In [1]:
```
# Cell 0 imports & paths
import pathlib, json, subprocess, shlex, textwrap, sys

ASSETS = pathlib.Path("assets")
SPEC   = {
    "container": "mov,mp4,m4a,3gp,3g2,mj2",  # what ffprobe returns for ISO BMFF
    "vcodec"   : "h264",
    "width"    : 640,
    "height"   : 360,
    "fps"      : 25,
    "dar"      : "16:9",
    "acodec"   : "aac",
    "achans"   : 2,
    "abitrate" : 192000,    # bits per second
}
```

In [2]:
```
# Cell 1 helper to call ffprobe
def probe(path: pathlib.Path) -> dict:
    """
    Return ffprobe json dict (streams + format).
    """
    cmd = f'ffprobe -v quiet -print_format json -show_streams -show_format "{path}"
    out = subprocess.check_output(shlex.split(cmd), text=True)
    return json.loads(out)
```

In [3]:
```
# Cell 2 QC checks
report_lines = []
bad_files    = []

for video in ASSETS.iterdir():
    if video.suffix.lower() not in {".mp4", ".mov", ".avi", ".mkv"}:
        continue
    meta = probe(video)
    fmt  = meta["format"]
    vstr = next(s for s in meta["streams"] if s["codec_type"] == "video")
    astr = next(s for s in meta["streams"] if s["codec_type"] == "audio")

    problems = []

    # container
    if fmt["format_name"] not in SPEC["container"]:
        problems.append(f"container {fmt['format_name']}")

    # video stream checks
    if vstr["codec_name"] != SPEC["vcodec"]:
        problems.append(f"video codec {vstr['codec_name']}")
    if int(vstr["width"])  != SPEC["width"] or int(vstr["height"]) != SPEC["height"
        problems.append(f"{vstr['width']}×{vstr['height']}")
    # fps
    num, den = map(int, vstr["r_frame_rate"].split("/"))
    fps = num / den
    if abs(fps - SPEC["fps"]) > 0.1:
        problems.append(f"{fps:.2f} fps")
    # display aspect ratio
    if vstr.get("display_aspect_ratio") != SPEC["dar"]:
        problems.append(f"DAR {vstr.get('display_aspect_ratio','?')}")

    # audio stream checks
    if astr["codec_name"] != SPEC["acodec"]:
        problems.append(f"audio codec {astr['codec_name']}")
    if int(astr.get("channels", 0)) != SPEC["achans"]:
        problems.append(f"{astr.get('channels')} ch")
    if int(astr.get("bit_rate", 0)) < SPEC["abitrate"]:
        problems.append(f"audio ≤ {int(astr.get('bit_rate',0))//1000} kb/s")

    if problems:
        bad_files.append(video)
        report_lines.append(f"{video.name}  –  " + ", ".join(problems))
    else:
        report_lines.append(f"{video.name}  –  OK")

# write report.txt
(ASSETS.parent / "report.txt").write_text("\n".join(report_lines))
print("\n".join(report_lines))
```

```
Cosmos_War_of_the_Planets.mp4  –  628×354, 29.97 fps, DAR 314:177
Last_man_on_earth_1964.mov  –  video codec prores, 23.98 fps, audio codec pcm_s16le
The_Gun_and_the_Pulpit.avi  –  container avi, video codec rawvideo, 720×404, DAR ?,
audio codec pcm_s16le
The_Hill_Gang_Rides_Again.mp4  –  OK
Voyage_to_the_Planet_of_Prehistoric_Women.mp4  –  video codec hevc, 29.97 fps, audio
codec mp3
```

In [4]:
```
# Cell 3 convert the bad files
for video in bad_files:
    out = video.with_stem(video.stem + "_formatOK").with_suffix(".mp4")
    cmd = [
        "ffmpeg", "-y", "-i", str(video),
        # video
        "-c:v", "libx264", "-preset", "slow", "-b:v", "3M",
        "-vf", "scale=640:360,fps=25,setdar=16/9",
        # audio
        "-c:a", "aac", "-b:a", "192k", "-ac", "2",
        str(out)
    ]
    print(">>", " ".join(cmd))
    subprocess.run(cmd, check=True)
print("Conversion done:", len(bad_files), "files fixed.")
```

```
>> ffmpeg -y -i assets\Cosmos_War_of_the_Planets.mp4 -c:v libx264 -preset slow -b:v
3M -vf scale=640:360,fps=25,setdar=16/9 -c:a aac -b:a 192k -ac 2 assets\Cosmos_War_o
f_the_Planets_formatOK.mp4
>> ffmpeg -y -i assets\Last_man_on_earth_1964.mov -c:v libx264 -preset slow -b:v 3M
-vf scale=640:360,fps=25,setdar=16/9 -c:a aac -b:a 192k -ac 2 assets\Last_man_on_ear
th_1964_formatOK.mp4
>> ffmpeg -y -i assets\The_Gun_and_the_Pulpit.avi -c:v libx264 -preset slow -b:v 3M
-vf scale=640:360,fps=25,setdar=16/9 -c:a aac -b:a 192k -ac 2 assets\The_Gun_and_the
_Pulpit_formatOK.mp4
>> ffmpeg -y -i assets\Voyage_to_the_Planet_of_Prehistoric_Women.mp4 -c:v libx264 -p
reset slow -b:v 3M -vf scale=640:360,fps=25,setdar=16/9 -c:a aac -b:a 192k -ac 2 ass
ets\Voyage_to_the_Planet_of_Prehistoric_Women_formatOK.mp4
Conversion done: 4 files fixed.
```

Quick media-format glossary **
***Container vs. codec* – MP4, MOV, AVI are "boxes"; H.264 and AAC are the payload. A file can be MP4 outside but still hold the wrong codecs.**
***Frame-rate (fps)* – The festival wants **25 fps**, the European broadcast standard. A variable or 29.97 fps clip stutters on PAL equipment, so we force `fps=25`.
Resolution & DAR – Scaling to **640×360** keeps the pixels square; `setdar=16/9` tells players the intended display shape.
Bit-rate – `-b:v 3M` is plenty for SD H.264; audio at **192 kb/s stereo** is transparent quality yet small.
`libx264 preset slow` – balances encoding time and quality; "slow" maximises PSNR at this resolution.
`ffprobe` exposes every field in JSON, so the QC script can reject any clip that drifts from these numbers and then `ffmpeg` fixes them in one pass.