ARRI Reference Tools Command-Line

USER MANUAL

Version history

Date	Author	Changes
2026-01-12	Matthias Hoffmann	Initial user manual for version 1.0.0
2026-03-06	Matthias Hoffmann	Added new chapter about the ‘–output’ argument, incl. possible templates
2026-06-15	Andreas Schaaf	Added new chapter about MXF/HDE as output file format
2026-10-02	Michael Mayer	Added “Measuring the rendering speed” to “The –output argument” chapter
2026-10-21	Matthias Hoffmann	Added the chapters about ASC MHL manifests and Histories; reviewed the whole manual

Contents

1 Introduction **3**
- 1.1 art-cmd tool . . . 3
- 1.2 look-builder tool . . . 3

2 General arguments of art-cmd **4**
- 2.1 The positional mode and the `--input` argument . . . 4
- 2.2 The optional `--start` and `--duration` arguments . . . 4
- 2.3 The special `--cli` argument . . . 5
- 2.4 Several `--input` arguments or directory of clips as `--input` argument . . . 5
- 2.5 Single frame file ARI and ARX clips as `--input` . . . 6
- 2.6 The `--metadata-overrides` argument . . . 7
- 2.7 The `--output` argument . . . 7
  - 2.7.1 Measuring the rendering speed . . . 9
- 2.8 The `--mhl` argument . . . 9

3 Modes of operation of art-cmd **11**
- 3.1 Mode `process` . . . 11
  - 3.1.1 ARRI Decoder Parameters . . . 11
  - 3.1.2 Batch processing . . . 11
  - 3.1.3 Target colorspace selection and looks . . . 12
  - 3.1.4 Output scaling . . . 13
- 3.2 Mode `copy` . . . 14
  - 3.2.1 Trimming clips . . . 14
  - 3.2.2 Recovering MXF files . . . 14
- 3.3 Mode `verify` . . . 15
  - 3.3.1 verification of single clips . . . 15
  - 3.3.2 pairwise verification of clips . . . 15
  - 3.3.3 JSON report file and batch verification . . . 15
  - 3.3.4 verification of ASC MHL manifests . . . 16
- 3.4 Mode `mhl` . . . 17
  - 3.4.1 Adding a generation to an ASC MHL History . . . 17
  - 3.4.2 Creating a standalone manifest . . . 17
  - 3.4.3 Emitting a creator-info template . . . 18
- 3.5 Mode `mhl-diff` . . . 19
- 3.6 Mode `mhl-flatten` . . . 20
- 3.7 Mode `export` . . . 21
  - 3.7.1 Explicit audio or metadata export . . . 21
  - 3.7.2 Metadata as CSV instead of JSON . . . 21
  - 3.7.3 Batch export . . . 21

4 Possible Input Formats of art-cmd **23**
- 4.1 MXF clip container formats . . . 23
- 4.2 MOV/ProRes clip container formats (legacy) . . . 23
- 4.3 ARI and ARX ArriFileV3 frame-files format (legacy) . . . 23

5 Possible Output Formats of art-cmd **23**
- 5.1 MXF clip container formats . . . 23
- 5.2 EXR files format . . . 23
- 5.3 MXF/ProRes file format . . . 24
- 5.4 MXF/HDE file format . . . 24
- 5.5 TIFF files format . . . 25

6 Support Contact **26**

7 Legal Notices **26**

1 Introduction

The *ARRI Reference Tools* are command-line programs that support workflows for reading, modifying, processing and checking all file types recorded or used by current and legacy generations of ARRI ALEXA and AMIRA cameras.

1.1 art-cmd tool

The art-cmd tool is able to

- *process* ARRICORE, ARRIRAW, ARRIRAW/HDE and ProRes sources into various formats used in post-production (like OpenEXR for linear VFX-workflows and TIFF or MXF/ProRes for storing image data in logarithmic colorspaces), and ARRIRAW into MXF/HDE.
- *copy* video-, audio- and metadata from various sources into an MXF container conforming to RDDs registered at SMPTE. Next to the MXF main standard ST 377-1 this is namely RDD 61 plus RDD 55 for MXF/ARRICORE, RDD 54 plus RDD 55 for MXF/ARRIRAW, RDD 44 (Apple) plus RDD 55 (ARRI) for MXF/ProRes.
- *verify* video frame data in clips by their checksums from metadata, as well as compare video-, audio- and metadata from two clips pairwise.
- *export* static and dynamic metadata, audio-tracks and look-files from clips.
- operate on a batch of multiple clips from nested directory structures containing various formats, according to generic recipes.
- *record* the files written by `process` (e.g. ARRIRAW encoded to MXF/HDE), `copy` and `export` in ASC Media Hash Lists (ASC MHL v1.1 and v2.0), as well as create ASC MHL manifests and Histories of existing files, verify and compare them, and flatten a History into one manifest.

1.2 look-builder tool

The look-builder tool is able to create look-file binaries - consumable by ARRI cameras - from 3D-LUTs in CUBE format. For details on how to use this tool, please refer to the `--help` page printed by the application.

2 General arguments of art-cmd

The general concept of parameterizing a command-line call is based on several modes acting on one or more input clips. Whenever a directory is stated as the input argument of `process`, `copy`, `export` or `verify`, all readable clips contained in the directory tree below are processed as a batch according to the additional arguments stated (in the `mhl` mode a directory is a tree of files to hash). Without an explicit `--output`, `process`, `copy` and `export` write into a directory named like the mode in the current working directory; `verify` writes no report, `mhl-diff` prints to the standard output, `mhl --mhl-history` writes into `<input>/ascmhl/`, and every other `mhl` call and `mhl-flatten` require `--output`.

Every argument may be given once, except `--input` and `--mhl-ignore`, which may be repeated. An argument given twice, e.g. `--output a/ --output b/`, or the mode given both as positional argument and as `--mode`, fails the call before any input is read or output written: one call runs one job, so a clip copied to several destinations takes one job per destination in a `--cli` file (see "The special `--cli` argument"), and `--mhl` takes several tokens joined by + in one value (see "The `--mhl` argument").

A path may hold any Unicode character, on Windows too (Windows 10 version 1903 or later; on an older Windows the name of the log file, `--logpath`, is limited to the system's ANSI code page).

2.1 The positional mode and the `--input` argument

The mode acting on given clip input(s) is defined by a positional argument (usually put as first argument in the call).

E.g.

```
art-cmd process --input C_0001C001_250807_130114_c1I82.mxf
```

processes video data of the MXF/ARRICORE clip `C_0001C001_250807_130114_c1I82.mxf` to OpenEXR single frame files `0000000.exr, 0000001.exr, ...` into the folder

`process/C_0001C001_250807_130114_c1I82/`

of the current working directory by default (as without `--output`, single frame output gets a folder per clip, named like the clip file).

By default process renders to linear ACESap0 (i.e. `--target-colorspace AP0/D60/linear`), 16bit uncompressed half-float data (i.e. `--video-codec exr_uncompressed/f16`).

The possible modes `process`, `copy`, `verify`, `export`, `mhl`, `mhl-diff` and `mhl-flatten` are described in more detail in the chapter "Modes of operation of art-cmd" following the current one.

2.2 The optional `--start` and `--duration` arguments

Next to the required `--input` argument, there are the optional `--start` and `--duration` arguments, defining the range of frames that is read from the clip(s) given as input(s). Counting the frames starts at 0, which is the first frame of the clip (so `--start 0` is equivalent to not setting it at all).

E.g.

```
art-cmd copy --input C_0001C001_250807_130114_c1I82.mxf \
             --start 1800 --duration 9000
```

cuts the input clip (by default to `copy/C_0001C001_250807_130114_c1I82.mxf` if no explicit `--output` is given), its copy containing the 9000 frames 1800 to 10799 of the original clip.

The `--start` and `--duration` arguments restrict the range of frames that `process`, `copy` and `export` read from the input clip(s). The other modes refuse them, as `verify` checks whole clips. `--duration` takes 1 or more frames, while `-1` (the default) reads all frames from `--start` on.

2.3 The special `--cli` argument

The special `--cli` argument, that can be stated instead of a mode and some input(s), takes a JSON file that conforms to the `cli.schema.json` schema (in `doc/` folder of the package), defining a sequence of one or more individual command-line calls.

E.g.

```
art-cmd --cli renderqueue.json
```

instead of calling the two examples from above, with a `renderqueue.json` containing

```
{
    "jobs": [
        {
            "mode": "process",
            "input": ["C_0001C001_250807_130114_c1I82.mxf"],
            "recipe":
            {
                "target-colorspace": "AP0/D60/linear",
                "video-codec": "exr_uncompressed/f16"
            },
            "output": "process/C_0001C001_250807_130114_c1I82/%07d.exr"
        },
        {
            "mode": "copy",
            "input": ["C_0001C001_250807_130114_c1I82.mxf"],
            "recipe":
            {
                "start": 1800,
                "duration": 9000
            },
            "output": "copy/C_0001C001_250807_130114_c1I82.mxf"
        }
    ]
}
```

renders the same results as the two individual calls.

A `--cli` call refuses job arguments on its command line (`--mode`, `--input`, `--output` and the recipe's); the others, e.g. `--render-platform`, `--mhl-ignore`, `--mhl-threads` and `--logpath`, apply to the whole call, so to every job.

A file that is no valid JSON, holds no object with a `"jobs"` array of objects, or no job, is refused. Before the first job runs, each job's arguments are checked against its mode and against one another (e.g. `--mhl` in a `verify` job, `--cct` outside `process`, or `mhl-flatten` without `--output`), so that such a misuse fails the call before anything is written; what needs a file, e.g. an input or a `--mhl-creator-info` template that an earlier job writes, is checked when its job starts. A job that fails while running does not stop the others: its error is logged with its number, and after the last job the call exits with error code 1, naming the failed jobs, e.g. `Exiting with error code 1, because 1 of the 3 jobs of the '--cli' file failed: 2`.

A job may omit `"output"` wherever the command line may omit `--output`: it then writes to the mode's default output (e.g. `./copy/`), or as its mode describes it without `--output`, e.g. `mhl --mhl-history` into the History of its `--input`, `mhl-diff` to the standard output and `verify` without a report. Likewise a job omits `"input"` for `mhl --mhl-emit-template`, which takes no `--input`; every other job needs one, or the file is refused before its first job runs.

2.4 Several `--input` arguments or directory of clips as `--input` argument

It is also possible to state several `--input` arguments in case the same operation should be applied on more than one clip. The output filetree is in this case generated by replicating the filetree of the inputs, starting from their common root in the filesystem.

E.g.

```
art-cmd process --input clips/arricore.mxf \
                --input clips/arriraw.mxf \
                --input clips/subdir/prores.mov \
                --video-codec prores422 --output out/pr422.mxf
```

will render

```
./out/
  ├── arricore/
  │   └── pr422.mxf
  ├── arriraw/
  │   └── pr422.mxf
  └── subdir/
      └── prores/
          └── pr422.mxf
```

In case the `clips/` directory only contains these three explicitly stated clips, following call (with just the common root directory as input argument)

```
art-cmd process --input clips/ --video-codec prores422 --output out/pr422.mxf
```

is *equivalent* to the call before, triggering a batch processing of all clips contained in the input directory.

The inputs are taken as given, symbolic links included: the output names, `[inputfile]` and the replicated file tree follow the names of the links, not of their targets. A clip given twice, also once through a link and once by its target, is refused, and so is an input that does not exist, e.g. a stray word of the call, or a printf pattern whose directory does not exist.

Not stating an explicit filename for the output clips is also possible, reducing the depth of the rendered filetree.

E.g.

```
art-cmd process --input clips/ --video-codec prores422
```

(implying `--output process/` by default) reuses the filenames of the original clips (a filename that equals the clip's `clipName` gets its codec indicator changed to that of the output, e.g. `A_0001C003_220824_061717_a12SQ.mxf` becomes `A_0001C003_220824_061717_p12SQ.mxf` with ProRes output) and the following filetree

```
./process/
  ├── arricore.mxf
  ├── arriraw.mxf
  └── subdir/
      └── prores.mxf
```

is rendered.

2.5 Single frame file ARI and ARX clips as `--input`

Legacy ARRIRAW or ARRIRAW/HDE clips, stored as *.ari or* .arx frames in directories, are supported by stating the path incl. numbering as printf-pattern.

E.g.

```
art-cmd copy --input ari_clip/D401C005_180614_R1ZA.%07d.ari
```

copies each frame of the ARI clip into `./copy/D401C005_180614_R1ZA.mxf`

Instead of the path-pattern it's also possible to just state the directory of the single frame clip as input.

E.g.

```
art-cmd copy --input ari_clip/ --start 2 --duration 5 --output trim/
```

called with the input directory

```
ari_clip/
├── D401C005_180614_R1ZA.0019130.ari (--start 0)
├── D401C005_180614_R1ZA.0019131.ari
├── D401C005_180614_R1ZA.0019132.ari <- first frame copied (--start 2)
├── D401C005_180614_R1ZA.0019133.ari
├── D401C005_180614_R1ZA.0019134.ari
├── D401C005_180614_R1ZA.0019135.ari
├── D401C005_180614_R1ZA.0019136.ari <- last frame copied (--duration 5)
├── D401C005_180614_R1ZA.0019137.ari
└── ...
```

copies frame three to seven to a new MXF/ARRIRAW clip `./trim/D401C005_180614_R1ZA.mxf`, trimmed to the five frames defined.

2.6 The `--metadata-overrides` argument

There is a set of metadata defined in the `overrides.schema.json` JSON schema that can be changed. A JSON file conforming to this schema can be provided to the `--metadata-overrides` argument.

E.g.

```
{
  "lensSqueezeFactor": "4/3",
  "director": "James Cameron",
  "productionCompany": "Lightstorm Entertainment"
}
```

used as `overrides.json` in

```
art-cmd copy --input recorded.mxf --metadata-overrides overrides.json \
             --output recorded_fixed_metadata.mxf
```

will change the director's and production company's name in `recorded_fixed_metadata.mxf` as well as the lens squeeze factor (and the aspect ratio as it is related to the lens squeeze factor). A file with an unknown key or a malformed value (e.g. `"lensSqueezeFactor": "1.5"`) is refused before anything is written.

2.7 The `--output` argument

The `--output` argument sets the filepath (incl. extension) or path to a directory to the desired output destination in the filesystem. When batch processing a whole directory, the inputs' file trees are replicated in the output path.

An existing directory is always taken as a directory, also one named like a file (e.g. `x.mxf/`, into which `process` without `--video-codec` renders OpenEXR frames), and so is a name without a known extension. `copy` write an `.mxf` file or into a directory, `export` into a directory or to a `.wav`, `.json` or `.csv` file, and `process` into a directory or to an `.mxf`, `.exr`, `.tif` or `.tiff` output that fits the `--video-codec` (`.mxf` for ProRes, HDE, `.exr` for the `exr_*` codecs). Any other output is refused before anything is written.

Decimal integer printf-format specifiers, like `lin_%07d.exr` or `log_%07d.tif`, are supported for single frame output filepaths.

It is also possible to state certain templates, enclosed by [] that will be replaced by metadata from the clip that is processed. Those are:

Template	*Replacement*
[inputfile]	the filename of the input clip as in the filesystem without extension (and without frame counter for ari/arx clips; frames named by their number alone give the name of their folder)
[clipdir]	the name (not the path) of the directory the input clip is located in
[clipdir+1]	the name of the parent directory of [clipdir]
...	...

Template	*Replacement*
[clipdir+4]	the name of the parent directory of [clipdir+3]
[clip]	the `clipName` (if present, “clip-NotPresent” else)
[reel]	the `reelName` (if present, “reel-NotPresent” else)
[scene]	the `scene` (if present, “scene-NotPresent” else)
[take]	the `take` (if present, “take-NotPresent” else)
[resolution]	the display resolution of the input clip as “WxH”
[camera]	the camera model, followed by `-<variant>` if present, e.g. “ARRI_ALEXA_35” or “ARRI_ALEXA_LF-3” (“camera-NotPresent” else)
[serialnr]	the serial number of the camera device (if present, “serialnr-NotPresent” else)
[anamorph]	the anamorphic factor of the lens with two decimals, e.g. “1.50” (“1.00” without a squeeze factor)
[projectfps]	the project rate, e.g. “24fps” or “23.976fps”
[sensorfps]	the sample rate of the sensor, e.g. “330fps” or “23.976fps”
[inputcodec]	the codec of the video data encoded in the clip source
[texture]	the texture name burned into the video data, without “.at1” (if present, “texture-NotPresent” else)

Spaces in [scene], [take], [camera], [inputcodec] and [texture] are replaced by `_`. The templates take the metadata of the input clip, with the values of `--metadata-overrides`. A rate or squeeze factor that is no rational, like the sensor rate `0/0` of ALEXA 35 SUP 1.1.0 clips, gives “sensorfps-NotPresent”, “projectfps-NotPresent” or “anamorph-NotPresent”. Bracketed text that is no template or frame counter, e.g. a misspelt `[CLIP]`, is written as it stands, with a warning.

For single frame output format (.exr, .tif) also:

Template	*Replacement*
[#####]	e.g. frame counter with 5 digits, beginning from 0, plus `--start` argument
[####+2]	e.g. frame counter with 4 digits, beginning from 2, plus `--start` argument
[#######,framenr]	e.g. frame counter with 7 digits, beginning from the clip’s start frame number, plus `--start` argument
[######-3,framenr]	e.g. frame counter with 6 digits, beginning from the clip’s start frame number minus 3, plus `--start` argument

A single-frame `--output` takes one frame counter (or printf specifier), and a WAV `--output` one printf specifier (a [###] counter stays as written there), in the file name: a second one, or one with a width ([#####], `%05d`) written in a folder that does not exist yet, is refused before anything is written. An `.exr` or `.tif` output without one is refused for a clip that renders more than one frame. Any other `%` in a folder name, e.g. in `out/50%off/`, in `out/f%d/` or from a template value, is taken literally, and one in the file name is written `%%`.

E.g.

```
art-cmd process --input arricore.mxf \
        --output '[resolution]_ana[anamorph]-[sensorfps]/frame[#####,framenr].exr'
```

will render

```
./
└── 4608x3164_ana1.50-50fps/
    ├── frame01500.exr
    ├── ...
    └── frame100500.exr
```

for an MXF/ARRICORE clip with 4608x3164 resolution that was recorded with a lens of squeeze factor 1.5 at 50 fps realtime (sensor and project rate), from timecode 00:00:30:00 to 00:33:30:00. Quote an `--output` with templates, as some shells (e.g. zsh) take the brackets for a file pattern.

2.7.1 Measuring the rendering speed

In `process` mode, `--output perf-measure` renders like a TIFF output, but writes nothing and logs the achieved frame rate per clip instead. The first 3 frames are skipped as warm-up, so render more than that (see `--duration`). `--video-codec` is refused with it, as nothing is encoded.

```
art-cmd process --input arricore.mxf --render-platform cuda --output perf-measure
```

2.8 The `--mhl` argument

An ASC Media Hash List (ASC MHL) records files together with their hashes (checksums), so that their integrity can be checked whenever they are copied, delivered or archived later on. In `process`, `copy` and `export` mode, the `--mhl` argument records the files that the call writes into its output folder. It records them in an ASC MHL v2.0 History, in an ASC MHL v1.1 manifest or in both, written into the same folder. Files that were in the output folder before are not recorded (see "Adding a generation to an ASC MHL History" for recording them as well).

E.g.

```
art-cmd process --input C_0001C001_250807_130114_a1I82.mxf \
                --video-codec arriraw_hde --output out/ --mhl v2-xxh64
```

encodes the MXF/ARRIRAW clip to MXF/HDE (see "MXF/HDE file format") and records the new clip with its XXH64 hash in an ASC MHL v2.0 History, which is the folder `ascmhl/` next to it:

```
./out/
  ├── C_0001C001_250807_130114_h1I82.mxf
  └── ascmhl/
      ├── 0001_out_2026-10-15_071821.mhl
      └── ascmhl_chain.xml
```

The manifest `0001_out_2026-10-15_071821.mhl` is the first generation of the History, named after its number, the output folder and the local date and time of its creation, and the `ascmhl_chain.xml` lists the generations of the History.

The value of the `--mhl` argument is a token v<M>-<A>, selecting the ASC MHL version by <M> and the hash algorithm by <A>, or several tokens joined by +:

Token	ASC MHL version	Algorithms <A>
v1-<A>	v1.1	md5, sha1, xxhash, xxhash64, xxhash64be, null
v2-<A>	v2.0	c4, md5, sha1, xxh64, xxh3, xxh128

XXH64 is called xxhash64be in ASC MHL v1.1 (its xxhash64 is a deprecated little-endian form) and xxh64 in v2.0.

E.g.

```
art-cmd process --input C_0001C001_250807_130114_a1I82.mxf \
                --video-codec arriraw_hde --output out/ --mhl v2-xxh64+v1-md5
```

called instead of the one above writes the same History, plus the ASC MHL v1.1 manifest `out/out_2026-10-15_071821.mhl` holding the MD5 hash of the clip for tools that only read ASC MHL v1.1 (the History records this manifest, too). With ASC MHL v1.1 tokens only, e.g. `--mhl v1-md5`, the call writes this manifest alone, without a History.

Usually the clips of a whole camera card are encoded as a batch and recorded in one History. E.g. with the clips of a Codex drive offloaded to

```
./C_0001/
  ├── C_0001C001_250807_130114_a1I82.mxf
  └── C_0001C002_250807_131502_a1I82.mxf
```

the call

```
art-cmd process --input C_0001/ --video-codec arriraw_hde \
                --output hde/C_0001/ --mhl v2-xxh64
```

renders

```
./hde/
  └── C_0001/
      ├── C_0001C001_250807_130114_h1I82.mxf
      ├── C_0001C002_250807_131502_h1I82.mxf
      └── ascmhl/
          ├── 0001_C_0001_2026-10-15_071821.mhl
          └── ascmhl_chain.xml
```

with both encoded clips recorded in the first generation of the History of `hde/C_0001/`.

A later call into the same output folder adds the next generation to its History, checking each output that the History already lists. Only clips encoded completely are recorded, so a clip that failed, e.g. on a full disk, is not listed in the History and can be encoded once more into the same folder by a call for this clip alone.

A clip that the History lists already, however, does not match its earlier version when encoded once more, as an MXF file records the time it was written. It replaces the earlier clip and is recorded as `failed`, and the call exits with error code 1 (`MHL: 'C_0001C001_250807_130114_h1I82.mxf' does not match the History (xxh64)`). Every later verification of the folder then fails until the earlier clip is restored from a copy, so a whole card is encoded once more into a new, empty folder only.

For the same reason, a second copy of the encoded clips is not encoded again, but made by copying the card folder byte by byte into an existing folder, e.g. `backup/` on a backup drive, with

```
cp -R hde/C_0001 backup/
```

so that the copy `backup/C_0001/` holds the same History, against which `verify` checks it as described in “verification of ASC MHL manifests”.

In the same way `copy` and `export` record the files they write. E.g.

```
art-cmd copy --input clips/ --mhl v2-xxh64
```

copies the clips of the `clips/` directory from “Several `--input` arguments or directory of clips as `--input` argument” to `./copy/` and records them in the History `./copy/ascmhl/`. An export records every file it writes, also the audio tracks that an export to an explicit `.wav` file (see “Explicit audio or metadata export”) writes to files of their own (`audio00.wav`, `audio01.wav`, `...`).

Each manifest records the host name, the tool and the dates of the call. An author, a location or a comment is added to an ASC MHL v2.0 manifest by `--mhl-creator-info`, as described in “Emitting a creator-info template”.

3 Modes of operation of art-cmd

3.1 Mode process

In process mode, image data read from the given input clip(s) are rendered to a new clip containing RGB image data (OpenEXR, TIFF or MXF/ProRes), or ARRIRAW input is encoded to MXF/HDE (`--video-codec arriraw_hde`). `--skip-audio` leaves out the audio of such a re-encode; it is refused for a rendered output.

3.1.1 ARRI Decoder Parameters

Exclusively in process mode the `--parameters` argument, defining parameters for decoding ARRIRAW and ARRICORE to uncompressed RGB pixels, is applicable.

E.g.

```
{
  "exposureIndexOverride": 700,
  "colorTemperatureOverride": 5300,
  "tintOverride": 3.0
}
```

as parameters JSON file `asa_cct_tint.json` used in a call like

```
art-cmd process --input arricore.mxf \
                --parameters asa_cct_tint.json \
                --video-codec prores4444xq \
                --output asa700_cct5300_tint3_LogC_prores4444xq.mxf
```

renders an MXF/ProRes with 4444XQ encoded `AWG4/D65/LogC4` (or `AWG3/D65/LogC3`, dependent on input clip) image data to

```
./
└── asa700_cct5300_tint3_LogC_prores4444xq.mxf
```

Note: The JSON-schema for all ARRICORE- and ARRIRAW-related parameters is defined in `parameters.schema.json`. For a more detailed explanation of the parameters and their possible values, there are also the specific XSD-schemas `ArriCoreDecoderParameters.xsd` and `ArriRawDecoderParameters.xsd` used by the ARRI Image SDK integrated in art-cmd.

The white balance and the exposure index can also be set without a parameters file, by `--cct` (the color temperature in Kelvin), `--tint` and `--asa` (the exposure index, at most 3200 for footage recorded by ALEXA Mini LF or older cameras), which take precedence over the values of a parameters file. Like `--parameters`, they are valid in process mode only, not for MXF/HDE output, and refused for ProRes clips.

E.g.

```
art-cmd process --input arricore.mxf --cct 5300 --tint 3 --asa 700 \
                --video-codec prores4444xq \
                --output asa700_cct5300_tint3_LogC_prores4444xq.mxf
```

renders the same MXF/ProRes as the call above with `asa_cct_tint.json`.

3.1.2 Batch processing

It is also possible to render several input clips using the same parameters JSON and additional arguments like `--target-colorspace`, converting to a defined colorspace.

E.g.

```
art-cmd process --input arricore.mxf \
                --input arriraw.mxf \
                --parameters asa_cct_tint.json \
                --target-colorspace Rec.2020/D65/PQ \
                --video-codec prores4444xq \
                --output asa700_cct5300_tint3_PQ_prores4444xq.mxf
```

renders both input clips as MXF/ProRes with 4444XQ encoded Rec.2020/D65/PQ image data to

```
./
├── arricore/
│   └── asa700_cct5300_tint3_PQ_prores4444xq.mxf
└── arriraw/
    └── asa700_cct5300_tint3_PQ_prores4444xq.mxf
```

A file that is not of a supported clip type is ignored, and a clip whose output cannot be set up (e.g. an output folder that cannot be created, or a clip that ends before `--start`) is skipped with a warning; a run left with no clip fails. Existing output files are overwritten. A clip that cannot be read (e.g. a damaged .mxf) or that fails while being processed is logged as an error naming it (`process of '<clip>' failed with: ...`), and the run exits with error code 1 after the remaining clips, naming each failed clip. A refused job or a failed clip leaves no new empty output folder, and no file when it fails before its first frame.

3.1.3 Target colorspace selection and looks

The `--ls-target-colorspaces` argument (in `process` mode only) can be used to query the possible values for the `--target-colorspace` argument in the current configuration.

E.g.

```
art-cmd process --input arricore.mxf \
                --video-codec prores4444xq \
                --ls-target-colorspaces
```

will list the following possible target colorspaces

```
AWG4/D65/LogC4
P3/D60/DCI
P3/DCI/DCI
P3/D65/DCI
P3/D65/PQ
Rec.2020/D65/BT.1886
Rec.2020/D65/HLG
Rec.2020/D65/PQ
Rec.709/D65/BT.1886
```

that can be used to convert an MXF/ARRICORE clip to MXF/ProRes.

Without a look, OpenEXR output takes the linear colorspaces only, MXF/ProRes and TIFF output the others. A LogC3 ARRIRAW clip also renders to `AWG4/D65/LogC4` and `AWG4/D65/linear`, a LogC3 ProRes clip to no AWG4 colorspace. A `--target-colorspace` that is not listed is refused before anything is rendered.

In case a look shall be applied, i.e. when `--embedded-look` or `--arri-look-file` is also set, the amount of possible output colorspaces reduces to the targets of the (clip-embedded or external) look file: its SDR and HDR DRT targets, otherwise its own target colorspace. Any other `--target-colorspace`, a LogC one included, is refused, and so are a clip without an embedded look for `--embedded-look`, a look of the other LogC generation (`.aml` for LogC3, `.alf4`/`.alf4c` for LogC4 clips) and, with a look, OpenEXR output, also when listing. A LogC3 ARRIRAW or HDE clip whose `--parameters` file sets `transferCurveOverride` to LogC4 is rendered to LogC4 and takes an `.alf4`/`.alf4c` look (a LogC3 ProRes clip does not).

E.g.

```
art-cmd process --input arricore.mxf \
                --video-codec prores4444xq \
                --arri-look-file sdr_and_hdr_drt.alf4c \
                --ls-target-colorspaces
```

could list the following possible target colorspaces

```
Rec.709/D65/BT.1886
Rec.2020/D65/PQ
```

in case the SDR DRT contained in `sdr_and_hdr_drt.alf4c` is a 3D-LUT converting to `Rec.709/D65/BT.1886` and the HDR DRT is a 3D-LUT converting to `Rec.2020/D65/PQ`.

Note: For MXF/ProRes and TIFF output without a look, and with an `--arri-look-file`, the colorspace listed first is the default (in case no `--target-colorspace` is set at all). OpenEXR output defaults to `AP0/D60/linear`, and `--embedded-look` to the target of the clip's embedded look (e.g. `Rec.709/D65/BT.1886`), whatever is listed first.

3.1.4 Output scaling

If the output should be scaled to a certain width or height (while preserving the aspect ratio), the `--output-width` or `--output-height` argument can be used. The other dimension follows the aspect ratio of the picture after `--desqueeze` and after a crop, binning or proxy mode of `--parameters`. Both arguments together set exactly that size, which may stretch the picture (with a warning); `--letterbox-size` keeps the aspect ratio instead. A desqueezed output records a lens squeeze factor of 1/1, so that it is not desqueezed once more. A size out of the range the clip renders to (the decoder's for ARRIRAW and ARRICORE, for ProRes 16 pixels up to twice the clip's size) is refused, naming the range.

E.g.

```
art-cmd process --input prores.mxf --video-codec prores4444xq --output-width 1920
```

(where the input `prores.mxf` has a resolution of 2880x1620) renders an MXF/ProRes with image data of resolution 1920x1080 as `./process/prores.mxf` output. Without `--video-codec`, the same call renders an OpenEXR sequence `./process/prores/0000000.exr`, `...` (in `AP0/D60/linear`) instead.

3.2 Mode copy

In copy mode all video- and audio-data from input clip(s) will remain untouched but metadata are converted to the MXF labels defined in RDDs 44, 54, 55 and 61 at SMPTE while creating MXF clip container file(s) from the input clip(s). This way all legacy file formats can be re-wrapped into an up-to-date MXF file and potentially broken MXF files from cameras can be recovered into a new, properly finalized MXF file.

E.g.

```
art-cmd copy --input prores.mov --output prores.mxf
```

copies the data from MOV/ProRes into an MXF/ProRes file, containing metadata as standardized in SMPTE ST 377-1 (MXF), RDD 44 (Apple) plus RDD 55 (ARRI). An existing directory as `--output`, also one named like `prores.mxf/`, gets the copy `<directory>/<clip>.mxf`.

The former mode names `trim` and `mxf-recovery` still run `copy` on the command line; a `--cli` file says `copy` (or `mxf-recovery`).

E.g.

```
art-cmd copy --input ari_clip/D401C005_180614_R1ZA.%07d.ari --output arriraw.mxf
```

copies video- and metadata from the ARI single frame files in `ari_clip/` to an MXF/ARRIRAW file, containing metadata as standardized in SMPTE ST 377-1 (MXF main std.) plus RDD 55 (ARRI main std.) and RDD 54 (ARRIRAW std.), to `arriraw.mxf`.

Note: The file pattern `D401C005_180614_R1ZA.%07d.ari` is optional in this call. Just stating `--input ari_clip/` results in the same behavior as described in chapter “Single frame file ARI and ARX clips as `--input`” above.

3.2.1 Trimming clips

The `copy` mode can also be used to cut or trim clips when used together with the `--start` and/or `--duration` arguments.

E.g.

```
art-cmd copy --input C_0001C001_250807_130114_c1I82.mxf \
             --start 1800 --duration 9000
```

cuts the input clip (by default to `copy/C_0001C001_250807_130114_c1I82.mxf` if no explicit `--output` is given), its copy containing the 9000 frames 1800 to 10799 of the original clip (counting from 0).

3.2.2 Recovering MXF files

The recovery function should be used if a recorded MXF file from the camera was not completed correctly, e.g. due to a sudden power failure. A new, properly completed MXF file is created by copying it and generating a valid file completion. E.g.

```
art-cmd copy --input unfinalized.mxf --output recovered.mxf
```

Note: The recovered file holds the frames up to the last complete one. Its size differs from that of the unfinalized file: storage space allocated on the recording medium during recording is not copied, but the file is rewrapped with its own partitions, index and fill (as with every `copy`), so it is not necessarily smaller.

3.3 Mode `verify`

The `verify` mode can be used to verify video frame data in clips by their checksums from metadata, as well as compare video-, audio- and metadata from two clips pairwise. Optionally a JSON report file can be generated when stated as `--output` argument.

3.3.1 verification of single clips

Verification of video frame data in clips by their checksums from metadata is performed when stating one `--input` argument.

E.g.

```
art-cmd verify --input arriraw.mxf
```

will log all recalculated video-checksums that do not match as human-readable messages.

Note: Only available for ARRIRAW and HDE input with checksum metadata. A clip without it (MOV, MXF/ProRes and MXF/ARRICORE, but also e.g. ALEXA XT `.ari` clips) gives `Verification was not possible because of missing checksums.`, is listed in the report's `failedClips` and fails the run with error code 1.

3.3.2 pairwise verification of clips

When providing two `--input` arguments (two clips, or two directories), clips are verified pairwise by comparing video-, audio- and metadata between the clips; more inputs, or a clip together with a directory, are refused. A directory of frame files counts as a clip.

E.g.

```
art-cmd verify --input arriraw.mxf --input arriraw_copy.mxf
```

will log all differences in video-, audio- or metadata within the two clips.

Note: In case one clip is shorter than the other but all its frames are contained in the longer clip, verification is still possible (e.g. after trimming a clip).

3.3.3 JSON report file and batch verification

It is also possible to create a JSON report file, conforming to the `verify_report.schema.json`, where verification results (and some additional info) are collected. This especially is useful when running verifications in batch mode (with one or two directories of clips as input).

E.g.

```
art-cmd verify --input clips/ --output batch_verify.json
```

will run a "verification of single clips" for each clip contained in the `clips/` directory and create a `batch_verify.json` file with results for each clip verified.

The folder of an `--output .json` file is created if it does not exist yet. A report that cannot be written (e.g. one whose folder cannot be created) fails the run with error code 1, after the verification. `--output` may also name an existing directory, which then gets a `verify_results.json`; a directory that does not exist yet is refused before anything is verified.

E.g.

```
art-cmd verify --input clips/ --input clips_copy/ --output batch_pair_verify.json
```

will run a "pairwise verification of clips" for each clip contained in the `clips/` and the `clips_copy/` directory (paired by their paths below the directories) and create a `batch_pair_verify.json` file with results for each clip pair verified. A clip in one directory only is logged as an error (`The clip '...' has no counterpart '...' in the other folder.`), listed in `failedClips` with the `readError` `clipA does not exist` or `clipB does not exist`, and fails the run with error code 1, as do two directories without any clip. Files that are no clips (e.g. `notes.txt`) are ignored.

The report's `logPath` is the log file written, or `null` without one (`--verbose quiet`, `--logpath ""` or a log file that could not be created).

3.3.4 verification of ASC MHL manifests

With an ASC MHL manifest (`.mhl`), the `ascmhl_chain.xml` of an ASC MHL History or an `ascmhl_collection.xml` (a collection of manifests and Histories, as other tools write it) as its only `--input`, `verify` hashes the files listed once more and compares the results with the recorded hashes.

E.g.

```
art-cmd verify --input backup/C_0001/ascmhl/ascmhl_chain.xml
```

checks the copy `backup/C_0001/` of the encoded card from “The `--mhl` argument”, logging each file verified, like `MHL: verified 'C_0001C001_250807_130114_h1I82.mxf' (xxh64)`. A History is checked against every file below the folder that holds it, so a file that has changed, is missing or is not recorded in the History (e.g. one added later) fails the verification. E.g. in case `C_0001C002_250807_131502_h1I82.mxf` was damaged and a `notes.txt` was added to the folder, the call logs

```
Exiting with error code 1, because:
MHL: hash mismatch on 'C_0001C002_250807_131502_h1I82.mxf' (xxh64)
MHL: unexpected file 'notes.txt' (not in manifest)
```

and exits with error code 1. A standalone manifest, like the ASC MHL v1.1 manifest written by `--mhl v2-xxh64+v1-md5`, is checked for the files it lists only. A generation in `ascmhl/`, e.g. `backup/C_0001/ascmhl/0001_C_0001_2026-10-15_071821.mhl`, is checked like the History at that generation, so a file added later, or ignored by a later generation only, is unexpected. An `--input` that does not exist is refused (`verify: '<path>' does not exist`).

As for clips, a JSON report file is written when stated as `--output` argument. It lists each file checked in `passedClips` or `failedClips`, the latter with the reason as `readError`, e.g. `MHL: hash mismatch (xxh64) expected=a6505e676767aa97 computed=a1a78415d4be2f87`.

A verification started in a folder that a History records (e.g. from within `backup/C_0001/`) writes its log file `art.log` into that folder. The run itself leaves that file out, but a verification started elsewhere reports it as an unexpected file. Therefore `verify` is best called from outside that folder (or with `--logpath ""`, writing no log file).

3.4 Mode `mhl`

The `mhl` mode hashes files that exist already, without processing or copying them: the files below a folder into a new generation of its ASC MHL History (`--mhl-history`, for ASC MHL v2.0 only), or a folder or a single file into a standalone manifest written to `--output`. The `--mhl` argument takes the tokens described in “The `--mhl` argument”, all of the same ASC MHL version.

3.4.1 Adding a generation to an ASC MHL History

E.g.

```
art-cmd mhl --input hde/C_0001/ --mhl v2-xxh64 --mhl-history
```

adds the second generation `hde/C_0001/ascmhl/0002_C_0001_2026-10-16_093012.mhl` to the History of the encoded card from “The `--mhl` argument”, e.g. after a camera report `C_0001_report.pdf` was put into the folder. The call hashes every file below the folder and records a file that is new to the History, like the report, as `original`, a file matching the History as `verified` and a file not matching it as `failed`, in which case the call exits with error code 1 (as it does when a file of the History is missing, also when every recorded file is gone). Unlike `verify`, it keeps the result in the History. A folder without a History gets its first generation this way. Symbolic links to folders are not entered and broken links are skipped, each with an info line in the log.

A folder whose `ascmhl/` holds generations but no `ascmhl_chain.xml` is refused, so restore the chain first. In a folder whose name holds `:`, e.g. `tc 10:00/`, the generation files get `_` in its place (`0001_tc 10_00_2026-10-16_093012.mhl`), and a History that another tool wrote there can be verified and extended (on Linux and macOS).

Files matching an ignore pattern, stated in gitignore syntax by `--mhl-ignore` (e.g. `--mhl-ignore '*.tmp'`), are not recorded. The generation records the patterns, so that they apply to the later generations of the History as well. A new pattern also hides the files that earlier generations recorded, from this generation and from every later verification; the call warns with their number and the first of them. The log file of the call (`--logpath`), when it lies in the folder, is left out the same way: the generation records it and its rotations (`art.1.log`, `...`) as patterns anchored to its path, which the later generations keep.

The `--mhl-threads` argument sets how many files are hashed at once (by default one per logical processor, at most 8, while `--mhl-threads 1` suits a spinning disk). `--mhl-ignore` and `--mhl-threads` apply to the whole call, so to every job of a `--cli` file; a job that writes a v2.0 manifest takes patterns of its own from its `--mhl-creator-info` template.

A file or folder whose name holds `:` or `\` cannot be recorded in an ASC MHL manifest: the call fails naming it (in `process`, `copy` and `export` after writing its outputs, without the manifest), so leave it out with `--mhl-ignore`. In a pattern `\` escapes the next character, so `back\slash.txt` is matched by `'back\\slash.txt'` or `'back?slash.txt'`, while `:` is written as it is, e.g. `--mhl-ignore 'tc 10:30.txt'`.

The files of a folder below `--input` that holds a History of its own (a nested History) are recorded in a new generation of that History, which the generation of `--input` references. E.g.

```
art-cmd mhl --input hde/ --mhl v2-xxh64 --mhl-history
```

adds a generation to the History of the card folder `hde/C_0001/` (and to those of further cards encoded into `hde/` the same way) and creates the History `hde/ascmhl/` referencing them. A verification of `hde/ascmhl/ascmhl_chain.xml` then checks the files of all cards at once.

3.4.2 Creating a standalone manifest

E.g.

```
art-cmd mhl --input clips/ --output clips.mhl --mhl v1-md5
```

writes the ASC MHL v1.1 manifest `clips.mhl` with the MD5 hashes of the clips in `clips/`, e.g. to be handed over together with the folder. The paths are recorded relative to the folder of the manifest (`clips/arricore.mxf`, `clips/arriraw.mxf`, `...`), so the manifest has to lie in the folder hashed or in a folder above it.

A v2.0 standalone manifest of a folder that holds an ASC MHL History, its own or a nested one, references the latest generation of that History instead of hashing its files; a v1.1 manifest hashes them.

3.4.3 Emitting a creator-info template

An author, a location or a comment is recorded in an ASC MHL v2.0 manifest from a creator-info template.

E.g.

```
art-cmd mhl --mhl-emit-template --mhl v2 --output creator.mhl
```

writes such a template to `creator.mhl`, where `--mhl v2` names the version only, as a template holds no hashes (`--mhl v1` writes a template for ASC MHL v1.1, which has other fields). Once its fields are filled in, e.g. as

```
<author email="dit@example.com" role="DIT">Jane Doe</author>
<location>Stage 5</location>
```

the template is passed as `--mhl-creator-info creator.mhl` to the `mhl`, `process`, `copy` or `export` call that writes the manifest, together with an `--mhl` argument of the template's version. An element the template does not define, e.g. a misspelt `<autor>`, is refused, naming the element and the template.

3.5 Mode `mhl-diff`

The `mhl-diff` mode compares an ASC MHL History or manifest with the files below the folder it records. This comparison uses the names and sizes of the files and does not hash them, so it quickly finds files that were added, removed or changed in size.

E.g.

```
art-cmd mhl-diff --input hde/C_0001/ascmhl/ascmhl_chain.xml \
                 --output diff.json
```

writes `diff.json` with the following content, in case `C_0001C002_250807_131502_h1I82.mxf` was deleted from the card folder `hde/C_0001/` of “Mode `mhl`” and a `notes.txt` was added to it:

```
{
    "inputA": "hde/C_0001/ascmhl/ascmhl_chain.xml",
    "inputB": "/path/to/hde/C_0001",
    "added": [
        "notes.txt"
    ],
    "removed": [
        "C_0001C002_250807_131502_h1I82.mxf"
    ],
    "changed": [],
    "problems": []
}
```

The call exits with error code 1, as it does whenever a file below the folder was added, removed or changed in size. A file that changed without changing its size is found by `verify` only. Unlike `verify`, which checks only the files that a standalone manifest lists, `mhl-diff` reports every other file below the folder of a standalone manifest as added. Without `--output`, the JSON is printed to the standard output among the log lines (alone with `--verbose quiet`).

With two manifests as `--input` arguments, `mhl-diff` compares the manifests with each other, also listing renamed files.

E.g.

```
art-cmd mhl-diff --input hde/C_0001/ascmhl/0001_C_0001_2026-10-15_071821.mhl \
                 --input hde/C_0001/ascmhl/0002_C_0001_2026-10-16_093012.mhl
```

lists the camera report `C_0001_report.pdf` from “Mode `mhl`” as added between the first two generations of the card’s History. Differences between two manifests are not treated as an error (exit code 0). A file that both list is compared by a hash algorithm they share; without one it counts as changed, and `mhl-diff` warns with the number of such files.

3.6 Mode `mhl-flatten`

The `mhl-flatten` mode merges all generations of an ASC MHL History, and of the Histories nested in it, into one ASC MHL v2.0 manifest, e.g. as a packing list for a delivery.

E.g.

```
art-cmd mhl-flatten --input hde/ascmhl/ascmhl_chain.xml \
                    --output hde/hde_flat.mhl
```

writes the manifest `hde/hde_flat.mhl`, listing every file of the encoded cards from “Mode `mhl`” with its earliest hash. Its paths are relative to `hde/`, the folder that the History records, wherever the manifest is written, and `verify` reads them relative to the folder of the manifest. So the manifest is written into `hde/`, and `verify --input hde/hde_flat.mhl` checks the files below `hde/`, while a manifest written elsewhere, e.g. next to `hde/`, fails its verification (`mhl-flatten` does not warn of this). A verification of `hde/ascmhl/ascmhl_chain.xml`, however, reports the flattened manifest as an unexpected file, unless a later generation of the History, added as described in “Adding a generation to an ASC MHL History”, ignores it.

E.g.

```
art-cmd mhl --input hde/ --mhl v2-xxh64 --mhl-history \
            --mhl-ignore hde_flat.mhl
```

adds such a generation, after which both verifications pass.

3.7 Mode `export`

E.g.

```
art-cmd export --input arriraw.mxf --output export_dir/
```

exports all metadata (static clip and dynamic frame metadata), audio data and look-file data to

```
./export_dir/
   ├── metadata.json
   ├── audio_track00.wav
   ├── audio_track01.wav
   ├── ...
   ├── LookName.aml/.alf4/.alf4c
   ├── LutNameA.cube
   ├── LutNameA.zip
   ├── LutNameB.cube
   ├── LutNameB.zip
   └── ...
```

Note: The arguments `--skip-metadata`, `--skip-audio` and `--skip-look` can be used to skip a certain part of the export. A clip that writes no file this way (or a `.wav` export of a clip without audio) is skipped with a warning, and a run in which no clip wrote a file fails.

A single clip, also one of frame files given as its directory, is exported directly into the `--output` directory, which must be empty; the log file of the call (`--logpath`) does not count.

3.7.1 Explicit audio or metadata export

By stating a file with explicit extension as `--output` instead of a directory, only audio (as `.wav`) or metadata (as `.json` or `.csv`) can be exported.

E.g.

```
art-cmd export --input arriraw.mxf --output audio.wav
```

exports all audio tracks as WAV to

```
./
├── audio00.wav
├── audio01.wav
└── ...
```

A printf specifier in the file name sets the numbering of the tracks, e.g. `--output audio_%d.wav` gives `audio_0.wav`, `audio_1.wav`, `...`; a `%` of the name is written `%%`. `--skip-metadata` with a `.json` or `.csv` output and `--skip-audio` with a `.wav` output are refused.

3.7.2 Metadata as CSV instead of JSON

The metadata export can also be done in CSV format (instead of the default JSON format) by stating an explicit CSV file as `--output` argument.

E.g.

```
art-cmd export --input arriraw.mxf --duration 100 --output metadata_table.csv
```

exports metadata of the first 100 frames as CSV to `./metadata_table.csv`, having 100 rows containing all dynamic metadata of the frames and also all static metadata (being equal in each column for every frame).

3.7.3 Batch export

Batch processing is also supported for `export` mode.

E.g. having the following `clips` directory

```
./clips/
   ├── arricore.mxf
   ├── arriraw.mxf
   └── subdir/
       └── prores.mxf
```

and calling

```
art-cmd export --input clips/ --output out/metadata_table.csv
```

exports all the clips' metadata to the structure

```
./out/
   ├── arricore/
   │   └── metadata_table.csv
   ├── arriraw/
   │   └── metadata_table.csv
   └── subdir/
       └── prores/
           └── metadata_table.csv
```

A file that is not of a supported clip type is ignored, and a clip whose output cannot be set up (e.g. an output directory that is not empty) is skipped with a warning; a run left with no clip fails. A clip that cannot be read (e.g. a damaged .mxf) or that fails while being processed is logged as an error, and the run exits with error code 1 after the remaining clips.

4 Possible Input Formats of art-cmd

4.1 MXF clip container formats

- MXF/ARRICORE clip containers according to RDD 55 and 61
- MXF/ARRIRAW and MXF/ARRIRAW-HDE clip containers according to RDD 55 and 54
- MXF/ProRes clip containers according to RDD 55 and 44
- MXF *ArriFileV3* clip containers (legacy, will be converted to a representation according to RDD 55 and 54 or 44)

Note: For more details, please refer to:

- https://pub.smpte.org/latest/rdd44/
- https://pub.smpte.org/latest/rdd54/
- https://pub.smpte.org/latest/rdd55/
- https://pub.smpte.org/latest/rdd61/

4.2 MOV/ProRes clip container formats (legacy)

MOV/ProRes clips (from legacy cameras) are also supported as input format, but will be converted to a representation according to RDD 55 and 44.

4.3 ARI and ARX ArriFileV3 frame-files format (legacy)

The single frame file formats `.ari` and the HDE-compressed `.arx` are supported as input. `--input` can be the directory that holds the frame files, one of the frame files, or a printf pattern such as `D401C005_180614_R1ZA.%07d.ari`. They are converted to a representation according to RDD 55 and 54. A frame file given as `--input` is read with the frames of its number pattern, whatever else its directory holds. A frame file whose name ends without a frame number, like the frame grab `Z_20230612_123032_R0D5_l.ari` of an ALEXA 65, is a clip of one frame, given as the file (also next to other frame files) or as its directory. A directory holding several such files, or frames whose numbers have gaps, is no clip; it is reported with the reason. A clip whose header holds no valid creation time gets the time of writing in an MXF output.

Very old ARI formats from D20/21 are not supported.

5 Possible Output Formats of art-cmd

5.1 MXF clip container formats

- MXF/ARRICORE clip containers according to RDD 55 and 61 (only `copy` mode)
- MXF/ARRIRAW clip containers according to RDD 55 and 54 (only `copy` mode)
- MXF/HDE clip containers according to RDD 55 and 54 (`copy` and `process` mode)
- MXF/ProRes clip containers according to RDD 55 and 44 (`copy` and `process` mode)

5.2 EXR files format

In `process` mode by default OpenEXR single frame files are rendered containing ACES `AP0/D60/linear` colors. It is only possible to render to OpenEXR when the `--target-colorspace` argument is set to a linear space, i.e. one of

```
AP0/D60/linear
AWG4/D65/linear
AWG3/D65/linear
```

E.g.

```
art-cmd process --input C_0001C001_250807_130114_c1I82.mxf \
                --target-colorspace AWG4/D65/linear \
                --output out/%07d.exr
```

will render

```
./out/
  ├── 0000000.exr
  ├── 0000001.exr
  └── ...
```

Note: OpenEXR is not supported when processing to non-linear colorspaces. `AWG3/D65/linear` and `AWG3/D65/LogC3` (for OpenEXR, MXF/ProRes and TIFF) are only possible for clips from cameras recording `AWG3/D65/LogC3`; for other clips they are refused (e.g. “Invalid input colorspace AWG4/D65/LogC4 for desired target colorspace AWG3/D65/LogC3”).

5.3 MXF/ProRes file format

MXF/ProRes as output file format is supported when processing to non-linear colorspaces, namely

```
AWG3/D65/LogC3
AWG4/D65/LogC4
P3/D60/DCI
P3/DCI/DCI
P3/D65/DCI
P3/D65/PQ
Rec.2020/D65/BT.1886
Rec.2020/D65/HLG
Rec.2020/D65/PQ
Rec.709/D65/BT.1886
```

For one of these values as `--target-colorspace`, the `--video-codec` must be set to one of

```
prores422proxy
prores422lt
prores422
prores422hq
prores4444
prores4444xq
```

which is then processed to MXF/ProRes.

E.g.

```
art-cmd process --input C_0001C001_250807_130114_c1I82.mxf \
                --target-colorspace Rec.709/D65/BT.1886 \
                --video-codec prores4444xq \
                --output out/
```

will render

```
./out/
  └── C_0001C001_250807_130114_p1I82.mxf
```

Note 1: For MXF/ProRes, `--output` is a directory (the clip is then named after the input) or a file name ending in `.mxf`. MOV/ProRes cannot be written: a name without a known output extension, such as `out.mov`, is taken as a directory and an MXF file is written into it; a name ending in `.exr`, `.tif`, `.wav` etc. is refused.

Note 2: The codec letter in the name of a camera-original clip changes to `_p` for ProRes (see the HDE chapter below).

5.4 MXF/HDE file format

MXF/HDE output is supported for ARRIRAW input (`.ari` or MXF/ARRIRAW) recorded on a Codex medium (e.g. Codex Capture Drive or Codex Compact Drive), and for MXF/HDE input. Other ARRIRAW clips, e.g. from CFast cards, are refused with “HDE encoding is not permitted for this clip”; ARRICORE and ProRes clips are refused with “source clip must be ARRIRAW or already-HDE”.

The `--video-codec` must be set to

```
arriraw_hde
```

which is then processed to MXF/HDE.

E.g.

```
art-cmd process --input C_0001C001_250807_130114_a1I82.mxf \
                --video-codec arriraw_hde \
                --output out/
```

will render

```
./out/
  └── C_0001C001_250807_130114_h1I82.mxf
```

Note 1: For MXF/HDE, `--output` is a directory or a file name ending in `.mxf`. `.arx` sequences cannot be written: a name such as `%07d.arx` is taken as a directory and an MXF file is written into it.

Note 2: If the input file is named after the clip (the `clipName` in its metadata) and that name is a camera-original name of the ALEXA 35 style (e.g. `C_0001C001_250807_130114_a1I82`), the codec letter in the output file name is changed to match the new codec (`_h` for HDE, `_p` for ProRes, `_a` for ARRIRAW): in the example above, `[...]_a[...].mxf` becomes `[...]_h[...].mxf`. This applies to a directory as `--output` when the input file is named exactly like the clip, and to an `--output` file named exactly `<clipName>.mxf`; any other output name, e.g. `<clipName>_v2.mxf`, keeps its name. The `clipName` in the metadata is changed in either case.

Note 3: MXF/HDE is encoded from the ARRIRAW data without rendering it, so the rendering options (`--parameters`, `--cct`, `--tint`, `--asa`, `--deflip`, `--desqueeze`, `--embedded-look`, `--arri-look-file`, `--target-colorspace`, `--letterbox-size`, `--output-width` and `--output-height`) are refused with `--video-codec arriraw_hde`.

5.5 TIFF files format

TIFF output to uncompressed 16bit RGB buffers is also supported (for all non-linear colorspaces; a linear `--target-colorspace` is refused) when set as extension of the `--output` argument.

E.g.

```
art-cmd process --input C_0001C001_250807_130114_c1I82.mxf \
                --target-colorspace Rec.2020/D65/PQ \
                --output out/%07d.tif
```

will render following TIFF files

```
./out/
  ├── 0000000.tif
  ├── 0000001.tif
  └── ...
```

with color values in `Rec.2020/D65/PQ`.

6 Support Contact

If you have any questions about the application, please contact us via `digitalworkflow@arri.de`.

7 Legal Notices

Please read contents of `EULA.txt`.