转码内核系统总结

郑建峰 <jianfengzheng@pptv.com>

# Summary

## Transcli Dir Layout

|  |  |  |
| --- | --- | --- |
| bin/ | external  binary | Ffmpeg,x264,x625,mediainfo,mp4box  Preset/\*.xml |
| common/ |  |  |
| doc/ |  |  |
| faac/ | external | AAC encoder |
| ipch/ |  |  |
| lame/ | external | MP3 encoder |
| lib/  include/ | external  archive | External lib |
| libsamplerate/ | external  source | Audio resample  http://www.mega-nerd.com/libsamplerate/ |
| libxml2/ | external  source |  |
| strpro/ | external? |  |
| transcli/ | internal | xml adjust (user xml -> internal prefs)  see “CCliHelperPPLive::AdjustPreset()” |
| transnode/ | internal | 1. Wrapper for avcodec & avformat 2. Trans scheduler: “CTransWorker” |
| watermark/ | internal |  |

## Internal filters

|  |  |
| --- | --- |
| 功能 | 代码 |
| autovolgain | common/zml\_gain\_analysis.c |
| logo | watermark/WaterMarkFilter.cpp |
| imagetail | transnode/TransWorkerSeperate.cpp |
| thumbnail | watermark/Thumbnailfilter.cpp |
| video enhance | transnode/videoEnhancer.c  CVideoEnhancer::process() |

## ffmpeg usage

### decoding

GetCmdString()

### AudioFilter

GenAudioFilterOptions()

|  |  |
| --- | --- |
| 功能 | ffmpeg -af |
| channels.mix | ffmpeg -af pan |
| volGain | ffmpeg -af volumn |

#### channels.mix (ffmpeg -af pan)

#### volGain (ffmpeg -af volumn)

### VideoFilter

GenVideoFilterOptions()

|  |  |
| --- | --- |
| 功能 | ffmpeg video f |
| deinterlace | ffmpeg -vf tinterlace  ffmpeg -vf yadif |
| fps change | ffmpeg -vf fps |
| denoise | ffmpeg -vf hqdn3d |
| delogo | ffmpeg -af delogo |
| cropdetect | ffmpeg -vf cropdetect |
| crop | ffmpeg -vf crop |
| scale | ffmpeg -vf scale |
| expand | ffmpeg -vf pad |
| rotate | ffmpeg -vf rotate |
| image subtitle | ffmpeg -vf overlay |
| text subtitle | ffmpeg -vf ass/subtitle |

## 旧版本ffmpeg 扩展

https://github.com/zoominla/ffmpeg.git

0d3262fbe24ec4219e143cdb04b39192e294cc9c

### -vf delogo

添加了开始时间/结束时间选项

### -vf ass

将fonts.conf文件定为${transcli所在文件夹}/codecs/fonts/fonts.conf

### -vf sclae

这里的扩展选项应该不会用到

### timestamp work around

|  |
| --- |
| Index: libavfilter/vf\_fps.c  ===================================================================  --- libavfilter/vf\_fps.c (revision 152520)  +++ libavfilter/vf\_fps.c (working copy)  @@ -225,6 +225,12 @@  return ret;  }    + // Work around for timestamp sudden change(prevent dup much more frames which result in  + // out of memory error), limit to 6 seconds  + if(delta > av\_q2d(s->framerate)\*CONPENSATE\_TS\_SUDDEN\_CHANGE\_MAX\_DURATION) {  + delta = 1;  + }  +  /\* can output >= 1 frames \*/  for (i = 0; i < delta; i++) {  AVFrame \*buf\_out;  Index: libavutil/opt.h  ===================================================================  --- libavutil/opt.h (revision 152520)  +++ libavutil/opt.h (working copy)  @@ -907,4 +907,7 @@  \* @}  \*/    +// Max conpensate timestamp sudden change duration is 8 seconds  +#define CONPENSATE\_TS\_SUDDEN\_CHANGE\_MAX\_DURATION 8  +  #endif /\* AVUTIL\_OPT\_H \*/  Index: libswresample/swresample.c  ===================================================================  --- libswresample/swresample.c (revision 152520)  +++ libswresample/swresample.c (working copy)  @@ -908,7 +908,8 @@  int64\_t delta = pts - swr\_get\_delay(s, s->in\_sample\_rate \* (int64\_t)s->out\_sample\_rate) - s->outpts + s->drop\_output\*(int64\_t)s->in\_sample\_rate;  double fdelta = delta /(double)(s->in\_sample\_rate \* (int64\_t)s->out\_sample\_rate);    - if(fabs(fdelta) > s->min\_compensation) {  + // Hack for preventing insert many empty audio(if delta is bigger than 6 seconds, no compensation)  + if(fabs(fdelta) > s->min\_compensation && fabs(fdelta) < CONPENSATE\_TS\_SUDDEN\_CHANGE\_MAX\_DURATION) {  if(s->outpts == s->firstpts || fabs(fdelta) > s->min\_hard\_compensation){  int ret;  if(delta > 0) ret = swr\_inject\_silence(s, delta / s->out\_sample\_rate); |

## x264 集成

以lib的方式集成到transcli

### 初始化

设置参数：bool CX264Encode::Initialize()

### 帧编码

CTransWorkerSeperate::transcodeSingleVideo()

CTransWorkerSeperate::transcodeSingleVideoComplex()

CTransWorkerSeperate::encodeVideo(size\_t videoIdx)

{

--> pVideoEncode->EncodeFrame(pFrameBuf, -1 /\*TODO:BufSize\*/);

}

### 降码率（lowerbitrate）

降码率算法是在x264中实现的一个额外功能，由转码内核决定是否开启。其实现位于

<http://svn01/repositories/r02/Cloud/codecs/x264-mod/ratecontroll.c>

的init\_pass2()函数中。

# XML 参数解析 (User->Internal)

* 读取XML文件并解析
  + 参见GetConfigFromXml()
* 对XML进行内部调整
  + 参见AdjustPreset()
  + 将用户配置调整为内部相对扁平的xml模板，参见pplive.cp的prefsTemplate[]定义
  + 内部xml的节点一般以overall开头，如“overall.audio.codec”
  + 如果某些选项用户未配置，会使用prefsTemplate[]中的默认值。

## Source

### Type

视频的种类会对单位/码率有所影响。例如如果是音乐就会适当调高音频码率。

|  |
| --- |
| // Ajust audio bitrate according to preset level and source type  // 0:Music 1:Movie 2:Episode 3:Anime 4:TVShow 5:Sport 6:Game  if(conf.source.type == 0) { // Increase audio bitrate of music  conf.target.acodec.bitrate += ppl\_def[presetLevel].abr\_inc\_for\_music;  } |

### Demuxer

|  |  |
| --- | --- |
| User | <demuxer>0</demuxer>  <!--0:default, 1: (deprecated?) lavf, change demuxer of ts file--> |
| Config | config->source.demuxer |

### Audio

* track=0时，根据stream选择其中一条音轨
* track=1时选取所有音轨，stream这时不起作用
* track=2表示把一条源轨的左右channel映射为两条输出音轨——该功能可能已经无效

|  |  |
| --- | --- |
| User | <audio stream="0" channel="both" decoder="auto" track="0"/>  <!--track=0:single, 1:original, 2:Dual(deprecated?)-->  <!--stream=$(trackid), select stream when track==0>  <!--channel="both", "left", "right", "mix", "original"-->  <!--decoder=”Mplayer”, "Mencoder", "FFMpeg", "AVS", "Copy", “auto”--> |
| Internal | “extension.audio.track1" = conf.source.audio\_stream  “extension.audio.tracknum” = conf.source.track\_config  “overall.audio.source” = conf.source.audio\_decoder  "audiofilter.channels.mix"  “overall.audio.channels” = 0:ori, 1:left, 2:right, 3:both/mix |

### Video

|  |  |
| --- | --- |
| Default | <node key=\"videofilter.crop.mode\">3</node> |
| User | <transcode><source>  <video stream="0" rectangle="{0,0,480,360}"  crop="auto" threshold="26" decoder="auto"/>  <!--crop: disable(0)/manual(1)/auto(4)/calc(2)/expand(3)-->  <!--rectangle: can be absolute coordinate {0,0,480,360}  or relative {0,0.126,1,0.92}--> |
| Config | config->source.video\_stream |
| Internal  (crop) | “videofilter.crop.detectLimit" = conf.source.threshold  “videofilter.crop.mode” = conf.source.crop\_mode  "videofilter.crop.left" = getl(config->source.video\_rect)  "videofilter.crop.top" = gett(config->source.video\_rect)  "videofilter.crop.width" = getw(config->source.video\_rect)  "videofilter.crop.height" = geth(config->source.video\_rect) |
| Internal  (expand) | "videofilter.expand.enabled"  "videofilter.expand.x"  "videofilter.expand.y"  "videofilter.expand.width“  "videofilter.expand.height" |
|  | |
| CTransWorkerSeperate::initialize()  {  performBlackBandAutoDetect()  calcDarWhenCropAndNormalizeCrop(pVideoEnc);  startDecoder()->Start()->GetCmdString(  AutoCropOrAddBand(  case cropMode in:  "calc") CalcAutoCrop();  "expand") ... nothing ...;  );  GenVideoFilterOptions(  case cropMode in:  "manual"|"calc"|"auto") ffmpeg -vf crop;  "expand") ffmpeg -vf pad;  );  )  } | |

#### cropmode

* cropmode = 1（manual），根据rectangle进行crop
* cropMode = 4（auto），根据threshold自动去探测黑边大小(ffmpeg -vf cropdetect)

|  |
| --- |
| bool CTransWorkerSeperate::initialize()  performBlackBandAutoDetect(srcFile, m\_videoEncs[0]);  CYuvUtil::AutoDetectCrop(const char\* fileName, int duration, int detectThreshold)  {  sprintf(vfStr, " -vf \"select=isnan(prev\_selected\_t)+gte(t-prev\_selected\_t\\,%d),cropdetect=%d:4\"", selectInterval, detectThreshold);  } |

* cropMode=2(calc)/3(expand), 根据DAR/SAR适配宽高比（16:9 <--> 4:3）的一种情况。只不过calc在scale之前进行截取，expand是在scale之后加黑边。
  + 参见

<http://www.animemusicvideos.org/guides/avtech3/theory-videoaspectratios.html>

<https://en.wikipedia.org/wiki/Pan_and_scan>

* + 内部计算过程见CDecoder::AutoCropOrAddBand()

### Clip

|  |  |
| --- | --- |
| User | <transcode><source>  <clipmode>0</clipmode>  <!--0:frame accurate(transcoding), 1:I-frame accurate(may be 5 secs error)-->  <clip start="00:00:02.0" end="00:01:10.0"/>  <clip start="00:01:20.0" end="00:01:40.0"/>  <!-- ... clips ... --> |
| Parse | config->source.clip\_mode  config->source.clipStartSet.push\_back(startTime);  config->source.clipEndSet.push\_back(endTime); |
| Internal | overall.task.losslessClip  overall.clips.count  overall.clips.start%d  overall.clips.end%d |
| Class | m\_clipStartSet  m\_clipEndSet |
| trans | if(m\_bDecodeNext || m\_pSplitter || !m\_clipStartSet.empty()) {  ret = transcodeSingleVideoComplex();  } else { // Simple mode  ret = transcodeSingleVideo();  } |
|  | |
|  | |

### delogo

delogo使用了”[ffmpeg -vf delogo](#_delogo_(ffmpeg_-vf)”, 并使扩展使用了start time和end time两个选项。

|  |  |
| --- | --- |
| default |  |
| User | <pos position="{135,50,223,120}" start="00:00:00.0" end="23:00:00" /> |
| Config | config->source.delogo\_num  config->source.delogo\_pos[...] |
| Internal | videofilter.delogo.enabled  videofilter.delogo.num  videofilter.delogo.pos%d |
| trans | use ffmpeg -vf delogo |
| Class | int rateCtrlMode = prefs->GetInt("overall.video.mode");  if(rateCtrlMode >= RC\_MODE\_2PASS && !m\_bInsertBlankVideo) {  m\_encoderPass = 2 /\*rateCtrlMode -1\*/;  } |
|  | |
|  | |

## target Core/CPU

|  |  |
| --- | --- |
| User | <cores>0</cores>  <!--Transcli running on some cores, default is 0, running on all cores.--> |
| config | config->target.coresNum |
| Internal | "overall.task.cores" |
| Class |  |
| trans |  |
|  | |
| // Get cores number  int runOnCores = pTaskPref->GetInt("overall.task.cores");  if(runOnCores <= 0) {  // Check environment variables  char\* strUseCores = getenv("TRANSCLI\_USE\_CORES");  if(strUseCores && (\*strUseCores)) {  runOnCores = atoi(strUseCores);  }  }  if(runOnCores > 0) {  CTransnodeUtils::BindToCores(runOnCores);  int threasNum = (int)(runOnCores\*1.5);  pTaskPref->SetInt("videoenc.x264.threads", threasNum);  pTaskPref->SetInt("videoenc.x265.threads", threasNum);  } | |

## target Audio

|  |  |
| --- | --- |
| User | <audio disable="false">  <codec format="aac" bitrate="48k" samplerate="24000"  timeshift="1.5s" brdown="true"/>  <volgain mode="auto" value="0.0"/>  </audio> |
| config | config->target.disable\_audio |
| Internal | "overall.audio.enabled" |
| Class | pAudioEnc = createAudioEncoder(encType, aFormat);  pEncoder->SetAudioInfo(audioInfo(fmt, br, sample,...), prefs);  m\_audioEncs.push\_back(pAudioEnc); |

### codec

|  |  |
| --- | --- |
| User | <codec format="aac" bitrate="48k" samplerate="24000" timeshift="1.5s" brdown="true"/>  <!--timeshift:not use-->  <!--brdown: 比较源视频的音频码率和设置的码率。选择其中一个比较小的值 --> |
| config | config->target.acodec |

#### format (codec-type)

|  |  |
| --- | --- |
| default |  |
| User | <codec format="aac" bitrate="48k" samplerate="24000" timeshift="1.5s" brdown="true"/> |
|  | struct {  const char \* name;  int fmt;  const char \*psz\_fmt;  int encoder;  const char \*psz\_encoder;  } audio\_map[] = {  /\*{"aac", AC\_AAC\_HE, "HE-AAC", AE\_NEROREF, "Nero Encoder"},  {"lcaac", AC\_AAC\_LC, "LC-AAC", AE\_FAAC, "FAAC"},\*/  {"aac", AC\_AAC\_HE, "HE-AAC", AE\_FDK, "FDK AAC"},  {"lcaac", AC\_AAC\_LC, "LC-AAC", AE\_FDK, "FDK AAC"},  {"mp3", AC\_MP3, "MP3", AE\_FFMPEG, "FFmpeg"},  {"mp2", AC\_MP2, "MP2", AE\_FFMPEG, "FFmpeg"},  {"ac3", AC\_AC3, "AC3", AE\_FFMPEG, "FFmpeg"},  {"eac3", AC\_EAC3, "E-AC3", AE\_DOLBY, "Dolby Encoder"},  {"copy", AC\_AC3, "AC3", AE\_FFMPEG, "FFmpeg"},  {"amr", AC\_AMR, "AMR", AE\_FFMPEG, "FFmpeg"},  {"disable", AC\_AC3, "AC3", AE\_FFMPEG, "FFmpeg"},  {"", 0, 0}  }; |
| config | config->target.acodec.name |
| Internal | "overall.audio.enabled" = (format != disable”)  "overall.audio.copy" = (format == “copy”)  "overall.video.autoSource" = (format != “copy”)  "overall.audio.format" = audio\_map[idx].psz\_fmt  "overall.audio.encoder" = audio\_map[idx].psz\_encoder |
|  | |
| if(strcmp(conf.target.acodec.name, "disable") == 0) { // disable audio  prefs.SetStreamPref("overall.audio.enabled", false, STAUDIO);  }  if(strcmp(conf.target.acodec.name, "copy") == 0) { // Copy audio  prefs.SetStreamPref("overall.audio.copy", true, STAUDIO);  prefs.SetStreamPref("overall.video.autoSource", false, STVIDEO);  }  prefs.SetStreamPref("overall.audio.format", audio\_map[idx].psz\_fmt, STAUDIO);  prefs.SetStreamPref("overall.audio.encoder", audio\_map[idx].psz\_encoder, STAUDIO);  if (audio\_map[idx].encoder == AE\_NEROREF) {  prefs.SetStreamPref("audioenc.nero.mode", 1, STAUDIO);  } else if(audio\_map[idx].encoder == AE\_DOLBY) {  // E-AC3 only support 48k samplerate, should up sample  prefs.SetStreamPref("audiofilter.resample.downSamplingOnly", false, STAUDIO);  } else if (audio\_map[idx].encoder == AE\_FDK) {  if (audio\_map[idx].fmt == AC\_AAC\_LC) {  prefs.SetStreamPref("audioenc.fdkaac.profile", "MPEG4 LC", STAUDIO);  }  } | |

#### bitrate

|  |  |
| --- | --- |
| default |  |
| User | <codec format="aac" bitrate="48k" samplerate="24000" timeshift="1.5s" brdown="true"/> |
| config | config->target.acodec.bitrate |
| Internal | "audioenc.faac.bitrate"  "audioenc.fdkaac.bitrate"  "audioenc.ffmpeg.bitrate"  "audioenc.nero.bitrate"  "audioenc.dolby.bitrate"  "audioenc.faac.mode" |
|  | |
| if (conf.target.acodec.bitrate > 0) {  prefs.SetStreamPref("audioenc.faac.bitrate", conf.target.acodec.bitrate, STAUDIO);  prefs.SetStreamPref("audioenc.fdkaac.bitrate", conf.target.acodec.bitrate, STAUDIO);  prefs.SetStreamPref("audioenc.ffmpeg.bitrate", conf.target.acodec.bitrate, STAUDIO);  prefs.SetStreamPref("audioenc.nero.bitrate", conf.target.acodec.bitrate, STAUDIO);  if (conf.target.acodec.bitrate == 96/192/256/384/448/...) {  dolbyBrcode = 0/1/2/3/4/5;  }  prefs.SetStreamPref("audioenc.dolby.bitrate", dolbyBrcode, STAUDIO);  prefs.SetStreamPref("audioenc.faac.mode", 1, STAUDIO);  } | |

#### samplerate

|  |  |
| --- | --- |
| default | <node key=\"audiofilter.resample.samplerate\">44100</node> |
| User | <codec format="aac" bitrate="48k" samplerate="24000" timeshift="1.5s" brdown="true"/> |
| config | config->target.acodec.samplerate |
| Internal | "audiofilter.resample.samplerate" |

#### timeshift

在ffmpeg里应该没有用到

|  |  |
| --- | --- |
| default |  |
| User | <codec format="aac" bitrate="48k" samplerate="24000" timeshift="1.5s" brdown="true"/> |
| config | config->target.acodec.timeshift |
| Internal | "overall.audio.delay1" |

#### brdown

|  |  |
| --- | --- |
| default | config->target.acodec.brdown = 1 |
| User | <codec format="aac" bitrate="48k" samplerate="24000" timeshift="1.5s" brdown="true"/>  <!--brdown: 比较源视频的音频码率和设置的码率。选择其中一个比较小的值 --> |
| config | config->target.acodec.brdown |
| Internal | "audiofilter.extra.brdown" |
|  | |
| CTransWorker::setAudioEncAttrib ()  {  if(audioPref->GetBoolean("audiofilter.extra.brdown")) {  ...  // If brSource < 8k, may be it's wrongly detected by ffprobe  if(brSource > 8 && brSetting > 0 && brSource < brSetting) {  brSetting = brSource;  }  ...  // If bitrate of single channel is too high, then it will fail.  if(brSetting / pAInfo->out\_channels > 96) {  brSetting = pAInfo->out\_channels \* 96;  }  audioPref->SetInt(brField, brSetting);  }  } | |

### volgain

|  |  |
| --- | --- |
| default | config->target.acodec.autoVolGain= 1 |
| User | <volgain mode="auto" value="0.0"/>  <!--mode:auto/manual,default is auto. value range:-200~60--> |
| config | config->target.acodec.autoVolGain |
| Internal | "audiofilter.volume.autoGain"  "audiofilter.volume.gain" |
| Class | m\_bAutoVolumeGain = audioPref->GetBoolean("audiofilter.volume.autoGain"); |

#### autoGain (common/zml\_gain\_analysis.c)

analyseMainAudioTrack()

auxAudioAnalyseEntry()

common/zml\_gain\_analysis.c

## target Video

|  |  |
| --- | --- |
| default | <node key=\"overall.video.enabled\">true</node> |
| default | config->target.vcodec.darNum = -1;  config->target.vcodec.darDen = -1;  config->target.vcodec.lower\_bitrate = 1; // Default enable lower bitrate  config->target.vcodec.video\_enhance = 1; // Default enable video enhance  config->target.vcodec.bframes = -1;  config->target.vcodec.encoding = 1;  config->target.vcodec.brdown = 1; |
| User | <video disable="false">  <profile>high</profile> <!--H.264:auto/baseline/main/high | HEVC:main-->  <lowerbitrate>1</lowerbitrate>  <!--0:disable decreasing bitrate, 1:enable(default)-->  <videoenhance>1</videoenhance>  <!--0:disable video enhance, 1:enable(default)-->  <rcmode>4</rcmode>  <!--1:ABR, 2:CRF, 3:CBR, 4:2-pass-->  <ref>4</ref>  <!--Reframes, default is 4-->  <codec format="h264" bitrate="380k"  rectangle="{0,0,176,144}" position="{10,10,166,134}"  encode="true" nobigger="ture" brdown="true"/>  <!--encode:true(do video encoding), false(not do video encoding)-->  <!--nobigger:true(?), false(?)-->  <fps>0</fps>  <dar>0.0</dar>  <!--Float value, Should be parsed to darNum and darDen-->  <keyint>5</keyint>  <!--Key frame interval. If keyint < 10, it's senconds else it's frame num.-->  <bframes>4</bframes>  <!--B frames num. Range:[0,16], default is 4.-->  </video> |
| config | config->target.disable\_video |
| Internal | "overall.video.enabled" |
| Class |  |
| trans |  |
|  | |
|  | |

### codec

|  |  |
| --- | --- |
| User | <codec format="h264" bitrate="380k"  rectangle="{0,0,176,144}" position="{10,10,166,134}"  encode="true" nobigger="ture"/> |

#### format (codec-type)

|  |  |
| --- | --- |
| User | <codec format="h264" ... encode="true"/>  <!--encode:true(do video encoding), false(not do video encoding)-->  <!--format:”h264”,”copy”,”disable”,... --> |
|  | struct {  const char \* name;  int fmt;  const char \*psz\_fmt;  int encoder;  const char \*psz\_encoder;  } video\_map[] = {  {"h264", VC\_H264, "H.264", VE\_X264, "X264"},  {"xvid", VC\_XVID, "XVID", VE\_XVID, "XVID"},  {"h263", VC\_H263, "H.263", VE\_FFMPEG, "FFmpeg"},  {"wmv9", VC\_WMV9, "WMV9", VE\_FFMPEG, "FFmpeg"},  {"mpeg2", VC\_MPEG2, "MPEG2", VE\_FFMPEG, "FFmpeg"},  {"mii", VC\_MII, "Mii", VE\_MII, "Mii Encoder"},  {"copy", VC\_RAW, "H.264", VE\_X264, "X264"},  {"hevc", VC\_HEVC, "HEVC", VE\_X265, "x265"},  {"disable", VC\_RAW, "H.264", VE\_X264, "X264"},  {"", 0, 0}  }; |
| Config | config->target.vcodec.name |
| Internal | "overall.video.enabled"  "overall.video.autoSource"  "overall.video.source"  "overall.video.format" = video\_map[idx].psz\_fmt  "overall.video.encoder" = video\_map[idx].psz\_encoder |
|  | |
| if(strcmp(conf.target.vcodec.name, "copy") == 0) { // Copy video  prefs.SetStreamPref("overall.video.copy", true, STVIDEO);  prefs.SetStreamPref("[overall.audio.source](#_decoder)", AD\_FFMPEG, STAUDIO);  prefs.SetStreamPref("[overall.video.autoSource](#_decoder_1)", false, STAUDIO);  } | |

#### bitrate

|  |  |
| --- | --- |
| default | <node key=\"overall.video.bitrate\">320</node> |
| User | <codec rectangle="{0,0,176,144}" position="{10,10,166,134}"/> |
| config | config->target.vcodec.bitrate |
| Internal | "overall.video.bitrate" |
| Class |  |
| trans |  |
|  | |
|  | |

##### presetLevel

|  |
| --- |
| AdjustPreset()  {  // Ajust audio bitrate according to preset level and source type  // 0:Music 1:Movie 2:Episode 3:Anime 4:TVShow 5:Sport 6:Game  if(conf.source.type == 0) { // Increase audio bitrate of music  conf.target.acodec.bitrate += ppl\_def[presetLevel].abr\_inc\_for\_music;  } } |
| void CTransWorker::adjustVideoOutParam(CVideoEncoder\* pVideoEnc, int overallBr)  {  ...  if(pvInfo && pPref) {  // If original video highest level is HD/SD, and source bitrate is low  int presetLevel = pPref->GetInt("overall.task.ppLevel");  if(presetLevel == PPTV\_LEVEL\_720) {  if(inWidth >= 600 && inWidth < 1000) { // Highest level is HD  if(brSource > 0 && brSource < 590) {  pPref->SetInt("overall.video.bitrate", brSource);  pPref->SetInt("videoenc.x264.targetQp", 0);  }  }  } else if(presetLevel == PPTV\_LEVEL\_360) {  if(inWidth < 600) { // Highest level is SD  if(brSource > 0 && brSource < 300) {  pPref->SetInt("overall.video.bitrate", brSource);  pPref->SetInt("videoenc.x264.targetQp", 0);  }  }  }  } |

#### rectangle & position

|  |
| --- |
| AdjustPreset()  {  int picW = destPos.right - destPos.left;  int picH = destPos.bottom - destPos.top;  if(picW <= 0 && picH <= 0) {  int destW = destVideoSize.right;  int destH = destVideoSize.bottom;  if(destW > 0 || destH > 0) {  picW = destW;  picH = destH;  }  }  prefs.SetStreamPref("videofilter.scale.enabled", true, STVIDEO);  prefs.SetStreamPref("videofilter.scale.width", picW, STVIDEO);  prefs.SetStreamPref("videofilter.scale.height", picH, STVIDEO); } |

* position只决定scale的宽高，不会做cropping
* rectangle在(picW <= 0 && picH <= 0)的情况下起作用
* 如果rect.right或rect.bottom其中一个小于0，另一个根据宽高比得出

|  |  |
| --- | --- |
| default | <node key=\"videofilter.scale.enabled\">true</node>\n\  <node key=\"videofilter.scale.width\">320</node>\n\  <node key=\"videofilter.scale.height\">240</node>\n\ |
| User | <codec rectangle="{0,0,176,144}" position="{10,10,166,134}"/> |
| config | config->target.vcodec.rect  config->target.vcodec.pos |
| Internal | "videofilter.scale.enabled"  "videofilter.scale.width"  "videofilter.scale.height" |
| Class | m\_pVInfo->res\_out |
| trans |  |
|  | |
| setVideoPref()  {  if(prefs->GetBoolean("videofilter.scale.enabled")) {  outSize.width = prefs->GetInt("videofilter.scale.width");  outSize.height = prefs->GetInt("videofilter.scale.height");  }  ...  video\_info\_t videoInfo = {..., outSize, ...};  ...  pEncoder->SetVideoInfo(videoInfo, prefs);  } | |

#### nobigger

|  |  |
| --- | --- |
| User | <codec nobigger=”true”/”false”/> |
| config | config->target.vcodec.dim\_nobigger |
| Internal | "videofilter.scale.scaleDown"  // Only scale down, no scale up |
|  | |
| void CTransWorker::adjustVideoOutParam(CVideoEncoder\* pVideoEnc, int overallBr)  {  int inWidth = pvInfo->res\_in.width;  if(pPref->GetBoolean("videofilter.scale.scaleDown")) { // Only scale down  fraction\_t inDar = pvInfo->src\_dar;  if(inDar.den > 0) {  int inCalcWidth = (int)(pvInfo->res\_in.height\*inDar.num/inDar.den);  if(inCalcWidth > inWidth) inWidth = inCalcWidth;  }  if(inWidth > 0 && pvInfo->res\_out.width > inWidth) {  // Judge if src w/h is equal to dest dar (if not square pixel, logo will distort)  float destDar = -1.f;  if(dar.den > 0) {  destDar = dar.num/(float)dar.den;  pvInfo->res\_out.width = inWidth;  pvInfo->res\_out.height = (int)(pvInfo->res\_out.width/destDar);  } else {  pvInfo->res\_out.width = inWidth;  pvInfo->res\_out.height = pvInfo->res\_in.height;  }    EnsureMultipleOfDivisor(pvInfo->res\_out.width, dividor);  EnsureMultipleOfDivisor(pvInfo->res\_out.height, dividor);  }  }  } | |

#### brdown

注意和lowerbitrate的区别

brdown是限定目标码率不高于源码率

bitrate/presetLevel本身也会对目标码率有所限制

lowerbitrate是事编码输出低于目标码率

|  |  |
| --- | --- |
| default | <node key=\"videofilter.extra.brdown\">false</node>  config->target.vcodec.brdown = 1 <-- would overwrite prefsTemplate |
| User | <codec brdown=”true”/”false”/> |
| config | config->target.vcodec.brdown |
| Internal | "videofilter.extra.brdown" |
| Class |  |
| trans |  |
|  | |
| void CTransWorker::adjustVideoOutParam(CVideoEncoder\* pVideoEnc, int overallBr)  {  ...  if(pvInfo && pPref) {  // Adjust bitrate according to original bitrate  int brSource = (int)(m\_srcVideoAttrib->bitrate/1000.f + 0.5f);  if(m\_srcVideoAttrib && pPref->GetBoolean("videofilter.extra.brdown")  && overallBr > 0)  {  int brSetting = pPref->GetInt("overall.video.bitrate");  if(brSource > 0 && brSource < brSetting && brSource > overallBr\*0.6f) {  pPref->SetInt("overall.video.bitrate", brSource);  }  }  ...  // If original video highest level is HD/SD, and source bitrate is low  int presetLevel = pPref->GetInt("overall.task.ppLevel");  if(presetLevel == PPTV\_LEVEL\_720) {  if(inWidth >= 600 && inWidth < 1000) { // Highest level is HD  if(brSource > 0 && brSource < 590) {  pPref->SetInt("overall.video.bitrate", brSource);  pPref->SetInt("videoenc.x264.targetQp", 0);  }  }  } else if(presetLevel == PPTV\_LEVEL\_360) {  if(inWidth < 600) { // Highest level is SD  if(brSource > 0 && brSource < 300) {  pPref->SetInt("overall.video.bitrate", brSource);  pPref->SetInt("videoenc.x264.targetQp", 0);  }  }  }  }  } | |

#### encode

在“false”情况下只对视频解码，跳过视频的编码，但watermark,imagetail,thumbnail功能不会受到影响。

|  |  |
| --- | --- |
| default | config->target.vcodec.encoding = 1 |
| User | <codec encode="true"/>  <!--encode:true(do video encoding), false(not do video encoding)--> |
| config | config->target.vcodec.encoding |
| Internal | overall.video.encode |
| Class | m\_bEnableVideoEncode |
| trans |  |
|  | |
| THREAD\_RET\_T CTransWorkerSeperate::transcodeSingleVideoComplex() {}  THREAD\_RET\_T CTransWorkerSeperate::transcodeSingleVideo() {} | |

### profile

|  |  |
| --- | --- |
| default | <node key=\"videoenc.x264.profile\">3</node> |
| User | <profile>high</profile> <!--H.264:auto/baseline/main/high | HEVC:main--> |
| config | config->target.vcodec.profile |
| Internal | “videoenc.x264.profile” |

### lowerbitrate (x264 rc extension)

在rcmode为2pass或ABR情况下启用

通过设置targetQp起作用，具体如何参见见 x264\_mod:init\_pass2()

会使码率低于目标码率

注意和brdown的区别：brdown是限定目标码率不高于源码率

|  |  |
| --- | --- |
| default | config->target.vcodec.lower\_bitrate = 1 |
| User | <lowerbitrate>1</lowerbitrate>  <!--0:disable decreasing bitrate, 1:enable(default)--> |
| config | config->target.vcodec.lower\_bitrate |
| Internal | // Use MinQp to constrain peak bitrate  if((conf.target.vcodec.rcmode == 0 || conf.target.vcodec.rcmode == 4) &&  conf.target.vcodec.lower\_bitrate == 1) { // enable lower bitrate  prefs.SetStreamPref("videoenc.x264.savePsnr", true, STVIDEO);  ...  prefs.SetStreamPref("videoenc.x264.targetQp", 28, STVIDEO);  prefs.SetStreamPref("videoenc.x264.removeQpLog", true, STVIDEO);  ...  } |
| Class | CX264Encode:: m\_x264Param.psz\_psnr\_file |
| trans |  |
|  | |
|  | |

### enhance （transnode/videoEnhancer.cpp）

|  |  |
| --- | --- |
| default | config->target.vcodec.video\_enhance = 1 |
| User | <videoenhance>1</videoenhance>  <!--0:disable video enhance, 1:enable(default)--> |
| config | config->target.vcodec.video\_enhance |
| Internal | if(conf.target.vcodec.video\_enhance == 1) {  // Intelligence enhance  prefs.SetStreamPref("videofilter.eq.mode", 1, STVIDEO);  prefs.SetStreamPref("videofilter.eq.colorlevel", 1.f, STVIDEO);  prefs.SetStreamPref("videofilter.eq.sharpness", 1.05f, STVIDEO);  prefs.SetStreamPref("videofilter.eq.contrastThreshold", 230, STVIDEO);  prefs.SetStreamPref("videofilter.eq.contrastLevel", 0.05f, STVIDEO);  } |
| Class | m\_pVideoEnhancer |
| trans |  |
|  | |
| bool CVideoEncoder::InitWaterMark()  {  ...  if(m\_pXmlPrefs->GetInt("videofilter.eq.mode") == 1) { //Intelligence enhance  float colorLevel = m\_pXmlPrefs->GetFloat("videofilter.eq.colorlevel");  float sharpness = m\_pXmlPrefs->GetFloat("videofilter.eq.sharpness");  int contrastThreshold = m\_pXmlPrefs->GetInt("videofilter.eq.contrastThreshold");  float contrastLevel = m\_pXmlPrefs->GetFloat("videofilter.eq.contrastLevel");  m\_pVideoEnhancer = new CVideoEnhancer;  if(m\_pVideoEnhancer) {  m\_pVideoEnhancer->init(m\_vInfo.res\_out.width, m\_vInfo.res\_out.height,  m\_vInfo.res\_out.width, 0);  m\_pVideoEnhancer->setColorLevel(colorLevel);  m\_pVideoEnhancer->setSharpenLevel(sharpness);  m\_pVideoEnhancer->setContrastThreshlod(contrastThreshold);  m\_pVideoEnhancer->setContrastLevel(contrastLevel);  }  #endif  }  ...  } | |
| uint8\_t\* CVideoEncoder::FilterFrame(uint8\_t\* pOrigBuf, int origBufSize)  {  if(m\_pVideoEnhancer) {  m\_pVideoEnhancer->process(processedData);  }  } | |

### rcmode (default 2pss, ABR)

"overall.video.mode" = conf.target.vcodec.rcmode - 1

默认是2pass,ABR

|  |  |
| --- | --- |
| default | <node key="overall.video.mode">3</node>\n\ |
| User | <rcmode>4</rcmode>  <!--1:ABR, 2:CRF, 3:CBR, 4:2-pass--> |
| enum | typedef enum {  RC\_MODE\_ABR, RC\_MODE\_VBR, RC\_MODE\_CBR, RC\_MODE\_2PASS, RC\_MODE\_3PASS  } rate\_ctrl\_mode; |
| Config | config->target.vcodec.rcmode |
| Internal | if(conf.target.vcodec.rcmode >= 1) {  prefs.SetStreamPref("overall.video.mode", conf.target.vcodec.rcmode-1, STVIDEO);  } |
| Class | int rateCtrlMode = prefs->GetInt("overall.video.mode");  if(rateCtrlMode >= RC\_MODE\_2PASS && !m\_bInsertBlankVideo) {  m\_encoderPass = 2 /\*rateCtrlMode -1\*/;  } |
|  | |
|  | |

#### x264 Initialize

|  |
| --- |
| bool CX264Encode::Initialize()  {  } |

#### x264 mbtree

// Set multiple pass

if(m\_encoderPass > 1 && !pVideoEnc->GetIsMultiPass()) {

pVideoEnc->SetEncodePass(m\_encoderPass);

pVideoEnc->SetPassLogFile(m\_streamFiles.GetEncoderStatFile(i));

}

std::string CFileQueue::GetEncoderStatFile(int videoStreamId)

{

\_\_int64 curTime = GetTickCount();

std::stringstream tmpStr;

tmpStr<<m\_tempDir<<"x264stat\_"<<m\_curProcessId<<"\_"<<curTime<<"\_"<<m\_workerId<<"\_"<<videoStreamId<<".log";

std::string retStr = tmpStr.str();

m\_encoderStatFiles.push\_back(retStr);

m\_encoderStatFiles.push\_back(retStr+".mbtree");

return retStr;

}

$ ls -1 transcli/temp/

x264stat\_7868\_1445627485076\_1\_0.log

x264stat\_7868\_1445627485076\_1\_0.log.mbtree

### fps (ffmpeg -vf fps)

|  |  |
| --- | --- |
| default | <node key=\"videofilter.frame.fps\">25</node>\n\  <node key=\"videofilter.frame.fpsScale\">1</node>\n\ |
| User | <fps> fpsIndex </fps>  switch(fpsIndex) {  case 1: fpsNum = 8; fpsDen = 1; break;  case 2: fpsNum = 12; fpsDen = 1; break;  case 3: fpsNum = 15; fpsDen = 1; break;  case 4: fpsNum = 18; fpsDen = 1; break;  case 5: fpsNum = 20; fpsDen = 1; break;  case 6: fpsNum = 24000; fpsDen = 1001; break;  case 7: fpsNum = 24; fpsDen = 1; break;  case 8: fpsNum = 25; fpsDen = 1; break;  case 9: fpsNum = 30000; fpsDen = 1001; break;  case 10: fpsNum = 30; fpsDen = 1; break;  case 11: fpsNum = 5; fpsDen = 1; break;  case 12: fpsNum = 10; fpsDen = 1; break;  case 13: fpsNum = 6; fpsDen = 1; break;  case 14: fpsNum = 7; fpsDen = 1; break;  } |
| config | conf.target.vcodec.fpsNum  conf.target.vcodec.fpsDen |
| Internal | "videofilter.frame.enabled"  "videofilter.frame.fps", conf.target.vcodec.fpsNum  "videofilter.frame.fpsScale", conf.target.vcodec.fpsDen |
| Class |  |
| trans |  |
|  | |
|  | |

### dar

目标宽高比

|  |  |
| --- | --- |
| default |  |
| User | <dar>0.0</dar>  <!--Float value, Should be parsed to darNum and darDen--> |
| config | config->target.vcodec.darNum = darNum;  config->target.vcodec.darDen = darDen; |
| Internal | if(conf.target.vcodec.darNum >= 0) {  prefs.SetStreamPref("overall.video.ar" = 2, STVIDEO);  prefs.SetStreamPref("overall.video.arNum", conf.target.vcodec.darNum, STVIDEO);  prefs.SetStreamPref("overall.video.arDen", conf.target.vcodec.darDen, STVIDEO);  }  if(aspectType == 2) {  outDar.num = prefs->GetInt("overall.video.arNum");  outDar.den = prefs->GetInt("overall.video.arDen");  } else if(aspectType == 3) {  outPar.num = prefs->GetInt("overall.video.arNum");  outPar.den = prefs->GetInt("overall.video.arDen");  }  if(outSize.width > 0 && outSize.height > 0 && outDar.num <= 0 && outPar.num > 0) {  CAspectRatio aspectConvert(outSize.width, outSize.height);  aspectConvert.SetPAR(outPar.num, outPar.den);  aspectConvert.GetDAR(&(outDar.num), &(outDar.den));  } |
| Class | typedef struct {  fraction\_t dest\_par;  fraction\_t src\_dar;  fraction\_t dest\_dar;  } video\_info\_t; |
| trans |  |
|  | |
|  | |

### ref

|  |  |
| --- | --- |
| default | <node key=\"videoenc.x264.frameref\">3</node> |
| User | <ref>4</ref>  <!--Reframes, default is 4--> |
| config | config->target.vcodec.reframes |
| Internal | "videoenc.x264.frameref"  "videoenc.x265.reframes" |

### keyint

|  |  |
| --- | --- |
| default | <node key=\"videoenc.x264.keyint\">0</node> |
| User | <keyint>5</keyint>  <!--Key frame interval. If keyint < 10, it's senconds else it's frame num.--> |
| config | config->target.vcodec.keyint |
| Internal | "videoenc.x264.keyint" |

### bframes

|  |  |
| --- | --- |
| default | <node key=\"videoenc.x264.b\_pyramid\">2</node> |
| User | <bframes>4</bframes>  <!--B frames num. Range:[0,16], default is 4.--> |
| config | config->target.vcodec.bframes |
| Internal | "videoenc.x264.bframes"  "videoenc.x265.bframes" |

## delogo (ffmpeg -vf delogo + time option)

|  |  |
| --- | --- |
| default |  |
| User | <delogo>  <pos position="{135,50,223,120}" start="00:00:00.0" end="23:00:00" />  <!--Absolute coordinate-->  <pos position="{0,0.126,1,0.92}" start="00:00:00.0" end="23:00:00" /> <!--Relative coordinate-->  </delogo> |
| config | config->source.delogo\_pos[delogoIdx].rect  config->source.delogo\_pos[delogoIdx].startTime  config->source.delogo\_pos[delogoIdx].endTime |
| Internal | "videofilter.delogo.enabled"  "videofilter.delogo.num"  “videofilter.delogo.pos%d" |
| Class |  |
| trans |  |
|  | |
| // Add multiple delogo filter (with time and position)  if (m\_pVideoPref->GetBoolean("videofilter.delogo.enabled")) {  int delogoThick = m\_pVideoPref->GetInt("videofilter.delogo.thickness");  if(delogoThick <= 0) delogoThick = 2;  int delogoNum = m\_pVideoPref->GetInt("videofilter.delogo.num");  if(delogoNum > 0) {  for (int i=0; i<delogoNum; ++i) {  char posKey[50] = {0};  sprintf(posKey, "videofilter.delogo.pos%d", i+1);  const char\* timePosStr = m\_pVideoPref->GetString(posKey);  std::vector<int> vctVal;  float delogoStart= 0.f, delogoEnd = 0.f;  int delogoX = 0, delogoY = 0, delogoW = 0, delogoH = 0;  if(StrPro::StrHelper::parseStringToNumArray(vctVal, timePosStr)) {  if(vctVal.size() != 6) continue;  delogoStart = vctVal.at(0)/1000.f; // ms to sec  delogoEnd = vctVal.at(1)/1000.f; // ms to sec  delogoX = vctVal.at(2);  delogoY = vctVal.at(3);  delogoW = vctVal.at(4);  delogoH = vctVal.at(5);  }  if(delogoStart > -0.1f && delogoEnd > -0.1f && delogoStart>delogoEnd) continue;  forePart << "delogo=" << delogoX << ':'  << delogoY << ':' << delogoW << ':'  << delogoH << ':' << delogoThick << ":0:" // 0 is value of 'show'  << delogoStart << ':' << delogoEnd << ',';  }  } else { // MediacoderNT delogo  int delogoX = m\_pVideoPref->GetInt("videofilter.delogo.x");  int delogoY = m\_pVideoPref->GetInt("videofilter.delogo.y");  int delogoW = m\_pVideoPref->GetInt("videofilter.delogo.w");  int delogoH = m\_pVideoPref->GetInt("videofilter.delogo.h");  int delogoThick = m\_pVideoPref->GetInt("videofilter.delogo.thickness");  forePart << "delogo=" << delogoX << ':'  << delogoY << ':' << delogoW << ':'  << delogoH << ':' << delogoThick << ":0:" // 0 is value of 'show'  << "-1:-1,";  }  } | |

## logo (watermark/WaterMarkFilter.cpp)

InitWaterMark()->parseWaterMarkInfo()

* file支持多logo交替出现，格式如下：

file=”{logo1\_path},{logo2\_path},...”

* interval(ms)：logo交替的间隙（一个logo显示多久然后换到下一个）
* logo的移动可以通过一条<relativepos>或者多条<pos>来实现。
* <relativepos>和<pos>同时出现时优先选用<relativepos>
* <relativepos>
* speed: type="int" text="Logo moving speed(pixels/s)"
* opaque: 不透明度 (Range: 0~1)
* route: Logo 移动路径.
  + - 'x':横向移动
    - 'y':垂直移动
    - 'c':对角线移动

这3中route可以任意组合，用逗号’,’分开

* waitTime: logo移动到一条路劲的尽头时等待一定时间。

|  |  |
| --- | --- |
| default |  |
| User | <logo>  <graph file="{D:\logoTest\community\_1中6.png}" interval="1000ms" />  <show>  <relativepos corner="2" offsetx="10" offsety="10" width="0" height="0"  speed="0" opaque="0.0" wait="0" route="x,c"/>  <!-- corner (1:LU 2:RU 3:RD 4:LD), exclusive with absolute position-->  <pos position="{5,5,60,60}" start="00:00:00" end="00:00:20" />  <!--Absolute position-->  </show>  </logo> |
| config | config->target.logo.graph\_files[index++]  config->target.logo.graph\_interval |
| Internal | “videofilter.overlay.enabled” = (config->target.logo.logo\_pos\_num > 0) |

### relativepos

|  |  |
| --- | --- |
| default |  |
| User | <relativepos corner="2" offsetx="10" offsety="10" width="0" height="0"  speed="0" opaque="0.0" wait="0" route="x,c"/> |
| config | config->target.logo.logo\_pos\_num = 1;  config->target.logo.relativePos = corener  config->target.logo.offsetx  config->target.logo.offsety  config->target.logo.width  config->target.logo.height  config->target.logo.waitTime  config->target.logo.speed  config->target.logo.opaque = opaque  config->target.logo.routes |
| Internal | "videofilter.overlay.pos", logoConfig.relativePos  "videofilter.overlay.offsetX"  "videofilter.overlay.offsetY"  "videofilter.overlay.speed"  "videofilter.overlay.waittime"  "videofilter.overlay.route"  "videofilter.overlay.opaque"  "videofilter.overlay.colored" |
| Class |  |
| trans | CWaterMarkFilter::AddShowTimeAndPosition() |
|  | |
|  | |

### pos

|  |  |
| --- | --- |
| default |  |
| User | <pos position="{5,5,60,60}" start="00:00:00" end="00:00:20" /> |
| config | config->target.logo.logo\_pos\_num  config->target.logo.log\_pos[delogoIdx].rect  config->target.logo.log\_pos[delogoIdx].startTime  config->target.logo.log\_pos[delogoIdx].endTime |
| Internal | videofilter.overlay.pos%d |

## filter

### denoise (ffmpeg -vf denoise)

denoise.temporal is not used in ffmpeg (for AVS or Mencoder)

|  |  |
| --- | --- |
| default |  |
| User | <filter>  <denoise mode="0" luma="4" chroma="3" temporal="6"/>  </filter> |
| config | config->target.filter.denoise.denoiseMode = xmlConfig.getAttributeInt("mode");  config->target.filter.denoise.lumaVal = xmlConfig.getAttributeInt("luma");  config->target.filter.denoise.chromaVal = xmlConfig.getAttributeInt("chroma");  config->target.filter.denoise.temporalVal = xmlConfig.getAttributeInt("temporal"); |
| Internal | "videofilter.deint.mode"  "videofilter.deint.algorithm" |

### deinterlace (ffmpeg -vf yadiff)

|  |  |
| --- | --- |
| default | <node key=\"videofilter.deint.mode\">2</node>\n\  <node key=\"videofilter.deint.algorithm\">YADIF</node>\n\ |
| User | <filter>  <deinterlace mode="0" algorithm="0"/>  <!--mode:0(auto), 1(enable), 2(disable)-->  </filter> |

如果编码器使用了interlace模式(“overall.task.interlace” = 0), 不使用deinterlace模式。

ffmpeg里有两个deinterlace的vfilter：tinterlace/yadif,现在默认使用yadif。

#### deintMode

|  |  |
| --- | --- |
| default | <node key=\"videofilter.deint.mode\">2</node>\n\ |
| User | <filter>  <deinterlace mode="0" algorithm="0"/>  <!--mode:0(auto), 1(enable), 2(disable)-->  </filter> |
| config | config->target.filter.deint.deintMode  enum deint\_mode\_t {  DEINT\_AUTO = 0,  DEINT\_ENABLE = 1,  DEINT\_DISABLE = 2,  }; |
| Internal | "videofilter.deint.mode": 0(disable), 1(enable/force), 2(auto/enable when field)  int deintMode = 2;  switch(conf.target.filter.deint.deintMode) {  case DEINT\_DISABLE: deintMode = 0; break;  case DEINT\_ENABLE: deintMode = 1; break;  }  prefs.SetStreamPref("videofilter.deint.mode", deintMode, STVIDEO); |

#### deintAlgorithm

|  |  |
| --- | --- |
| default | <node key=\"videofilter.deint.algorithm\">YADIF</node>\n\ |
| User | <filter>  <deinterlace mode="0" algorithm="0"/>  </filter> |
| config | config->target.filter.deint.deintAlgorithm = xmlConfig.getAttributeInt("algorithm"); |
| API  Internal | int deintAlgo = 7; //YADIF  switch(conf.target.filter.deint.deintAlgorithm) {  case DEINT\_BLEND: deintAlgo = 0; break;  case DEINT\_LINEAR: deintAlgo = 1; break;  }  prefs.SetStreamPref("videofilter.deint.algorithm", deintAlgo, STVIDEO); |
| NON-API  Internal | "overall.task.interlace"  "videosrc.ffmpeg.tinterlace"  "videofilter.deint.fieldOrder"  “videofilter.deint.yadif”  “videofilter.deint.algorithm” |

## Mux

|  |  |
| --- | --- |
| default |  |
| User | <mux format="flv" disable="false"/>  <!--Muxer format:flv/mp4/ts/hls disable:enable or disable muxing--> |

## segment

|  |  |
| --- | --- |
| default |  |
| User | <segment type="average" unit="time" size="40" postfix="index"  threshold="1800" lastsegpercent="20"/>  <!--type:none(0)/average(1)/ normal(2)-->  <!--unit:size(0)/time(1) -->  <!--size:segment size(sec)-->  <!--postfix:index/time/.mp4.%d-->  <!--lastsegpercent:20%-->  <!--threshold:duration>=1800 segment--> |
| config | struct segment\_t {  int type; // 0/1/2(none/average/normal)  int unit; // 0/1 (size/time)  int size; // segment size (secs or kbytes)  char postfix[MAX\_PATH];  int threshold; // segment if duration is bigger than threshold(secs)  int lastSegPercent; // If last segment is bigger than percent% of size, then split.  };  config->target.segmentConfig |
| Internal | "extension.split.type"  "extension.split.unitType"  "extension.split.subfix", conf.target.segmentConfig. postfix  "extension.split.segSize"  "extension.split.lastSegPercent"  "extension.split.threshold" |
| Class |  |
| trans | bool CTransWorkerSeperate::parseSegmentConfig(CXMLPref\* prefs) |
|  | |
| bool CTransWorkerSeperate::parseSegmentConfig(CXMLPref\* prefs)  {  ...  if(segmentType && (\*segmentType) && strcmp(segmentType, "none") != 0 && needSegment) {  m\_pSplitter = new CMediaSplitter();  ...  m\_pSplitter->Init();  }  return true;  } | |

## hls

|  |  |
| --- | --- |
| default |  |
| User | <hls dur="5" listsize="100000" postfix=".mp4" startindex="1"/>  <!--Apple HLS format. -->  <!--dur:segment duration(s), -->  <!--listsize:how many segments one list contain, -->  <!--startindex:segment file start number --> |
| config | struct hls\_t {  int dur; //duration  int listSize; // item count in list  int startIndex; // segment start index(start file number), default is 0  char postfix[MAX\_PATH];  }; |

## playlist

|  |  |
| --- | --- |
| default |  |
| User | <playlist type=".m3u8" name="" postfix=".mp4" live="false"/>  <!--type:.m3u8/.m3u/.pls/.wpl/.csv-->  <!--name:playlist name(when empty use source file name)-->  <!--postfix:name postfix, live:if live stream--> |
| config | struct playlist\_t {  char type[BUF\_LEN];  char name[MAX\_PATH];  char postfix[MAX\_PATH];  bool bLive;  }; |

## subtitle (ffmpeg -vf overlay/ass/subtitle)

根据字幕种类的不同：

* iamge subtitle使用ffmpeg -vf overlay
* text subtitle使用ffmpeg -vf ass/subtitle
  + GenTextSubOptions()
  + 如果是视频置的text字幕，要先ExtractTextSub()到字幕文件

|  |  |
| --- | --- |
| default | config->target.extract\_sub\_id = -2; // default:no extract sub |
| User | <subtitle id="-1" timeshift="0s" extract="-2"/>  <!--id:-1 auto select embed subtitle, -2 disable embed subtitle-->  <!--extract:-2 don't extract, -1 extract all, >=0:extract specific id subtitle--> |
| config | config->target.subtitle\_timeshift  config->target.sub\_id  config->target.extract\_sub\_id |
| Internal | "overall.subtitle.sid" = config->target.sub\_id  "overall.subtitle.delay" = conf.target.subtitle\_timeshift  "overall.subtitle.extract" = conf.target.extract\_sub\_id  "overall.subtitle.mode" = ?  "overall.subtitle.embedType" = “pgssub/dbdsub/subrip/ssa” (`ffprobe result`)  "overall.subtitle.fuzziness"  "videofilter.overlay.text" |

### subOption/subType

|  |
| --- |
| // 0:disable, 1:external text sub(srt/ass), 2:internal bitmap sub(PGS/DVDSub) |
| GetCmdString {  int subOption = 0;  int subTitleMode = m\_pVideoPref->GetInt("overall.subtitle.mode");  if(subTitleMode > 0) {  std::string externSub = GetSubFile(mediaFile);  if(!externSub.empty()) {  subOption = 1;  } else {  int subIndex = m\_pVideoPref->GetInt("overall.subtitle.sid");  const char\* embedSubType = m\_pVideoPref->GetString("overall.subtitle.embedType");  if(subIndex >= 0 && embedSubType) {  if(!\_stricmp(embedSubType, "pgssub") || !\_stricmp(embedSubType, "dvdsub")) {  subOption = 2;  } else if(!\_stricmp(embedSubType, "subrip") || !\_stricmp(embedSubType, "ssa")) {  // Extract embed text subtitle to external subtitle  ExtractTextSub(mediaFile, subIndex, externSub, embedSubType);  subOption = 1;  }  }  }  }    std::string videoFilterStr = GenVideoFilterOptions(subOption);    if(!videoFilterStr.empty()){  if(subOption == 1) { // External text sub  videoFilterStr += GenTextSubOptions(mediaFile, externSub);  }  if(subOption == 2) {  cmd << " -filter\_complex \"" << videoFilterStr << "\""; // -fix\_sub\_duration";  } else {  cmd << " -filter:v " << videoFilterStr;  }  }  } |

### adjustSubtitleAttrib()

|  |
| --- |
| GetMediaInfo((((--------  [  probeDoc = GetMediaInfoXML( "ffprobe -i strPreset.ifile" )  m\_pMainMediaInfo <== parseSubtitleInfo( probeDoc )  ]  --------)))) |
| CTransWorkerSeperate::ParseSetting((((--------  [  adjustEncodeSetting((((--------  adjustSubtitleAttrib(mediaInfo, pVideoEnc->GetVideoPref());  --------))))    extractSubtitle((((--------  CProcessWrapper::Run( "ffmpeg -i $SRC -an -vn ..." );  --------))))  ]  --------)))) |

### mode

|  |  |
| --- | --- |
| mcnt.xml | <node key="subtitle" type="node" text="Subtitle">  <node key="mode" type="enum" text="Mode">  <enum>Disabled</enum>  <enum sel="1">Render</enum>  <enum>Copy</enum>  </node> |

### fuzziness

|  |  |
| --- | --- |
| mcnt.xml | <node key="subtitle" type="node" text="Subtitle">  <node key="fuzziness" type="enum" text="Subtitle Fuzziness">  <desc>Adjust matching fuzziness when searching for subtitles</desc>  <enum>Exact match</enum>  <enum sel="1">Load all subs containing movie name</enum>  <enum>Load all subs in the current directory</enum>  </node> |

### videofilter.overlay.text

|  |  |
| --- | --- |
| User | <node key="overlay" type="node" text="Image overlay / Watermark">  <node key="text" type="file" text="Subtitle file used as watermark"/> |
| std::string CDecoder::GetSubFile(const char\* srcFile)  {  const char\* overlaySubFile = m\_pVideoPref->GetString("videofilter.overlay.text");  if (overlaySubFile && FileExist(overlaySubFile)) {  return overlaySubFile;  }  } | |

## thumbnail (watermark/Thumbnailfilter.cpp)

最多允许设置两个<thumbnail>

start： 开始时间 (float/sec)

end： 结束时间 (float/sec)

type： 1(png)/2(jpg)/3(bmp)/4(gif)

interval/count： 如果interval>0, count=(start-end)/interval

否则interval =(start-end)/ count

stitching： 将thumbnail拼成一幅大的图像

pack： 将thumbnail打包成ipk

同时打开pack和sticching时pack优先

|  |  |
| --- | --- |
| default |  |
| User | <thumbnail start="6" end="0" interval="6"  row="10" col="10" width="180" height="0"  type="2" count="0" quality="85"  stitch="true" pack="true" postfix="\_shot"/>  <!--start:start time, end:end time, interval:frequency-->  <!--row:tile row, col:tile column-->  <!--type:1(png)/2(jpg)/3(bmp)/4(gif)-->  <!--count:thumbnail count-->  <!--quality:Jpeg quality(max:100)-->  <!--stitch:tile thumbnail to big image-->  <!--pack:pack images to ipk file-->  <!--postfix:postfix of thumbnail's file name--> |
| config | config->target.thumbnails[0]  config->target.thumbnails[1] |
| Internal | "videofilter.thumb.enabled"  "videofilter.thumb1.enabled" |
| Class | CVideoEncoder::m\_pThumbnail  CVideoEncoder::m\_pThumbnail1 |
| trans | InitWaterMark()->parseThumbnailInfo()  FilterFrame()->GenerateThumbnail() |

### ipk格式及打包

bool CThumbnailFilter::StopThumbnail();

## imagetail (transnode/TransWorkerSeperate.cpp)

将图片作为原始帧数据，编码到视频末尾，分static,dynamic两种方式，static优先

* static：将一张图片（image）作为额外的视频源编码到视频末尾，并重复duration秒
  + image: 静态方式下 的图片路径
  + druation: 静态方式下 的图片重复时间
* dynamic：将folder目录下的所有图片作为额外的视频源编码到视频末尾
  + folder: 动态方式下存储图片的路径
* cropmode:当图片宽高比和视频宽高比不同时，用crop/expand适配视频，然后再scale

|  |  |
| --- | --- |
| default |  |
| User | <imagetail image="" folder="" duration="2" cropmode="0"/>  <!--将图片作为原始帧数据，编码到视频末尾，分static,dynamic两种方式，static优先-->  <!--static：将一张图片（image）作为额外的视频源编码到视频末尾，并重复duration秒-->  <!--dynamic：将folder目录下的所有图片作为额外的视频源编码到视频末尾-->  <!--image: 静态方式下 的图片路径-->  <!--druation: 静态方式下 的图片重复时间-->  <!--folder: 动态方式下存储图片的路径-->  <!--cropmode: 0:crop,1:expand--> |
| config | config->target.imageTail.imagePath  config->target.imageTail.duration  config->target.imageTail.imageFolder  config->target.imageTail.cropMode |
| Internal | "videofilter.imagetail.imagepath"  "videofilter.imagetail.duration"  "videofilter.imagetail.imagefolder"  "videofilter.imagetail.cropmode" |
| Class | m\_pImgTail = parseImageTailConfig(CXMLPref\* prefs) |
| trans | CTransWorkerSeperate::addImageTailToVideoStream(CVideoEncoder\* pEncoder);  m\_pImgTail->GetNextFrame();  bool CImageSrc::loadAndConvertImage(const char\* imagePath); |

### cropmode (watermark/ImageSrc.cpp)

cropmode:当图片宽高比和视频宽高比不同时，用crop/expand适配视频，然后再scale：

|  |
| --- |
| bool CImageSrc::loadAndConvertImage(const char\* imagePath)  {  try {  ...  int x0 = 0, y0 = 0, x1 = 0, y1 = 0;  if(m\_yuvDarDen > 0 && m\_yuvDarNum > 0) {  getCropParam(x0, y0, x1, y1, img.width(), img.height());  }  if(x0 != 0 || y0 != 0) { // Crop or Expand  img.crop(x0, y0, x1, y1, true);  }  img.resize(m\_yuvW, m\_yuvH, -100, -100, 5);  RGB2YV12(m\_yuvFrame, img);  } catch (cimg\_library::CImgIOException& e) {  logger\_err(LOGM\_TS\_VE, "%s!\n", e.what());  return false;  }  return true;  } |

## avalign (transnode/TransWorkerSeperate.cpp)

blankaudio:源片没有音频时插入空白音轨

blankvideo:源片没有视频时插入空白视频

padding: 音视频长度不一致时在音频末补音频或在视频末补视频。如果音视频长度差别太大需要设置errorignore为2.

|  |  |
| --- | --- |
| default | <node key=\"overall.task.alignAVData\">true</node>\n\  <node key=\"overall.audio.insertBlank\">true</node>\n\  // video.insertBlank is default to be faulse |
| User | <avalign blankaudio="true" blankvideo="true" padding="true"/>  <!---Audio/video data(time) alignment.-->  <!---blankaudio:true/false, whehter insert blank audio track when there are no audio tracks in source file).-->  <!---blankvideo:true/false, whether insert blank video track when there are no video tracks in source file).-->  <!---padding: true/false, whether padding audio with silence or padding video with black frame to align a/v time)--> |
| config | config->target.disable\_insert\_blank\_audio  config->target.enable\_insert\_blank\_video  config->target.disable\_padding\_avdata |
| Internal | "overall.audio.insertBlank”  "overall.video.insertBlank"  "overall.task.alignAVData" = !config->target.disable\_padding\_avdata |
| Class | m\_bInsertBlankAudio  m\_bInsertBlankVideo  m\_bEnableAlignAVData |
| trans | bool CTransWorkerSeperate::appendBlankAudio(CAudioEncoder\* pEncoder)  bool CTransWorkerSeperate::appendBlankVideo(CVideoEncoder\* pEncoder) |
|  | |
|  | |

## errorignore

|  |  |
| --- | --- |
| User | <errorignore>0</errorignore>  <!--ignoreErrIdx: 0(no ignore)-->  <!--1(仅忽略解码器异常退出：exit\_code=32)-->  <!--2(仅忽略音视频时长差值大于20s：exit\_code=33)-->  <!--3(ignore both)--> |
| config | config->target.ignoreErrIdx |
| Internal | "overall.task.ignoreError" |
| Class | m\_ignoreErrIdx |
| trans | bool CTransWorker::**validateTranscode**(int decoderExitCode) |
|  | |
| CTransWorkerSeperate::firstPassAnalyse()  {  int decoderExitCode = closeDecoders();  deleteDecoders();  // Validate if transcode is complete  if (!m\_bInsertBlankAudio && !m\_bInsertBlankVideo &&  !validateTranscode(decoderExitCode)) {  break;  }  }  CTransWorkerSeperate::mainFunc()  {  int decoderExitCode = closeDecoders();  // Validate transcoding, check if error exists  if(!validateTranscode(decoderExitCode)) {  ret = -1; break;  }  } | |

### exitcode

int CProcessWrapper::Wait(int timeout)

bool CProcessWrapper::IsProcessRunning(int\* pexitcode)

## ignorecode

|  |  |
| --- | --- |
| User | <ignorecode></ignorecode>  <!--Ignore error code--> |
| config | config->target.ignoreErrCodeStr |
| Internal | "overall.task.ignorecode" |
| Class | m\_ignoreErrCodes |
| trans | bool CTransWorker::errIgnored(int errCode)  {  for (size\_t i=0; i<m\_ignoreErrCodes.size(); ++i) {  if(m\_ignoreErrCodes[i] == errCode) return true;  }  return false;  } |
|  | errIgnored(EC\_NO\_VIDEO\_TRACK) |
|  | errIgnored(EC\_DUR\_LESS\_THAN\_ONE\_SEC) |
|  | |
|  | |

### error\_code

|  |
| --- |
| typedef enum {  EC\_NO\_ERROR = 0, // no error  EC\_NOT\_INITED = 1, // not initialized  EC\_INIT\_ERROR = 2, // initialization error  EC\_START\_ERROR = 3, // worker start error  EC\_CREATE\_THREAD\_ERROR = 4, // thread create failed  EC\_DECODER\_ERROR = 5, // decoder error  EC\_AUDIO\_INIT\_ERROR = 6, // audio init erro  EC\_VIDEO\_INIT\_ERROR = 7, // video init erro  EC\_INVALID\_PARAM = 8, // invalid params  EC\_INVALID\_PREFS = 9, // invalid preset loaded  EC\_INVALID\_LANG\_FILE = 10, // invalid language file loaded  EC\_DIR\_IO\_ERROR = 11, // directory I/O access error  EC\_FILE\_IO\_ERROR = 12, // file I/O access error  EC\_INVALID\_SETTINGS = 13, // invalid combination of settings  EC\_INVALID\_AUDIO\_FORMAT = 14, // invalid audio format  EC\_INVALID\_VIDEO\_FORMAT = 15, // invalid video format  EC\_INVALID\_MEDIA\_FILE = 16, // invalid media file source  EC\_AUDIO\_SOURCE\_ERROR = 17, // an audio source error occurred  EC\_AUDIO\_ENCODER\_ERROR = 18, // an audio encoder error occurred  EC\_VIDEO\_SOURCE\_ERROR = 19, // an video source error occurred  EC\_VIDEO\_ENCODER\_ERROR = 20, // an video encoder error occurred  EC\_MUXER\_ERROR = 21, // an muxer error occurred  EC\_PREF\_KEY\_NOT\_FOUND = 22, // pref key not found  EC\_TAG\_WRITE\_ERROR = 23, // tag writing error  EC\_NODE\_DOWN = 24, // a node has broken down  EC\_XMLRPC\_SOCKET\_ERROR = 25, // xmlrpc socket timeout or other error  EC\_XMLRPC\_INVALID\_PAYLOAD = 26, // xmlrpc: invalid payload  EC\_XMLRPC\_OTHER\_ERROR = 27, // other xmlrpc error, TODO: breaking down is needed  EC\_AUDIO\_ENCODER\_ABNORMAL\_EXIT = 28, // audio encoder exit unexpectedly  EC\_VIDEO\_ENCODER\_ABNORMAL\_EXIT = 29, // video encoder exit unexpectedly  EC\_INVILID\_AUDIO\_ATTRIB = 30, // Invalid audio attribute  EC\_INVILID\_VIDEO\_ATTRIB = 31, // Invalid video attribute  EC\_DECODER\_ABNORMAL\_EXIT = 32, // decoder exit unexpectedly  EC\_AV\_DURATION\_BIG\_DIFF = 33, // a/v duration differs more than 20s.  EC\_ENCODER\_NOT\_EXIT = 34, // CLI Encoder is forced to shut down.  EC\_NOT\_ENOUGH\_DISK\_SPACE = 35, // Disk space is not enough.  EC\_BAD\_AV\_DATA = 36, // Output mp4 audio/video data corrupt(No frame/audio element 0.0)  EC\_GEN\_THUMBNAIL\_ERROR = 37, // Generate ipk file failed.  EC\_INVALID\_CLIP\_PARAM = 38, // Clip param is invalid  EC\_DUR\_LESS\_THAN\_ONE\_SEC = 39, // Duration is less than 1 second  EC\_NO\_VIDEO\_TRACK = 40, // No video track  } error\_code\_t; |

## additon

允许直接设置x264,x265，...的选项，不过并不是所有选项都可以直接设置。转码内核做了一层转接。详细代码在AdjustPreset()中。

config->target.codecConfig[X264\_ADDITION\_OPTION].option

config->target.codecConfig[X265\_ADDITION\_OPTION].option

config->target.codecConfig[MP4BOX\_ADDITION\_OPTION].option

### x264 option

|  |
| --- |
| const char\* x264Extras[] = {"ref","bframes","me", "subme", "merange","b-adapt",  "b-pyramid", "aq-mode", "weightp", "weightb", "trellis",  "mbtree", "nal-hrd", "scenecut", "rc-lookahead"};  const char\* x264PresetItem[] = {"videoenc.x264.frameref", "videoenc.x264.bframes",  "videoenc.x264.me", "videoenc.x264.subme", "videoenc.x264.me\_range",  "videoenc.x264.b\_adapt", "videoenc.x264.b\_pyramid", "videoenc.x264.aq\_mode",  "videoenc.x264.weight\_p", "videoenc.x264.weight\_b", "videoenc.x264.trellis",  "videoenc.x264.mbtree", "videoenc.x264.nalhrd", "videoenc.x264.scenecut",  "videoenc.x264.rc\_lookahead" }; |

### x265 option

|  |
| --- |
| const char\* x265Extras[] = {"ref", "bframes", "me", "subme", "sao", "amp", "rect", "b-adapt", "wpp", "ctu",  "frame-threads", "rd", "lft", "b-pyramid", "cutree", "aq-mode", "weightp", "weightb", "open-gop",  "vbv-bufsize", "vbv-maxrate", "vbv-init", "psnr", "ssim", "max-merge", "rc-lookahead",  "tu-intra-depth", "tu-inter-depth", "b-intra", "psnr", "ssim"};  const char\* x265PresetItem[] = {"videoenc.x265.reframes", "videoenc.x265.bframes", "videoenc.x265.me",  "videoenc.x265.subme", "videoenc.x265.sao", "videoenc.x265.amp", "videoenc.x265.rect", "videoenc.x265.badapt",  "videoenc.x265.wpp", "videoenc.x265.ctu", "videoenc.x265.frameThreads", "videoenc.x265.rdLevel",  "videoenc.x265.loopFilter", "videoenc.x265.bpyramid", "videoenc.x265.cuTree",  "videoenc.x265.aqMode", "videoenc.x265.weightp", "videoenc.x265.weightb", "videoenc.x265.openGop",  "videoenc.x265.vbvBufferSize", "videoenc.x265.vbvMaxrate", "videoenc.x265.vbvBufferInit",  "videoenc.x265.psnr", "videoenc.x265.ssim","videoenc.x265.maxMerge", "videoenc.x265.lookahead",  "videoenc.x265.ctuIntra", "videoenc.x265.ctuInter", "videoenc.x265.bIntra", "videoenc.x265.pnsr", "videoenc.x265.ssim"}; |

### mp4box option

|  |
| --- |
| firstSegSize = 20;  commonSegSize = 20;  // Temporally set segsize = 20M, use old segment method  sprintf(cprtStr, "PPTV-"TS\_MAJOR\_VERSION"."TS\_MINOR\_VERSION"."SVN\_REVISION"(%d,%d)[1]",  firstSegSize, commonSegSize);  prefs.SetStreamPref("overall.tagging.copyright", cprtStr, STMUXER);  const char\* mp4Option = conf.target.codecConfig[MP4BOX\_ADDITION\_OPTION].option;  if(mp4Option && \*mp4Option) {  const char\* pCh = NULL;  if(pCh = strstr(mp4Option, "-hint")) { // hint  std::string hintStr = GetStringBetweenDelims(pCh, "=", ",");  if(hintStr.compare("true") == 0) {  prefs.SetStreamPref("muxer.mp4box.hint", true, STMUXER);  }  }  if(pCh = strstr(mp4Option, "-sbr")) { // sbr  std::string hintStr = GetStringBetweenDelims(pCh, "=", ",");  int sbrNum = atoi(hintStr.c\_str());  prefs.SetStreamPref("muxer.mp4box.sbr", sbrNum, STMUXER);  }  if(pCh = strstr(mp4Option, "-brand")) { // brand  std::string brandStr = GetStringBetweenDelims(pCh, "=", ",");  prefs.SetStreamPref("muxer.mp4box.brand", brandStr.c\_str(), STMUXER);  }  if(pCh = strstr(mp4Option, "-version")) { // version number  std::string verStr = GetStringBetweenDelims(pCh, "=", ",");  int verNum = atoi(verStr.c\_str());  prefs.SetStreamPref("muxer.mp4box.version", verNum, STMUXER);  }  } |

# xml template

## m\_pMainMediaInfo

### ffprobe -of xml

#### sample-1 (single audio, no sub)

|  |
| --- |
| $ ./codecs/ffprobe -analyzeduration 500000000 -probesize 500000000 -i "/home/jfzheng/workspace/151029\_dec\_fail/TestAudio\_dest.mp4" -of xml -show\_streams -show\_format -detect\_inter\_frames 4 -v quiet -show\_entries format=duration,size,bit\_rate,format\_name,probe\_score:format\_tags=encoder,metadatacreator:stream\_tags=language,rotate |
| <?xml version=**"1.0"** encoding=**"UTF-8"**?>  <ffprobe>  <streams>  <stream index=**"0"** codec\_name=**"h264"** codec\_long\_name=**"H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10"** profile=**"High"** codec\_type=**"video"** codec\_time\_base=**"1/100"** codec\_tag\_string=**"avc1"** codec\_tag=**"0x31637661"** width=**"1920"** height=**"1080"** has\_b\_frames=**"2"** sample\_aspect\_ratio=**"1:1"** display\_aspect\_ratio=**"16:9"** pix\_fmt=**"yuv420p"** level=**"42"** interlaced\_frame=**"0"** top\_field\_first=**"0"** r\_frame\_rate=**"50/1"** avg\_frame\_rate=**"50/1"** time\_base=**"1/12800"** start\_pts=**"0"** start\_time=**"0.000000"** duration\_ts=**"7303936"** duration=**"570.620000"** bit\_rate=**"12301691"** nb\_frames=**"28531"** nb\_read\_frames=**"10"**>  <disposition default=**"1"** dub=**"0"** original=**"0"** comment=**"0"** lyrics=**"0"** karaoke=**"0"** forced=**"0"** hearing\_impaired=**"0"** visual\_impaired=**"0"** clean\_effects=**"0"** attached\_pic=**"0"**/>  <tag key=**"language"** value=**"und"**/>  </stream>  <stream index=**"1"** codec\_name=**"aac"** codec\_long\_name=**"AAC (Advanced Audio Coding)"** codec\_type=**"audio"** codec\_time\_base=**"1/44100"** codec\_tag\_string=**"mp4a"** codec\_tag=**"0x6134706d"** sample\_fmt=**"fltp"** sample\_rate=**"44100"** channels=**"2"** channel\_layout=**"stereo"** bits\_per\_sample=**"0"** r\_frame\_rate=**"0/0"** avg\_frame\_rate=**"0/0"** time\_base=**"1/44100"** start\_pts=**"-2048"** start\_time=**"-0.046440"** duration\_ts=**"9845935"** duration=**"223.263832"** bit\_rate=**"128301"** nb\_frames=**"9616"** nb\_read\_frames=**"9"**>  <disposition default=**"1"** dub=**"0"** original=**"0"** comment=**"0"** lyrics=**"0"** karaoke=**"0"** forced=**"0"** hearing\_impaired=**"0"** visual\_impaired=**"0"** clean\_effects=**"0"** attached\_pic=**"0"**/>  <tag key=**"language"** value=**"und"**/>  </stream>  </streams>  <format format\_name=**"mov,mp4,m4a,3gp,3g2,mj2"** duration=**"570.620000"** size=**"881343217"** bit\_rate=**"12356289"** probe\_score=**"100"**>  <tag key=**"encoder"** value=**"Lavf55.33.100"**/>  </format>  </ffprobe> |

### GetMediaInfo()

probeDoc = GetMediaInfoXML( tempXml<input> )

parseGeneralInfo( probeDoc )

parseVideoInfo( probeDoc )

parseAudioInfo( probeDoc )

parseSubtitleInfo( probeDoc )

## overall (transcli/pplive.cpp)

prefsTemplate[]

## MEvaluater.h (common/mcnt.xml)

#define MCCORE\_XML "mcnt.xml"

MEvaluater::GetDefaultNodeValueByKey(const char\* key)