Console command

Русский · User guide · Operations · Wrappers · AI agent

Contents

- Commands
- Running an operation
- The output
- Outcome and exit codes
- Where the result goes
- Wrappers
- Clearing the folders

`media_cli.cmd` in the root of the program runs what the buttons of the window run: an operation, with the same fields and the same presets. The window is the face of this command: whatever it starts, it starts through the command, and the first line of every run in its journal is a ready command that can be repeated in a console.

The command is for the cases where an operation is started not by a person at the window: a batch file, another tool, an AI agent, a long night of encodes.

Commands

Command	What it does
`describe`	The operations, their fields with choices and defaults, the presets and the layout of the window - one line of JSON.
`run`	Run an operation.
`wrapper NAME`	Run a wrapper of `Scripts\` by its name: `media_cli.cmd wrapper ff-x264-crf14`.

Command	What it does
`map`	The map of the program from its data: sections, operations, fields, presets, wrappers, the texts in both languages. `--out FILE` writes it to a file; `--docs` writes the reference documents (`OPERATIONS_*`, `SCRIPTS_*` and the tables of this guide).
`components`	The versions of yt-dlp, Deno, mpv, FFmpeg, 7-Zip and the GPU driver; `--latest` adds the newest releases (asked of GitHub at most once a day).
`plan --operation install_ffmpeg`	What an installer would download: the files, their source and size. Nothing is downloaded.
`version`	The version of the program.

`--root FOLDER` before the command names another copy of the program; `--language ru` or `en` - the language of the journal (the language of the window when not named).

Running an operation

```
.\media_cli.cmd run --operation storage_run
.\media_cli.cmd run --operation editing_run --set editing_profile=prores_lt_mov
.\media_cli.cmd run --operation storage_run --set storage_profile=hevc10_crf14 --set
encode_backend=cuda `
    --source "D:\Footage\Day 1" --target "D:\Encoded" --result
D:\Encoded\outcome.json
.\media_cli.cmd run --operation youtube_run --set
"youtube_url=https://www.youtube.com/watch?v=..."
```

Key	Purpose
`--operation ID`	The id of the operation. The table below gives the id of every mode of the window; `describe` gives all of them.
`--set NAME=VALUE`	The value of one field; may be repeated. A flag is `true` or `false`, a row of ticks - its values separated by commas (`--set target_codecs=x264,hevc_x265`). A preset field (`storage_profile`, `editing_profile` ...) sets the fields it names when it is named, and the other `--set` are laid over it; left unnamed, every field keeps its own default - the preset a form shows first is not applied by itself.

Key	Purpose
`--params FILE`	A JSON file "field: value". `--set` is laid over it.
`--source PATH`	A folder (its subfolders too) or one file. Without the key - `input` of the program.
`--target PATH`	A folder. Without the key - `output` of the program.
`--only KINDS`	The kinds of file to take from the Source: `mkv` or `mov,mxf`.
`--result FILE`	Where to write the outcome: a `.json` file outside the Source and outside `config\`. A full path, or call the command from the program folder.
`--cancel-file FILE`	The stop file: when it appears, FFmpeg is ended and a half-written file is taken away.
`--dry-run`	Run nothing: say the FFmpeg or yt-dlp commands the operation would start. An operation that cannot only say them is refused.

The names of the fields and their choices are those of `describe` and of the list of operations. A field that is not set goes with its default - as in a form of the window just opened. A relative `--source` or `--target` is counted from the folder of the program; a relative `--result`, `--params` or `--cancel-file` - from the folder the command is called from.

What each mode of the window is

Section	Mode	Call
AUDIO	EXTRACT	`media_cli.cmd run --operation audio_run --set audio_workflow=extract`
AUDIO	SOUND IN VIDEO	`media_cli.cmd run --operation audio_run --set audio_workflow=in_video`
AUDIO	CONVERT	`media_cli.cmd run --operation audio_run --set audio_workflow=audio_files`
ENCODE	ARCHIVE	`media_cli.cmd run --operation archive_run`

Section	Mode	Call
ENCODE	EDITING	`media_cli.cmd run --operation editing_run`
ENCODE	STORAGE	`media_cli.cmd run --operation storage_run`
ENCODE	DELIVERY	`media_cli.cmd run --operation delivery_run`
ENCODE	RAW	`media_cli.cmd run --operation raw_develop_run`
PACKAGE	VIDEO	`media_cli.cmd run --operation remux_run --set remux_stream_mode=video`
PACKAGE	AUDIO	`media_cli.cmd run --operation remux_run --set remux_stream_mode=audio`
PACKAGE	VIDEO AND AUDIO	`media_cli.cmd run --operation remux_run --set remux_stream_mode=video_audio`
PACKAGE	SUBTITLES	`media_cli.cmd run --operation subtitles_extract`
CUT	KEEP	`media_cli.cmd run --operation trim_run --set trim_action=keep`
CUT	CUT OUT	`media_cli.cmd run --operation trim_run --set trim_action=cut`
CUT	SPLIT	`media_cli.cmd run --operation trim_run --set trim_action=split`
FRAME RATE	VARISPEED	`media_cli.cmd run --operation fps_run --set fps_mode=varispeed`
FRAME RATE	CONFORM	`media_cli.cmd run --operation fps_run --set fps_mode=conform`
COLOR	LUT	`media_cli.cmd run --operation grading_run`
DOWNLOAD	BEST MP4	`media_cli.cmd run --operation youtube_run --set youtube_profile=best_mp4`
DOWNLOAD	SAFE MP4	`media_cli.cmd run --operation youtube_run --set youtube_profile=safe_mp4`
DOWNLOAD	OWN VIDEO	`media_cli.cmd run --operation youtube_run --set youtube_profile=video_custom`

Section	Mode	Call
DOWNLOAD	FOR UPLOAD	`media_cli.cmd run --operation youtube_run --set youtube_profile=upload_ready`
DOWNLOAD	OWN AUDIO	`media_cli.cmd run --operation youtube_run --set youtube_profile=audio_custom`
DOWNLOAD	SUBTITLES	`media_cli.cmd run --operation youtube_run --set youtube_profile=subs_only`

SOURCE INFO, SERVICE and the buttons beside a field:

Operation	What it is
`probe_source`	Probe Source: Run ffprobe once for Source, print a compact table and update the media status line.
`source_info`	Source Info: Run ffprobe for Source and print every audio stream: codec, sample format, sample rate, bit depth, channels, bitrate and tags.
`inventory`	Inventory: Count project folders.
`selftest`	Selftest: Check ffmpeg, ffprobe, yt-dlp and Deno.
`profile_doctor`	Profile Doctor: Validate GUI operations, script wrappers and the shared preset catalog.
`hardware_capabilities`	Hardware capabilities: Run short FFmpeg smoke tests for CPU, QSV/NVENC/AMF H.264 and HEVC, SVT-AV1 and dav1d.
`preset_test_matrix`	Preset Test Matrix: Run portable 1-second smoke tests across audio, remux, FPS, LUT, archive, editing, storage and delivery presets.
`install_7zip`	Install 7-Zip: Install or refresh portable 7-Zip; required before FFmpeg 7z payloads.
`install_ffmpeg`	Install FFmpeg: Install or refresh the portable FFmpeg of Gyan (GPL) in Tools\ffmpeg by the NVIDIA driver: the newest stable, 9.0.2 at least, without an NVIDIA card and from the LTS driver 610.88; below it 8.0.1 (570-610.87) or 7.1 so that NVENC works, and not every operation is possible.
`install_ytdlp`	Install yt-dlp: Install or refresh portable yt-dlp and Deno into Tools.

Operation	What it is
`install_mpv`	Install mpv: Install or refresh portable mpv into Tools/mpv: the player of the trim.
`install_cineraw`	Install ProRes RAW development: Build cineraw of libcraw into Tools/cineraw from its sources: a portable MSYS2 for the build (about 185 MB downloaded, 5 minutes), removed afterwards; cineraw develops ProRes RAW on a video card with Vulkan 1.4 (NVIDIA RTX).
`youtube_formats`	Formats of the link: What formats the link offers - resolution, codecs, size - listed in the journal, nothing downloaded. One of them is taken with the profile Own video.
`youtube_cookies_import`	Take under DPAPI: The cookies file of the field Cookies file sealed with DPAPI for this account of Windows into config\cookies.dpapi: downloads with Cookies - File take this copy, and the open file is no longer needed.

The output

The lines of the operation's journal, as in the window: `[OK]`, `[WARN]`, `[ERR]`, `[CMD]` - the command FFmpeg was started with, `[PREFLIGHT]` - the checks before the run, `[PROGRESS] clip.mp4 -> x264 [#####---] 42% 00:01:12/00:02:50 speed=3.4x` with FFmpeg's own line `frame= … fps= … time= … speed=` under it, and FFmpeg's last word `Lsize=` at the end. The same lines go to the journal file in `logs\`, and the report of the run - to `report\`.

A program that reads the output by a pipe also gets service lines: `::media::{"started": …}`, `{"progress": 0.42}`, `{"activity": …}`, `{"finished": {"outcome": …}}`. A console does not show them.

Outcome and exit codes

Code	Outcome	What it means
0	`done`	The operation finished.
1	`failed`	It ran and something broke, or a file failed or was refused: the journal says what.
2	- / `refused`	A wrong call - a field the operation has not, a value that is none of its choices, `--dry-run` of an operation that cannot only say its commands; or the checks before the run refused it - no file of its kind in the Source, no FFmpeg. Nothing was run.

Code	Outcome	What it means
130	`stopped`	Stopped by the stop file (`--cancel-file`), before the first file or during one.

The file of `--result`:

```
{
 "operation": "storage_run",
 "outcome": "done",
 "ok": true,
 "message": "Operation finished.",
 "data": {
  "processed": 1,
  "files": 1,
  "log_file": "...\\logs\\2026-10-15 22-34-54-804_storage_run.log",
  "batch_report_md": "...\\report\\2026-10-15 22-34-54-
804_storage_run\\batch_report.md"
 },
 "parameters": { "storage_profile": "x264_crf14", "target_codecs": ["x264"], "crf":
14, "...": "..." },
 "source": "D:\\Footage\\Day 1",
 "target": "D:\\Encoded",
 "seconds": 2.1
}
```

`parameters` are all the values the operation ran with, the defaults included: the run can be repeated from them exactly. `data` is what the operation tells: a media operation `processed` (files written), `skipped` (passed by, the reason in the journal), `files` (files of the Source), `failed` where it goes on past a failed file and `refused` where it turned a file down; a download `processed`, `downloaded` - the paths of what it wrote - and `existing`, found there already.

Where the result goes

Into a folder of the operation in the Target, and inside it a folder of the settings - named by the target and its values, not by the preset: `Storage\x264_crf14_slow\clip.mp4`, `Editing\prores422\clip.mov`, `Archive\…`, `Delivery\…`, `Audio\…`, `Remux\…`, `Trim\…`, `FPS\…`, `Color\…`, `Subtitles\…`; the subfolders of the Source are kept below it. `--dry-run` prints the exact path. Downloads go to `Download\<uploader>\<date> - <title> [<id>].<ext>` (a playlist: `Download\<number> - <title> [<id>].<ext>`).

A file already there is not written over unless `--set overwrite=true`: the journal says `[SKIP] NAME already exists and Overwrite is off`, FFmpeg is not started for it, and the outcome counts it in `skipped`, not in `processed`.

Wrappers

95 files of `Scripts\` - `ff-*`, `audio-*`, `yt-*` - each a ready task with its values, called by a double click, from a console or by name through the command: `media_cli.cmd wrapper ff-prores-lt-mov`. Each is listed with the same call through `run` in the list of wrappers. The wrappers of a trim take the file and the points from the variables `AUDION_TRIM_FILE`, `AUDION_TRIM_START`, `AUDION_TRIM_END`; the downloads - the links after the name, or `input\urls.txt`.

The menus `launcher_project.cmd`, `launcher_tools.cmd` and `cli\launcher-*.cmd` find a wrapper by typing (FZF).

Clearing the folders

The operations that clear the Source, the Target or the working folder delete for good and are not run from the command: the window clears them through the Recycle Bin.