOCK

Developer options

These options are not intended to be used by the end-user

--test	Download only part of video
for testing extractors	
--load-pages	Load pages dumped by --write-
pages	
--youtube-print-sig-code	For testing youtube signatures
--allow-unplayable-formats	List unplayable formats also
--no-allow-unplayable-formats	Default

Old aliases

These are aliases that are no longer documented for various reasons

--avconv-location	--ffmpeg-location
--clean-infojson	--clean-info-json
--cn-verification-proxy URL	--geo-verification-proxy URL
--dump-headers	--print-traffic
--dump-intermediate-pages	--dump-pages
--force-write-download-archive	--force-write-archive
--load-info	--load-info-json
--no-clean-infojson	--no-clean-info-json
--no-split-tracks	--no-split-chapters
--no-write-srt	--no-write-subs
--prefer-unsecure	--prefer-insecure
--rate-limit RATE	--limit-rate RATE
--split-tracks	--split-chapters
--srt-lang LANGS	--sub-langs LANGS
--trim-file-names LENGTH	--trim-filenames LENGTH
--write-srt	--write-subs
--yes-overwrites	--force-overwrites

Sponskrub Options

Support for SponSkrub has been deprecated in favor of the `--sponsorblock` options

--sponskrub	--sponsorblock-mark all
--no-sponskrub	--no-sponsorblock
--sponskrub-cut	--sponsorblock-remove all
--no-sponskrub-cut	--sponsorblock-remove -all
--sponskrub-force	Not applicable
--no-sponskrub-force	Not applicable
--sponskrub-location	Not applicable
--sponskrub-args	Not applicable

No longer supported

These options may no longer work as intended

--prefer-avconv	avconv is not officially supported by yt-dlp (Alias: --no-prefer-ffmpeg)
--prefer-ffmpeg	Default (Alias: --no-prefer-avconv)
-C, --call-home	Not implemented
--no-call-home	Default
--include-ads	No longer supported
--no-include-ads	Default
--write-annotations	No supported site has annotations now
--no-write-annotations	Default
--compat-options separate-video-versions	No longer needed

Removed

These options were deprecated since 2014 and have now been entirely removed

-A, --auto-number (ext)s"	-o "%(autonumber)s-%(id)s.%(ext)s"
-t, -l, --title, --literal	-o "%(title)s-%(id)s.%(ext)s"

CONTRIBUTING

See [CONTRIBUTING.md](#) for instructions on [Opening an Issue](#) and [Contributing code to the project](#)

WIKI

See the [Wiki](#) for more information