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

Contents

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 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).
- *copy* video-, audio- and metadata from various sources into a 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.

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, all readable clips contained in the directory tree below, are considered to be processed as a batch according to the additional arguments stated whenever applicable. Each mode implies some defaults for the output that is created (by default in a directory named like the mode's argument value in the current working directory, when no explicit output destination is given).

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 subdirectory `process/` of the current working directory by default (i.e. `--output process/%07d.exr`).

By default containing 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` and `export` 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 requirend `--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 frames 1800 to 10800 of the original clip.

The `--start` and `--duration` are general arguments, restricting the range of frames from input clip(s) some mode is acting on, that can be stated whenever it is desired to render a new clip as a cut from the original clip.

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 seqence of 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/%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 invidual calls.

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 explicitely 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.

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 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).

3 Modes of operation of art-cmd

3.1 Mode process

In process mode, image data read from the given input clip(s) are altered while rendering a new clip containing RGB image data.

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 a 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 its possible values, there are also the specific XSD-schemas ArriCoreDecoderParameters.xsd and ArriRawDecoderParameters.xsd used by the ARRI Image SDK integrated in art-cmd.

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
```

.

3.1.3 Target colorspace selection and looks

The `--ls-target-colorspaces` argument 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 a MXF/ARRICORE clip to MXF/ProRes.

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 colorspaces supported by the (clip-embedded or external) look file.

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: The colorspace listed first is always the default (in case no `--target-colorspace` is set at all).

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.

E.g.

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

(where the input `prores.mxf` has a resolution of 2880x1620) renders a MXF/ProRes with image data of resolution 1920x1080 as `./process/prores.mxf` output.

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 a MXF/ProRes file, containing metadata as standarized in SMPTE ST 377-1 (MXF), RDD 44 (Apple) plus RDD 55 (ARRI).

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 a MXF/ARRIRAW file, containing metadata as standarized 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 frames 1800 to 10800 of the original clip.

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: During recovery, the file size is also reduced until the last valid image data. The difference in file size is due to the fact that some emtpy storage space allocated on the recording medium during recording is freed up again by the recovery.

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, not supported for MOV, MXF/ProRes and MXF/ARRICORE input because of missing checksum metadata.

3.3.2 pairwise verification of clips

When providing more than one `--input` argument, clips are verified pairwise by comparing video-, audio- and metadata between the clips.

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 that 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

Is is also possible to create a JSON report file, conforming to the `verify_report.schema.json`, where verification results (and some additional info) is 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.

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 (that are required to have a structurally identical file tree underneath) and create a `batch_pair_verify.json` file with results for each clip pair verified.

3.4 Mode `export`

E.g.

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

exports all metadata (static clip- 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.

3.4.1 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
└── ...
```

3.4.2 Metadata as CSV instead of JSON

The metadata export can also be done in CSV format (instead of the default JSON format) by stating a 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).

3.4.3 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
```

.

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 format (by stating the directory containing all files as `--input`), but will be converted to a representation according to RDD 55 and 54.

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/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 and `AWG3/D65/linear` is only possible for clips from cameras recording `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

```
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_c1I82.mxf
```

Note: Filenames ending with `.mxf` are required for the `--output` argument when rendering MXF/ProRes, as writing MOV/ProRes containers is not supported.

5.4 TIFF files format

TIFF output to uncompressed 16bit RGB buffers is also supported (for all non-linear colorspaces) 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`.