

[深信服] TSC SDK 说明文档

一. TSC SDK 文件包内容说明

进入 tscsdk-center 查看当前目录下的所有文件：

```
total 20068
-rwxr-xr-x 1 root root 20536928 Oct 20 17:11 ffmpeg
drwxr-xr-x 2 root root      4096 Aug  1 22:31 lib
-rw-r--r-- 1 root root        54 Jul 14 12:08 sdk_config
drwxr-xr-x 2 root root      4096 Oct 20 15:36 src
```

名称	内容
ffmpeg	已经 sdk 相关能力的 ffmpeg 程序
lib	sdk 核心函数库和头文件
sdk_config	sdk 用于鉴权的配置文件
src	将 sdk 集成到 ffmpeg 的 wrapper 文件，包含 ten264 和 ten265 编码器。该文件可以为开发者提供 sdk 函数接口调用的参考。
demo	sdk 接口调用示例和测试视频片段
doc	sdk 说明文档

1.1 SDK 相关接口集成方法

lib 目录下提供了 sdk 的头文件和库文件：

```
total 9852
-rwxr-xr-x 1 root root 9984792 Oct 20 17:15 libtscsdk_center.so
-rw-r--r-- 1 root root 100159 Oct 20 17:10 libtscsdk.h
```

在编程时引用其中的头文件，在编译时向编译器添加指定的编译参数即可：

引用头文件：

```
#include "libtscsdk.h"
```

添加编译参数

```
-l lib
```

```
-L lib
```

```
-l tscsdk_center
```

1.2 鉴权文件配置说明

sdk_config 鉴权文件中的内容为 json 格式，其主要结构如下：

```
{  
  "net_card": "eth0",  
  "dst_ip": ["9.151.210.104"]  
}
```

其中，net_card 值需要设置为主机的第一个网卡名称，可通过 ifconfig 命令查看；dst_ip 为鉴权服务所在地址，该字段支持在列表中同时配置多个 ip 地址。

二. 0 延迟编码参数

2.1 ffmpeg 编码参数

- ten264 编码器

```
time ./ffmpeg -vsync 0 -r 24 -s 1920x1080 -pix_fmt nv12 -i movie_1920x1080_24fps_30s.yuv -vframes 730  
-threads 4 -vcodec libten264 -preset ultrafast -ten264opts threads=4:crf=26.6:keyint=infinite:p-  
opt=1,0,4,1.5:tune=zerolatency_shxf -y movie_1920x1080_24fps_30s_crf26.6-sharp.264
```

其中，crf 的值分别设置在 21.5、26.6、31.7、36。

注意事项：

- ten264 编码器中，tune 参数需要通过 -ten264opts 进行设置，如上述命令所示，目前只支持设置为“tune=zerolatency_shxf”。

- ten265 编码器

```
time ./ffmpeg -vsync 0 -r 24 -s 1920x1080 -pix_fmt yuv420p -i movie_1920x1080_24fps_30s_420.yuv -
```

```
vframes 730 -vcodec libten265 -ten265-params tune=shortlatency:preset=9:crf=26:keyint=1000:ltr=0:max-dpb=16:aq-mode=1:level=5:gop-struct=lowdelayB:pool-threads=4 -y  
movie_1920x1080_24fps_30s_yuv420p_crf26.264
```

其中，crf 的值分别设置在 26、30、36、40。

注意事项：

- **ten265 编码器不支持 nv12 格式的输入数据，需要将其转换为 yuv420p 格式再送入编码器。**用 ffmpeg 转换格式的命令示例：

```
ffmpeg -pix_fmt nv12 -s 1920x1080 -i nv12.yuv -pix_fmt yuv420p yuv420p.yuv
```

2.2 编码主观增强

每次调用视频主观增强接口，都会尝试在 sdk 创建一个新的主观增强模块，每一帧视频画面在送入编码器前，sdk 会首先调用所有创建成功的主观增强模块，对画面进行对应的处理。接口定义如下：

```
/**  
 * @brief add preprocess module into sdk  
 *  
 * @param ext_para[in] sdk lincense auth handler  
 * @param opts[in] opts string  
 * @return int  
 */  
int add_preprocess(void **ext_para, char *opts);
```

注意事项：

- 该接口需要在成功调用 **xxx_encoder_open** 开启编码器后才可以调用，否则 lincense auth handler 是非法值，将导致函数返回失败。
- **args** 中的参数也是成对出现

伪代码示例：

```
xxx_encoder_open(&extra, .....);  
char args[120] = "sharp_size=3:sharp_amount=0.5"  
add_preprocess(&extra, opts);
```

2.3 sharp 主观增强模块

参数说明

参数名称	说明	取值范围
sharp_size	sharp 尺寸	int [3, 7]
sharp_amount	sharp 强度	float [0, 1]

效果对比

- 编码器: ten264
- 编码参数: preset ultrafast threads=4:crf=21.5:keyint=infinite:p-opt=1,0,4,1.5:tune=zerolatency_shxf
- sharp 参数: "sharp_size=3:sharp_amount=0.5"

未开启 sharp	开启 sharp
real 0m3.961s user 0m7.704s sys 0m1.300s	real 0m4.532s user 0m9.124s sys 0m0.744s



- 编码器: ten265
- 编码参数: tune=shortlatency:preset=9:crf=26:keyint=1000:ltr=0:max-dpb=16:aq-mode=1:level=5:gop-struct=lowdelayB:pool-threads=4
- sharp 参数: "sharp_size=3:sharp_amount=0.5"

未开启 sharp	开启 sharp
real 0m7.277s	real 0m8.152s
user 0m24.840s	user 0m29.312s
sys 0m0.972s	sys 0m1.028s



2.4 Region of interest (ROI) 画质设置

- **ten264 编码器**

关闭 ROI 功能：



开启 ROI 功能：



- **ten265 编码器**

关闭 ROI 功能：



开启 ROI 功能 (块效应减弱):



三. ten264 编码器编程指引

3.1 编码设置参数

相关接口：

```
/**
 * @brief init ten264_param_t structure
 *
 * @param t[out] the ten264_param_t structure
 */
void ten264_param_default( ten264_param_t *t);

/**
 * @brief apply preset to ten264_param_t structure
 *
 * @param t[out] the ten264_param_t structure
```

```

* @param preset[in] must in ten264_preset_names
* @return int, 0: success, other: failure
*/
int ten264_param_default_preset( ten264_param_t *t, const char *preset);

/**
* @brief parse a param, and set to the ten264_param_t structure
*
* @param t[out] the ten264_param_t structure
* @param name[in] param name
* @param value[in] param value
* @return int, 0: success, other: failure
*/
int ten264_param_parse( ten264_param_t *t, const char *name, const char *value );

```

伪代码示例：

```

ten264_param_t params;
ten264_param_default(&params);
ten264_param_default_preset(&params, "ultrafast");
ten264_param_parse(&params, "tune", "zerolatency_shxf");
.....
ten264_param_parse(&params, "crf", "30");

```

注意事项：

- **ten264 编码器中，tune 参数需要通过 ten264_param_parse 进行设置，如上述伪代码所示。**

3.2 开启编码器

相关接口：

```

/**
* @brief open an encoder with ten264_param_t structure
*
* @param t[in] the ten264_param_t structure
* @param ext_para[out] sdk lincense auth handler
* @param config[in] sdk config file path
* @return ten264_t* the encoder handler, NULL on failure
*/
ten264_t *ten264_encoder_open( ten264_param_t *t, void **ext_para, const char *config);

/**
* @brief return the SPS and PPS that will be used for the whole stream.
*
* returns the number of bytes in the returned NALs.
*
* returns negative on error.

```

```

*      the payloads of all output NALs are guaranteed to be sequential in memory.
*
* @param t[in] the encoder handler
* @param pp_nal[out] the output NAL headers
* @param pi_nal[out] the number of NAL units outputted in pp_nal
* @return int
*/
int ten264_encoder_headers( ten264_t *t, ten264_nal_t **pp_nal, int *pi_nal );

```

伪代码示例：

```

ten264_t *enc;
void *extra;
enc = ten264_encoder_open(&params, &extra, "sdk_config");

ten264_nal_t *nal;
int nnal;
int size = ten264_encoder_headers(enc, &nal, &nnal);

```

3.3 编码视频帧

相关接口：

```

/**
 * @brief init ten264_picture_t structure
 *
 * @param pic[out] the ten264_picture_t structure
 */
void ten264_picture_init( ten264_picture_t *pic );

/**
 * @brief encode one picture.
 *      returns the number of bytes in the returned NALs.
 *      returns negative on error and zero if no NAL units returned.
 *      the payloads of all output NALs are guaranteed to be sequential in memory.
 *
 * @param t[in] the encoder handler
 * @param pp_nal[out] the output NAL units
 * @param pi_nal[out] the number of NAL units outputted in pp_nal
 * @param pic_in[out] input picture
 * @param pic_out[out] output picture
 * @param non_mono[in] no use, send NULL
 * @param motion[in] no use, send 0
 * @param texture[in] no use, send 0
 * @param ext_para[in] sdk lincense auth handler

```

```

* @return int
*/
int ten264_encoder_encode( ten264_t *t, ten264_nal_t **pp_nal, int *pi_nal, ten264_picture_t *pic_in,
ten264_picture_t *pic_out,
int *non_mono, double motion, double texture, void **ext_para);

```

伪代码示例：

```

ten264_picture_t pic, pic_out;
ten264_picture_init(&pic);
ten264_picture_init(&pic_out);
/* fill img datas */
for (i = 0; i < num_planes; i++) {
    pic.img.plane[i] = frame->data[i];
    pic.img.i_stride[i] = frame->linesize[i];
}
if (ret = ten264_encoder_encode(enc, &nal, &nnal, &pic, &pic_out, NULL, 0, 0, &extra) < 0)
    error;
/* save nal datas */

```

3.4 关闭编码器

相关接口：

```

/**
* @brief return the number of currently delayed (buffered) frames
*
* this should be used at the end of the stream, to know when you have all the encoded frames.
*
* @param t[in] the encoder handler
* @return int, num of delayed (buffered) frames
*/
int ten264_encoder_delayed_frames( ten264_t *t );

```

```

/**
* @brief close an encoder
*
* @param t[in] the encoder handler
* @param ext_para[in] sdk lincense auth handler
*/
void ten264_encoder_close( ten264_t *t, void **ext_para);

```

伪代码示例：

```

/* flush encoder */
int ret;
do {

```

```

    ret = ten264_encoder_encode(enc, &nal, &nnal, NULL, &pic_out, NULL, 0, 0, &extra);
    if (ret < 0)
        error;
    /* save nal datas */
} while (!ret && ten264_encoder_delayed_frames(enc));
/* close encoder */
ten264_encoder_close(enc, &extra);

```

四. ten265 编码器编程指引

4.1 编码设置参数

相关接口：

```

/* ten265_param_alloc:
 *   Allocates an ten265_param_t instance. The returned param structure is not
 *   special in any way, but using this method together with ten265_param_free()
 *   and ten265_param_parse() to set values by name allows the application to treat
 *   ten265_param_t as an opaque data struct for version safety */
ten265_param_t *ten265_param_alloc(void);

/* ten265_param_default:
 *   fill ten265_param_t with default values*/
void ten265_param_default(ten265_param_t *param);

/* ten265_param_parse:
 *   set one parameter by name.
 *   returns 0 on success, or returns one of the following errors.
 *   note: BAD_VALUE occurs only if it can't even parse the value,
 *   numerical range is not checked until ten265_encoder_open().
 *   value=NULLPTR means "true" for boolean options, but is a BAD_VALUE for non-booleans. */
uint32_t ten265_param_parse(ten265_param_t *param, const char *name, const char *value);

```

伪代码示例：

```

ten265_param_t *params;
params = ten265_param_alloc();
ten265_param_default(param);
ten265_param_parse(param, "preset", "9");
.....
ten265_param_parse(param, "crf", "30");

```

注意事项：

- **ten265 编码器中，建议将参数“pool-threads”设置为“4”。**

4.2 开启编码器

相关接口：

```
/* ten265_encoder_open:
 *      create a new encoder handler*/
ten265_t *ten265_encoder_open(ten265_param_t *param, void **ext_para, float vquality, const char *
config);

/* ten265_encoder_header:
 *      return the VPS, SPS and PPS size that will be used for the whole stream.*/
int32_t ten265_encoder_header(ten265_t *handle, ten265_nal_t **pp_nal, int32_t *nnal);
```

伪代码示例：

```
ten265_t *enc;
void *extra;
enc = ten265_encoder_open(param, &extra, "sdk_config");

ten264_nal_t *nal;
int nnal;
int size = ten265_encoder_header(enc, &nal, &nnal);
```

4.3 编码视频帧

相关接口：

```
/* ten265_encoder_encode:
 *      put the inpic into the encoder, get the outpic from the encoder, return the num of outpic */
int32_t ten265_encoder_encode(ten265_t *handle, ten265_inpic_t *inpic, ten265_outpic_t *outpic, void
**ext_para);
```

伪代码示例：

```
ten265_inpic_t pic;
ten265_outpic_t *pic_out;
/* fill img datas */
for (i = 0; i < num_planes; i++) {
    pic.plane[i] = frame->data[i];
    pic.stride[i] = frame->linesize[i];
}
if (ret = ten265_encoder_encode(enc, &pic, pic_out, &extra) < 0)
    error;
```

```
/* save nal datas */
// pic_out->nal
```

注意事项：

- **ten265 编码器不支持 nv12 格式的输入数据，需要将其转换为 yuv420p 格式再送入编码器。**用 ffmpeg 转换格式的命令示例：

```
ffmpeg -pix_fmt nv12 -s 1920x1080 -i nv12.yuv -pix_fmt yuv420p yuv420p.yuv
```

4.4 关闭编码器

相关接口：

```
/* ten265_encoder_close:
 *      close an encoder handler */
int32_t ten265_encoder_close(ten265_t *handle, void **ext_para);

/* ten265_param_free:
 *      Use ten265_param_free() to release storage for an ten265_param_t instance
 *      allocated by ten265_param_alloc() */
void ten265_param_free(ten265_param_t *param);
```

伪代码示例：

```
ten265_encoder_close(enc, &extra);
ten265_param_free(param);
```

五. SDK 编码参数附录

SDK 内置的 ffmpeg 已经集成了 sdk 所提供的视频编码器：Ten264 和 Ten265。编码器类型以及编码器参数都可以在视频处理时通过命令参数进行设置。

每种编码器的指定和设置方式如下：

编码器名称	指定方式	设置方式
Ten264	-vcodec libten264 -c:v libten264	-ten264opts name1=value1:name2=value2
Ten265	-vcodec libten265 -c:v libten265	-ten265-params name1=value1:name2=value2

每种编码器的具体参数及其详细说明如下列 2 个表格所示。

Ten264 编码器常用参数说明：

参数名称	参数说明
preset	指定编码器编码参数集合的配置。0: "ultrafast", 1: "superfast", 2: "veryfast", 3:"faster", 4:"fast", 5:"medium", 6:"slow", 7:"slower", 8:"veryslow",9: "placebo"。
bitrate	ABR 模式下面的输出视频的码率。
crf	CRF 模式下面的 CRF 数值。
aq-mode	0：关闭 aqmode 1：开启 aqmode 2：基于方差的 aqmode 3：基于方差的 aqmode 并且偏向于暗场景。默认是 2 会产生更好的 ssim 结果。
vbv-maxrate	vbv 的最大码率。默认情况下这个数值和配置的码率相同。
vbv-buFSIZE	vbv 缓冲 buffer 的大小。默认情况下这个数值是配置的码率的四倍。
rc-lookahead	lookahead 长度。
scenecut	是否开启场景切换。默认开启，一般不建议关闭。
keyint	关键帧最长间隔。默认是 256，可以根据实际业务情况配置，一般配置成 2~5s 的时间间隔的帧数。
threads	使用的线程池的线程数。
lookahead-threads	预分析，lookahead 使用线程数。
profile	"baseline", "main", "high", "high422", "high444"

Ten265 编码器常用参数说明：

参数名称	参数说明
------	------

preset	指定编码器编码参数集合的配置。-1: ripping 0: placebo; 1:veryslow; 2: slower; 3: slow; 4: universal; 5 medium; 6: fast; 7: faster; 8:veryfast; 9 superfast。
rc	码率控制方式 。0:CQP 1:ABR_VBV 2:ABR 3:CRF_VBV 4:CRF
bitrate	ABR 模式下面的输出视频的码率。
crf	CRF 模式下面的 CRF 数值，取值范围：[1,51]。
aq-mode	0：关闭 aqmode 1：开启 aqmode 2：基于方差的 aqmode 3：基于方差的 aqmode 并且偏向于暗场景。默认是 2 会产生更好的 ssim 结果。
vbv-maxrate	vbv 的最大码率。默认情况下这个数值和配置的码率相同。
vbv-bufsize	vbv 缓冲 buffer 的大小。默认情况下这个数值是配置的码率的四倍。
rc-lookahead	lookahead 长度。
scenecut	场景切换阈值，取值范围[0,100]。0:关闭，默认开启，一般不建议关闭。
open-gop	是否开启 open gop。0：关闭，1：开启。默认开启，在直播场景中为了支持随机接入，建议关闭这个功能。
keyint	关键帧最长间隔。默认是 256，可以根据实际业务情况配置，需要大于 50 且是 8 的倍数。
ltr	是否要支持长期参考帧 。0：关闭，1: 开启。默认开启，如果播放 HEVC 视频的硬件设备比较差，建议关闭这个功能。
pool-threads	WPP 使用的线程池的线程数。默认和 CPU 的核数相同，如果想减少 CPU 占用，可以降低这个数目。