Console command

Русский · User guide · Operations · Presets · AI agent

Contents

- Commands
- Running an operation
- A frame before and after
- The output
- Outcome and exit codes
- Where the result goes
- Stopping

`vs_cli.cmd` in the root of the program runs what the buttons of the window run: a preset, a profile or a service, with the same fields. 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 a preset is started not by a person at the window: a batch file, another tool, an AI agent, a long night of renders.

Commands

Command	What it does
`describe`	The program, the operations with their fields, choices and defaults, the layout of the window and what the machine has for the neural presets (`machine`: `rtx`, `mlrt`, `neural`) — one line of JSON.
`run`	Run an operation.
`frame`	One frame of a video before and after a preset with its values, as two PNG.

Command	What it does
`list presets` / `encoders` / `profiles` / `paths`	Lines to read: the presets with their defaults, the 34 encoders, the profiles, the folders of the program.
`map`	The map of the program from its data. `--out FILE` writes it to a file; `--docs` writes the reference documents (`OPERATIONS_*` and the tables of this and other guides).
`components`	VapourSynth, the plugins, MLRT, FFmpeg, the GPU driver; `--latest` adds the newest releases (asked of GitHub at most once a day), `--refresh` asks now.
`plan --operation install_vapoursynth`	What an installer would download: the files, their source and size. Also `install_vs_plugins`, `install_ffmpeg`, `install_mlrt`. Nothing is downloaded.
`form`	The form server of the window: a JSON request a line in, an answer a line out.
`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

```
.\vs_cli.cmd run --operation film_35mm
.\vs_cli.cmd run --operation film_35mm --set stock=500T --set intensity=1.2 --encoder prores_422hq
.\vs_cli.cmd run --operation shadow_denoise_sota --set vs_accel=cpu --source D:\Footage --target D:\Clean
.\vs_cli.cmd run --operation profile_run --set profile=filmic_warm_35mm --source D:\Footage\take.mov
.\vs_cli.cmd run --operation deband_safe --dry-run
.\vs_cli.cmd run --operation doctor
```

Key	What it does
`--operation ID`	The operation: a preset, `profile_run`, `profile_save` or a service.

Key	What it does
`--source PATH`	A file, or a folder — every video in it and its subfolders; `input\` of the program when not named.
`--target PATH`	The folder results go under; `output\` when not named.
`--set NAME=VALUE`	A field of the operation; as many as needed. A field not named keeps its default.
`--encoder NAME`	An encoder of `list encoders` put on the cards of the encoder; a `--set` of a card is laid over it.
`--params FILE`	The fields as a JSON object `{"name": value}`; `--set` is laid over it.
`--dry-run`	The commands of vspipe and FFmpeg for every file of the Source, nothing is run.
`--result FILE`	The outcome as JSON, written whatever it is.
`--cancel-file FILE`	A file whose appearance stops the run.
`--only mov,mxf`	Only these kinds of file of the Source.
`--live`, `--colours`	For a caller that reads by a pipe: the frame bar as service lines, the colour codes kept.

A relative `--source` and `--target` count from the root of the program; `--result`, `--params`, `--cancel-file` — from the folder of the caller.

The fields every preset has — the processing and the cards of the encoder, and the run:

These fields stand on every page: the processing and the cards of the encoder are one for the whole window, set on one page they are set on all.

Field	Kind	Default	Choices	What it is
`vs_accel` Processing	choice	`auto`	`auto` Auto, `cpu` CPU, `gpu` GPU	Where BM3D runs. Auto: on the video card when VapourSynth sees a working Vulkan device, else on the CPU. The other filters always run on the CPU.

Field	Kind	Default	Choices	What it is
`encode_family` Codec	choice	`x264`	`x264` x.264, `x265` x.265, `prores` ProRes, `dnxhr` DNxHR	What the result is written with.
`encode_backend` Encoder	choice	`cpu`	`cpu` CPU, `nvenc` NVIDIA, `qsv` QuickSync, `amf` AMF	The CPU or the hardware encoder of the video card. A hardware one this machine has not passed (SERVICE - Hardware capabilities) is off. (shown when encode_family is one of x264, x265)
`encode_quality` Quality	choice	`14`	`14` 14, `17` 17, `21` 21	14 - near lossless, for the archive; 17 - almost invisible losses; 21 - web, proxy, preview. (shown when encode_family is one of x264, x265)
`encode_container` Container	choice	`mp4`	`mp4` MP4, `mkv` MKV	The file x.264 and x.265 are written into. (shown when encode_family is one of x264, x265)
`prores_profile` ProRes profile	choice	`lt`	`lt` LT, `422` 422, `hq` HQ	LT is enough for grading; HQ for keying and repeated passes. (shown when encode_family is prores)
`prores_wrapper` Wrapper	choice	`mov`	`mov` MOV, `mxf` MXF	MOV for Resolve and Final Cut; MXF for Premiere and Avid. (shown when encode_family is prores)
`dnxhr_profile` DNxHR profile	choice	`hq`	`hq` HQ, `hqx` HQX, `lb` LB, `sq` SQ	LB - low bandwidth, SQ - standard, HQ - high quality, HQX - 10 bit. (shown when encode_family is dnxhr)
`overwrite` Overwrite	tick	`false`		Make the result again where it is already there. Off: a file that is already done is passed over.

Where the result goes. `<Target>\<Palette>\<preset>_<encoder>\<the folders of the Source>\<name>.<ext>`; a profile - `<Target>\Profiles\<profile>_<encoder>\...`. The name of the source is kept. A result that is there is passed over unless *Overwrite*.

A hardware encoder this machine has not passed SERVICE → Hardware capabilities is turned into the processor, and a `[NOTE]` line says so. `vs_accel` changes something only on the BM3D presets and the neural ones; on the rest it is accepted and does nothing.

A value is checked before the run. A number outside its range, a choice that is none of its field's, a field the operation does not have, `--dry-run` of an operation that cannot say its commands — each is a wrong call: exit 2 and the reason, nothing is run. A value is never moved quietly into its range.

A JSON value from PowerShell. A field whose value is JSON (`profile_values` of `profile_save`) goes through `--params FILE`: PowerShell breaks the quotes of JSON on its way into a `.cmd`.

A frame before and after

```
.\vs_cli.cmd frame --operation polaroid --source D:\Footage\take.mov
.\vs_cli.cmd frame --operation mild_denoise --set strength=strong --source
D:\Footage\take.mov --at 0.25 --out D:\Look --max-width 0
```

Key	What it does
`--operation ID`	A preset, or `profile_run` with `--set profile=NAME`.
`--source PATH`	A video, or a folder — its first video.
`--at SHARE`	Where in the clip: a share of its length, 0 to 1; 0.5 (the middle) when not named.
`--side both|before|after`	Which pictures: both when not named.
`--out FOLDER`	Where `before.png` and `after.png` go; `._runtime\frame\` when not named.
`--max-width N`	A wider frame is scaled down to N; 1920 when not named, 0 — as it is (to judge a denoiser at 100 %).
`--set`, `--params`	The fields, as in `run`.

It runs the same `.vpy` with the same values a run takes and writes nothing into the Target. The answer is one line of JSON: `before` and `after` with `path`, `frame`, `frames`, `width`, `height`, `fps`; `seconds`, `params`, and `noise` for a denoiser. A preset that changes the number of frames (RIFE) is read at the same share of its length. When it cannot — no engine, a neural preset without MLRT, no video — `ok` is `false`, `error` says why, exit 1; a wrong call is exit 2.

The output

The journal is ordinary lines:

```
[PREFLIGHT] OK engine: installed
[PREFLIGHT] OK Source: 3 video(s) in D:\Footage
[RUN] preset: film_looks/film_35mm, encoder h264_crf14, BM3D auto
[FILE] 1/3 take_01.mov
[PROGRESS] take_01.mov [##########--------------]  42% 105/250 18.40 fps ETA 00:08
Frame: 105/250 (18.40 fps)
[OK] take_01.mp4: 48,210,334 B in 13.6 s
[DONE] 3 done, 0 passed over, 0 failed
```

`[NOISE] mild_denoise: noise 5.34 / 3.44 / 2.75 (8-bit, Y/U/V) -> DFTTest sigma 57.1` — what a denoiser measured and what it made of it. `[SKIP]` — a result already there; `[FAIL]` — a file that failed, the next goes on; `[DRY RUN]`, `[STOPPED]`, `[REFUSED]`, `[NOTE]` say what they say. A caller that reads by a pipe gets service lines `::vs::{...}` (`started`, `progress`, `activity`, `finished`) too; a console does not. The journal of every run is `logs\<time>_<operation>.log`, its report `report\<time>_<operation>\run_report.json` and `.md`, with a JSON of every file beside them.

Outcome and exit codes

Code	What it means
`0`	Done.
`1`	A file failed (the others went on).
`2`	A wrong call, or a run refused before it began: no engine, no video in the Source, a neural preset without MLRT, MLRT asked without an RTX card, a Source of camera RAW alone (the engine does not read it). The `[PREFLIGHT] FAIL` lines say why; nothing is written.
`130`	Stopped.

`--result FILE` writes, whatever the outcome:

```
{"operation": "film_35mm", "outcome": "done", "ok": true, "message": "Operation
finished.",
 "processed": 3, "skipped": 0, "failed": 0, "parameters": {"stock": "250D", "...":
"..."}, "seconds": 41.2,
 "log_file": "...", "source": "D:\\Footage", "target": "E:\\...\\output", "data":
{"files": ["..."], "encoder": "h264_crf14"}}
```


`outcome` is `done`, `failed`, `refused` or `stopped`. The file is a `.json` outside the Source.

Where the result goes

`<Target>\<Palette>\<preset>_<encoder>\<the folders of the Source>\<name>.<ext>` — `Precision\`, `Film Looks\`, `Retro\`, `Restoration\`; a profile — `<Target>\Profiles\<profile>_<encoder>\…`. The name of the source is kept. A result is written as `<name>.writing-<pid>-<token>.<ext>` and takes its name only when whole. A result already there is passed over unless `--set overwrite=true`, so a second run into the same Target reads `done` with `processed` 0.

The encoders and the cards each one is:

Encoder	Cards
`dnxhr_hq`	`encode_family=dnxhr`, `dnxhr_profile=hq`
`dnxhr_hqx`	`encode_family=dnxhr`, `dnxhr_profile=hqx`
`dnxhr_lb`	`encode_family=dnxhr`, `dnxhr_profile=lb`
`dnxhr_sq`	`encode_family=dnxhr`, `dnxhr_profile=sq`
`h264_amf_q14`	`encode_family=x264`, `encode_backend=amf`, `encode_quality=14`, `encode_container=mp4`
`h264_amf_q17`	`encode_family=x264`, `encode_backend=amf`, `encode_quality=17`, `encode_container=mp4`
`h264_amf_q21`	`encode_family=x264`, `encode_backend=amf`, `encode_quality=21`, `encode_container=mp4`
`h264_crf14`	`encode_family=x264`, `encode_backend=cpu`, `encode_quality=14`, `encode_container=mp4`

Encoder	Cards
h264_crf17	encode_family=x264, encode_backend=cpu, encode_quality=17, encode_container=mp4
h264_crf21	encode_family=x264, encode_backend=cpu, encode_quality=21, encode_container=mp4
h264_nvenc_q14	encode_family=x264, encode_backend=nvenc, encode_quality=14, encode_container=mp4
h264_nvenc_q17	encode_family=x264, encode_backend=nvenc, encode_quality=17, encode_container=mp4
h264_nvenc_q21	encode_family=x264, encode_backend=nvenc, encode_quality=21, encode_container=mp4
h264_qsv_q14	encode_family=x264, encode_backend=qsv, encode_quality=14, encode_container=mp4
h264_qsv_q17	encode_family=x264, encode_backend=qsv, encode_quality=17, encode_container=mp4
h264_qsv_q21	encode_family=x264, encode_backend=qsv, encode_quality=21, encode_container=mp4
h265_amf_q14	encode_family=x265, encode_backend=amf, encode_quality=14, encode_container=mp4
h265_amf_q17	encode_family=x265, encode_backend=amf, encode_quality=17, encode_container=mp4
h265_amf_q21	encode_family=x265, encode_backend=amf, encode_quality=21, encode_container=mp4
h265_crf14	encode_family=x265, encode_backend=cpu, encode_quality=14, encode_container=mp4
h265_crf17	encode_family=x265, encode_backend=cpu, encode_quality=17, encode_container=mp4
h265_crf21	encode_family=x265, encode_backend=cpu, encode_quality=21, encode_container=mp4

Encoder	Cards
`h265_nvenc_q14`	`encode_family=x265`, `encode_backend=nvenc`, `encode_quality=14`, `encode_container=mp4`
`h265_nvenc_q17`	`encode_family=x265`, `encode_backend=nvenc`, `encode_quality=17`, `encode_container=mp4`
`h265_nvenc_q21`	`encode_family=x265`, `encode_backend=nvenc`, `encode_quality=21`, `encode_container=mp4`
`h265_qsv_q14`	`encode_family=x265`, `encode_backend=qsv`, `encode_quality=14`, `encode_container=mp4`
`h265_qsv_q17`	`encode_family=x265`, `encode_backend=qsv`, `encode_quality=17`, `encode_container=mp4`
`h265_qsv_q21`	`encode_family=x265`, `encode_backend=qsv`, `encode_quality=21`, `encode_container=mp4`
`prores_422`	`encode_family=prores`, `prores_profile=422`, `prores_wrapper=mov`
`prores_422_mxf`	`encode_family=prores`, `prores_profile=422`, `prores_wrapper=mxf`
`prores_422hq`	`encode_family=prores`, `prores_profile=hq`, `prores_wrapper=mov`
`prores_422hq_mxf`	`encode_family=prores`, `prores_profile=hq`, `prores_wrapper=mxf`
`prores_lt`	`encode_family=prores`, `prores_profile=lt`, `prores_wrapper=mov`
`prores_lt_mxf`	`encode_family=prores`, `prores_profile=lt`, `prores_wrapper=mxf`

Stopping

`--cancel-file FILE`: when the file appears, the engine ends vspipe and FFmpeg, removes the half-written file and keeps the finished ones; the outcome is `stopped`, exit 130. The window's STOP is this file.