UVS Media SDK

用户手册

淘宝官方购买请搜索
pipivision USB3.0 高清 HDMI 视频采集卡

一、 概述

UVS Media SDK 是基于 Windows DirectShow 或者 Linux V4L2 多媒体框架实现的音视频应用功能库，通过 UVS Media SDK，用户可以更加方便、高效地进行音视频开发，UVS Media SDK 可以支持以下功能：

音视频设备管理，包括设备枚举、设备插入检测、设备打开及关闭；
音视频预览，音量控制，视频效果调节；
音视频数据多种模式采集，包括数据回调、锁定或拷贝方式
音视频采集的数据带时间戳；
支持图像色彩空间转换，例如 YUY2/YV12/NV12/I420/I422/I444/RGB24/RGB32 等格式；
支持图像抓拍保存功能，可以保存为 BMP、JPG 或 PNG 格式；
支持预览和编码 OSD 文字、图片、规则几何形状叠加，以及叠加透明度和模式控制；
支持音频多种编码格式，MP3/AAC 等；
支持视频多种编码格式，H.264/H.265 等；
支持 Intel Media SDK 和 NVidia CUDA 视频编码加速（Windows XP 除外）；
支持设备同时编码多路不同分辨率视频流和录像；
支持音视频录像，支持录像异常结束恢复功能（mp4 文件格式）；
支持 mp4 录像文件分割、Title 编辑、合并功能；
支持 RTSP/RTMP 音视频编码数据推流；
支持 NDI 协议无损音视频数据推流；

二、开发环境

UVS Media SDK for Windows 包括定义头文件、链接库和动态库文件 UVSMedia.dll、UVSCommon.dll 以及视频编码库，其中编码库是可选的，用户可以根据需要使用。

UVS Media SDK for Linux 包括定义头文件和动态库文件 UVSMedia.so、UVSCommon.so 以及视频编码库，其中编码库是可选的，用户可以根据需要使用。

SDK 二次开发示例有 QT、C#、Delphi、Java、VC++等，具体使用请参考相关源代码，本文档将以 C 语言为例对 SDK API 功能进行介绍。

三、主要功能一览表

操作系统支持 OS Support	Windows XP/7/8/10	Ubuntu 18/20/21 及统信 UOS(x86_64)
数据采集模式 Data Grab	Callback, Lock/Unlock, Copy	
数据时间戳 Timestamp	Audio/Video Data With Timestamp	
文字图片叠加 OSD	Time, Text, Image, Line, Rectangle, Circle, Preview/Encode OSD	
视频编码 Video Encode	H.264, H.265, Intel® Media SDK, Nvidia® CUDA Video Acceleration	
视频转换 Video Conversion	YUY2, UYVY, NV12, YV12, I420, I422, I444, RGB24, RGB32	
音频编码 Audio Encode	AAC, MP3	
录像文件 File Recode	MP4, AVI(Windows Only), FLV	
抓拍文件 Snapshot	bmp, jpg, png	
RTSP	H.264, H.265, MJPG/AAC, MP3	
RTMP	H.264/AAC	
NDI® Lossless Stream	Lossless Audio/Video NDI Stream	
MP4 录像 MP4 File Operate	Repair, Cut, Merge, Title Add	
多路编码流 Multi Stream	REC, RTSP, RTMP, Multiple Video Stream Encoding Each Channel	

四、API 接口的一般调用流程

操作步骤	调用接口
初始化	uvs_media_init
设备管理	uvs_enum_device uvs_set_device_callback uvs_get_device_count
设备打开	uvs_dev_open
工作配置	uvs_set_audio_device uvs_set_video_property
设备启动	uvs_dev_control
功能操作	uvs_start_preview uvs_set_time_osd uvs_set_audio_raw_frame_callback uvs_set_video_raw_frame_callback uvs_save_video_snapshot uvs_set_video_encode_config uvs_start_record uvs_start_encode ….
设备关闭	uvs_dev_close
退出	uvs_media_exit

五、初始化和退出

使用 SDK 的任何功能前，必须先初始化，之后可以获取当前安装的设备个数和信息。不再使用 SDK 功能后，必须反初始化，否则可能导致异常。

```
uvsobj_handle obj = NULL;
int device_open(void)
{
    int num;
    uvs_dev_info_t dev[16];

    uvs_media_init(0);

    num = 16;
    uvs_enum_device(uvs_dev_video_capture, dev, &num);
    if (num == 0) return UVS_ERR;

    obj = uvs_dev_open(&dev[0], &result);
    if (result != UVS_OK) return result;

    /* set audio capture device if necessary */
    // uvs_enum_device(uvs_dev_audio_capture, ...);
    // uvs_set_audio_device(obj, uvs_dev_audio_capture, ...);
```

```
    result = uvs_dev_control(obj, uvs_dev_run);

    return result;
}
```

六、USB 设备状态

对于 USB 设备，可能存在插拔的情况，另外当音视频输入状态或格式改变时，也可能触发该状态通知，因此一般情况下必须处理。

事件处理由单独的线程调用，因此需要注意对相关资源进行线程同步控制，否则可能导致异常，事件处理函数定义如下：

```
void CALLBACK device_callback(uvs_dev_notify_e notify, const uvs_dev_info_t *info, uvsobj_handle devobj, void *userdata)
{
    if (notify == uvs_dev_changed)
    {
        // open device
    }
    else
    { // uvs_dev_closed
        if (devobj)
        {
            // close the device with handle
        }
    }
}
```

注册对应的事件才能生效：

```
uvs_set_device_callback(device_callback, NULL);
```

七、视频预览

对于打开的设备，可以将视频显示到指定的窗口，同时还支持预览控制，例如冻结显示、设置显示位置和视频裁剪等。

```
int preview_start(void)
{
    int result = device_open();
    if (result != UVS_OK) return result;

    result = uvs_start_preview(obj, hwnd, uvs_disp_type_d3d);
    if (result != UVS_OK) return result;

    return result;
}

void preview_stop(void)
```

```
{
	uvs_stop_preview(obj);
	device_close();
}
```

八、打开设备并获取数据

通过打开 uvs_enum_device 获取到的 uvs_dev_video_capture 设备，用户可以使用句柄获取数据，获取数据有 Callback、Lock/Unlock、Copy 方式，前二种方式适合简单的非耗时数据处理，对于耗时的数据处理，建议使用 Copy 方式。使用 Copy 方式时，可以使用内部缓冲区，用户也可以使用自己管理的数据缓冲区，具体请参考相关说明。

```
void CALLBACK frame_callback(uvsobj_handle obj, const uvs_frame_info_t *frame, void *userData)
{
	if (frame)
	{
		// do something
	}
}

int callback_start(void)
{
	int result = device_open();
	if (result != UVS_OK) return result;

	result = uvs_set_video_raw_frame_callback(obj, frame_callback, NULL);
	if (result != UVS_OK) return result;

	// do something

	return result;
}

void callback_stop(void)
{
	uvs_set_video_raw_frame_callback(obj, NULL, NULL);
	device_close();
}

int frame_lock(void)
{
	int result;
	uvs_frame_info_t frame = { uvs_frame_NONE };

	result = uvs_lock_video_raw_frame(obj, &frame, 1000);
```

```
    if (result != UVS_OK) return result;

    // do something

    return result;
}

void frame_unlock(void)
{
    uvs_unlock_video_raw_frame(obj);
}

int frame_copy(void)
{
    int result;
    uvs_frame_info_t frame = { uvs_frame_NONE };

    result = uvs_copy_video_frame(obj, NULL, &frame, 1000);
    if (result != UVS_OK) return result;

    // do something

    return result;
}
```

九、音视频编码

启动音视频编码以降低数据量进行录像存储或 RTSP、RTMP 推流，本 SDK 支持每个设备同时编码多个不同分辨率和格式的视频流，每个视频编码流都可以指定唯一的索引号，具体数量仅受系统资源限制。

视频编码器的启用需要对应的模块支持，使用 Video Acceleration 编码还需要硬件支持，否则将无法使用，这些编码模块以单独的库方式存在，使用时必须拷贝到当前目录或系统目录，用户可以通过接口 uvs_query_video_encoder(NULL, …)查询对应的模块是否可用。

```
int encode_start(void)
{
    int result = device_open();
    if (result != UVS_OK) return result;

    /* set audio codec if necessary */
    // uvs_set_audio_property(obj, NULL, uvs_audio_codec_AAC);

    result = uvs_set_video_encode_config(obj, 0, uvs_video_codec_sw_h264, ...);
    if (result != UVS_OK) return result;

    result = uvs_start_encode(obj, 0);
```

```
    if (result != UVS_OK) return result;

    return result;
}

void encode_stop(void)
{
    uvs_stop_encode(obj, 0);
    device_close();
}
```

十、录像控制

音视频编码设置成功后，可以进行录像控制，由于可能存在多个编码流，因此录像也需要指定编码流索引号。

```
int record_start(void)
{
    int result = encode_start();
    if (result != UVS_OK) return result;

    /* set metadata if necessary */
    // uvs_set_record_metadata(obj, 0, "title", "comment", NULL, NULL, NULL);

    // path example, "D:\\test%H%m%s.mp4" for Windows, "/test%H%m%s.mp4" for Linux
    result = uvs_start_record(obj, 0, path, NULL, FALSE, TRUE, 0, 60000);
    if (result != UVS_OK) return result;

    return result;
}

void record_stop(void)
{
    uvs_stop_record(obj, 0);
    encode_stop();
}
```

十一、RTSP 推流

音视频编码设置成功后，可以进行 RTSP 推流，由于可能存在多个编码流，因此 RTSP 推流也需要指定编码流索引号。要启动 RTSP 推流，必须先使用端口创建 RTSP 服务，如果端口被占用或者端口还在等待释放，可能导致服务创建失败，此时可以多次尝试创建 RTSP 服务。

```
    int rtsp_start(void)
    {
        uvs_url_t url;
        int result = encode_start();
```

```
        if (result != UVS_OK) return result;

        result = uvs_media_server_rtsp_create(8554, 0, NULL, NULL);
        if (result != UVS_OK) return result;

        result = uvs_media_stream_rtsp_start(obj, 0, 8554, FALSE, "test", NULL);
        if (result != UVS_OK) return result;

        uvs_media_stream_rtsp_get_url(obj, 0, &url);
        return result;
    }

    void rtsp_stop(void)
    {
        encode_stop();
        uvs_media_stream_rtsp_stop(obj, 0);
        uvs_media_server_rtsp_destroy(8554);
    }
```

十二、RTMP 推流

音视频编码设置成功后，可以进行 RTMP 推流，由于可能存在多个编码流，因此 RTMP 推流需要指定编码流索引号。

```
int rtmp_start(void)
{
    int result = encode_start();
    if (result != UVS_OK) return result;

    result = uvs_media_stream_rtmp_send(obj, 0, "rtmp://your.domain /0", FALSE);
    if (result != UVS_OK) return result;

    return result;
}

void rtmp_stop(void)
{
    uvs_media_stream_rtmp_stop(obj, 0);
    encode_stop();
}
```

十三、错误代码

错误代码	说明
UVS_OK	成功
UVS_ERR	失败
UVS_ERR_OUTOFMEMORY	内存分配失败

UVS_ERR_INVALIDARG	参数错误
UVS_ERR_WAIT_TIMEOUT	响应超时
UVS_ERR_NOT_INITIALIZED	SDK 未初始化
UVS_ERR_NOT_ENOUGH_BUFFER	需要的内存空间不足
UVS_ERR_NOT_READY	操作未就绪
UVS_ERR_NOT_SUPPORT	不支持的操作
UVS_ERR_FILE_OPEN	文件创建或打开失败
UVS_ERR_ALREADY_EXISTS	不支持的重复操作

十四、API 接口介绍

本 SDK 的涉及字符串操作的 API 接口，分为 ANSI 和 UNICODE16 版本（增加 W 后缀），接口参数有[IN]Input 类型，[OUT]Output 类型，[IN][OUT]Input 以及 Output 类型，除非特别注明，否则参数一律为[IN]类型，以下分别对 SDK 的 API 接口进行介绍。

1、uvs_media_init

说明	SDK 初始化，使用其他 API 接口之前必须最先调用
参数	int flags SDK 库初始化标志，1 设备即时检测，可能出现多次设备事件通知；2 独占方式打开设备（多进程）；0 默认值。
返回值	成功返回 UVS_OK，否则失败

2、uvs_media_exit

说明	SDK 退出，不再使用 API 接口功能
参数	无
返回值	成功返回 UVS_OK，否则失败

3、uvs_get_version

说明	获取 SDK 版本号
参数	无
返回值	返回当前 SDK 版本号，例如 0x02000001，表示版本 2.0.0.1

4、uvs_enum_device

说明	音视频设备枚举，可以分别获取几类设备
参数	uvs_dev_type_e devType 需要枚举的设备类型，uvs_dev_video_capture 视频捕获设备，uvs_dev_audio_capture 音频捕获设备，uvs_dev_audio_renderer 音频播放设备； uvs_dev_info_t devInfo[] 存放获取到的设备信息数组； int *devNum 参数类型为[IN][OUT]，初始化为数组 uvs_dev_info_t 个数，返回实际的设备数。
返回值	成功返回 UVS_OK，否则失败

5、uvs_set_device_callback

说明	设备通知回调函数设置，可以侦测设备状态改变
参数	pUVSDeviceCallback pCB 回调函数指针，为 NULL 表示结束回调通知；

	void *pUserData 用户自定义数据指针，该参数将传递给回调函数。 回调函数 pUVSDeviceCallback pCB 使用的参数有： uvs_dev_notify_e nofity 设备状态通知，uvs_dev_changed 设备改变，uvs_dev_closed 设备关闭； uvs_dev_info_t *info 相关设备的信息； uvsobj_handle obj 对应的句柄，当收到 uvs_dev_closed 设备关闭通知时，如果该设备已打开，则需要调用接口 uvs_dev_close 关闭对应的设备； void *pUserData 用户自定义数据指针。
返回值	成功返回 UVS_OK，否则失败

6、uvs_get_device_count

说明	获取视频捕获设备个数
参数	无
返回值	返回视频捕获设备数，0 表示无可用设备

7、uvs_query_video_encoder

说明	检查可用的视频编码器
参数	uvsobj_handle obj 设备句柄，一般为 NULL； uvs_video_codec_e videoCodec 视频编码器类型， uvs_video_codec_intel_h264 Intel H264 编码加速， uvs_video_codec_intel_h265 Intel H265 编码加速， uvs_video_codec_intel_jpeg Intel MJPG 编码加速， uvs_video_codec_nvidia_h264 NVidia H264 编码加速， uvs_video_codec_nvidia_h265 NVidia H265 编码加速， uvs_video_codec_sw_h264 软件 H264 编码。
返回值	成功返回 UVS_OK，否则失败

8、uvs_calc_buffer_size

说明	计算视频帧缓冲大小
参数	uvs_frame_type_e frameType 视频帧类型，可以是 YUV/RGB 格式； int videoWidth 视频宽度； int videoHeight 视频高度； int nStride 数据对齐长度，0 表示默认 4 字节对齐的长度。
返回值	成功返回 UVS_OK，否则失败

9、uvs_set_nosignal_image

说明	设置无视频预览显示的图片
参数	const char *szFileName 图片文件路径，支持 bmp、jpg、png 文件，为 NULL 时表示不显示图片； uvs_draw_mode_e drawMode 图片显示模式，支持拉伸模式 uvs_draw_stretch 和居中模式 uvs_draw_center； COLORREF bkColor 使用居中模式时，填充的颜色。
返回值	成功返回 UVS_OK，否则失败

10、uvs_dev_open

说明	打开设备，操作设备之前必须调用该接口获取设备句柄。
参数	uvs_dev_info_t *devInfo 需要打开的设备信息结构指针，该信息通过设备枚举或设备检测回调获得； int *result 参数类型为[OUT]，返回操作结果，传入 NULL 表示忽略。
返回值	成功返回设备句柄，失败返回 NULL。

11、uvs_dev_close

说明	关闭设备，对应的设备句柄不能再使用。
参数	uvsobj_handle obj 需要关闭的设备句柄。
返回值	成功返回 UVS_OK，否则失败

12、uvs_enum_audio_format

说明	枚举音频捕获设备支持的数据格式
参数	uvsobj_handle obj 对应的设备句柄； uvs_audio_format_t formats[] 存放音频格式信息的数组； int *formatNum 参数类型为[IN][OUT]，初始化为数组 uvs_audio_format_t 个数，返回实际的个数。
返回值	成功返回 UVS_OK，否则失败

13、uvs_enum_video_format

说明	枚举视频捕获设备支持的数据格式
参数	uvsobj_handle obj 对应的设备句柄； uvs_source_type_e sourceType 视频源 pin 类型，uvs_source_any 任意类型，uvs_source_preview 预览，uvs_source_capture 捕获，一般可设为 uvs_source_any； uvs_video_format_t formats[] 存放视频格式信息的数组； int *formatNum 参数类型为[IN][OUT]，初始化为数组 uvs_video_format_t 个数，返回实际的个数。
返回值	成功返回 UVS_OK，否则失败

14、uvs_set_audio_device

说明	设置音频设备，包括音频捕获设备和音频播放设备，必须在设备开始运行之前设置。
参数	uvsobj_handle obj 对应的设备句柄； uvs_dev_type_e devType 音频设备类型，uvs_dev_audio_capture 音频捕获设备，uvs_dev_audio_renderer 音频播放设备； uvs_dev_info_t *devInfo 音频设备信息结构指针，可以通过枚举设备获得。
返回值	成功返回 UVS_OK，否则失败

15、uvs_create_property_page

说明	创建设备属性页模态对话框，查看对应的信息。

参数	uvsobj_handle obj 对应的设备句柄； uvs_property_page_e propertyPage 属性页类型； HWND hOwner 关联的拥有者窗口句柄； int x 对应的 X 坐标； int y 对应的 Y 坐标； char *szCaption 窗口标题。
返回值	成功返回 UVS_OK，否则失败

16、uvs_dev_control

说明	设备状态控制，开始音视频数据处理前必须先运行设备。
参数	uvsobj_handle obj 对应的设备句柄； uvs_dev_state_e state 设备控制类型，uvs_dev_run 设备运行，uvs_dev_pause 设备暂停，uvs_dev_stop 设备停止。
返回值	成功返回 UVS_OK，否则失败

17、uvs_get_audio_property

说明	获取音频数据格式和编码器类型。
参数	uvsobj_handle obj 对应的设备句柄； uvs_audio_format_t *format 参数类型为[OUT]，当前的音频数据格式； uvs_audio_codec_e *audioCodec 参数类型为[OUT]，当前的音频编码器类型，uvs_audio_codec_none 当前无音频编码器，uvs_audio_codec_MP3 MP3 音频编码格式，uvs_audio_codec_AAC AAC 音频编码格式。
返回值	成功返回 UVS_OK，否则失败

18、uvs_set_audio_property

说明	设置音频编码器类型。
参数	uvsobj_handle obj 对应的设备句柄； uvs_audio_format_t *format 音频数据格式，该参数保留为 NULL； uvs_audio_codec_e *audioCodec 当前的音频编码器类型，uvs_audio_codec_none 无音频编码，uvs_audio_codec_MP3 MP3 音频编码格式，uvs_audio_codec_AAC AAC 音频编码格式。
返回值	成功返回 UVS_OK，否则失败

19、uvs_set_audio_mute

说明	音频播放静音控制开关。
参数	uvsobj_handle obj 对应的设备句柄； BOOL bMute 静音控制，为 TRUE 表示静音，否则取消静音。
返回值	成功返回 UVS_OK，否则失败

20、uvs_get_audio_volume

说明	音频播放音量获取
参数	uvsobj_handle obj 对应的设备句柄； int *volume 参数类型为[OUT]，音量大小，取值为 0 到 100。

返回值	成功返回 UVS_OK，否则失败

21、uvs_set_audio_volume

说明	音频播放音量设置
参数	uvsobj_handle obj 对应的设备句柄； int volume 音量大小，取值为 0 到 100。
返回值	成功返回 UVS_OK，否则失败

22、uvs_get_audio_balance

说明	音频播放声道平衡获取
参数	uvsobj_handle obj 对应的设备句柄； int *balance 参数类型为[OUT]，左右声道平衡，取值为-100 到 100，0 表示中间值。
返回值	成功返回 UVS_OK，否则失败

23、uvs_set_audio_balance

说明	音频播放声道平衡设置
参数	uvsobj_handle obj 对应的设备句柄； int balance 左右声道平衡，取值为-100 到 100，0 表示中间值。
返回值	成功返回 UVS_OK，否则失败

24、uvs_get_video_effect_range

说明	获取视频效果参数范围
参数	uvsobj_handle obj 对应的设备句柄； uvs_video_effect_e effect 视频效果类型，uvs_video_brightness 图像亮度，uvs_video_contrast 图像对比度，uvs_video_hue 图像色调，uvs_video_saturation 图像饱和度，uvs_video_sharpness 图像锐利度，uvs_video_gamma 图像伽玛值，uvs_video_gain 图像增益； int *minVal 取值范围最小值； int *maxVal 取值范围最大值； int *stepDelta 取值步长，一般为 1； int *defaultVal 对应效果的默认值。
返回值	成功返回 UVS_OK，否则失败

25、uvs_get_video_effect

说明	获取视频效果参数值
参数	uvsobj_handle obj 对应的设备句柄； uvs_video_effect_e effect 视频效果类型； int *param 参数类型为[OUT]，存放对应的参数值。
返回值	成功返回 UVS_OK，否则失败

26、uvs_set_video_effect

说明	设置视频效果参数值

参数	uvsobj_handle obj 对应的设备句柄； uvs_video_effect_e effect 视频效果类型； int param 需要设置的参数值。
返回值	成功返回 UVS_OK，否则失败

27、uvs_get_video_property

说明	获取当前的视频格式，包括视频分辨率、帧率、色彩空间。
参数	uvsobj_handle obj 对应的设备句柄； uvs_source_type_e sourceType 视频源 pin 类型，uvs_source_any 任意类型，uvs_source_preview 预览，uvs_source_capture 捕获，一般可设为 uvs_source_any； uvs_video_format_t *format 参数类型为[OUT]，存放对应的视频格式参数。
返回值	成功返回 UVS_OK，否则失败

28、uvs_set_video_property

说明	设备的视频格式设置，必须在设备开始运行之前设置。
参数	uvsobj_handle obj 对应的设备句柄； uvs_source_type_e sourceType 视频源 pin 类型，uvs_source_any 任意类型，uvs_source_preview 预览，uvs_source_capture 捕获，一般可设为 uvs_source_any； uvs_video_format_t *format 需要设置的视频格式参数。
返回值	成功返回 UVS_OK，否则失败

29、uvs_get_video_mirror

说明	获取视频翻转状态
参数	uvsobj_handle obj 对应的设备句柄； BOOL *bHorizMirror 参数类型为[OUT]，水平翻转状态； BOOL *bVertMirror 参数类型为[OUT]，垂直翻转状态。
返回值	成功返回 UVS_OK，否则失败

30、uvs_set_video_mirror

说明	设置视频翻转状态
参数	uvsobj_handle obj 对应的设备句柄； BOOL bHorizMirror 视频水平翻转，将影响所有的视频数据； BOOL bVertMirror 视频垂直翻转，将影响所有的视频数据。
返回值	成功返回 UVS_OK，否则失败

31、uvs_get_video_status

说明	获取设备状态
参数	uvsobj_handle obj 对应的设备句柄； BOOL *bSignal 输入是否有视频信号，TRUE 表示有视频信号，否则无视频信号； BOOL *bMode 接口模式，对于 USB 设备，TRUE 为 USB3.0，否则为 USB2.0。

返回值	成功返回 UVS_OK，否则失败

32、uvs_set_video_encode_config

说明	设置视频编码参数
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，从 0 起始的任意不重复整数，可以支持多个流同时编码，每个流都有对应的索引号； uvs_video_codec_e videoCodec 视频编码器类型； uvs_encode_config_t *config 视频编码参数。 结构 uvs_encode_config_t 定义如下： uvs_target_usage_e targetUsage 编码模式，uvs_target_usage_quality 使用高质量编码模式，占用资源较多，uvs_target_usage_balance 使用平衡编码模式，uvs_target_usage_performance 使用高性能编码模式，编码效率最高； uvs_codec_profile_e codecProfile 编码级别，对应的编码复杂度程度，级别越高占用的资源越多，编码质量也更好，同时也会影响解码复杂度。uvs_codec_profile_auto 自动选择，uvs_codec_profile_baseline 基本级别，uvs_codec_profile_main 普通级别，uvs_codec_profile_high 高级别； uvs_avc_entropy_e entropyCoding H264 熵编码算法，avc_entropy_CABAC CABAC 算法，uvs_avc_entropy_CAVLC CAVLC 算法； uvs_scale_usage_e scaleUsage 编码图像缩放模式，各模式使用的图像缩放算法不同，质量越高则占用的资源越多，uvs_scale_usage_performance 高性能模式，速度最快，uvs_scale_usage_balance 性能平衡模式，uvs_scale_usage_quality 图像质量优先模式，uvs_scale_usage_high_quality 图像缩放高质量模式，占用资源最多； int scaleWidth 编码图像宽度，以像素为单位，值为 0 表示图像原始大小； int scaleHeight 编码图像高度，以像素为单位，值为 0 表示图像原始大小； int cropLeft 编码图像裁剪 X 坐标； int cropTop 编码图像裁剪 Y 坐标； int cropWidth 编码图像裁剪宽度，值为 0 表示图像原始大小； int cropHeight 编码图像裁剪宽度，值为 0 表示图像原始大小； float frameRate 视频编码帧率，值为 0 表示视频原始帧率； uvs_video_rcmode_e rcMode 编码码率控制模式，uvs_video_rcmode_VBR 动态码率编码，在指定的码率范围内根据视频图像自动调节，uvs_video_rcmode_CBR 固定码率编码，按照固定的码率编码，uvs_video_rcmode_CQP 使用固定的 QP 进行编码，码率也是动态变化的，uvs_video_rcmode_AVBR 动态码率编码，相对 VBR 模式，码率变化幅度更加平滑； UINT encBitRate 编码码率，单位为 kbit/sec，码率越高图像质量越好，建议取值为 4000 到 10000； UINT maxBitRate 最大编码码率，单位为 kbit/sec； UINT encQuality 编码质量，取值为 1 到 50，该值越低图像质量越好，码率也越高，建议取值为 25 到 35；

	UINT GOPLength 关键帧间隔，一般为帧率的倍数，值为 0 表示使用默认值。
返回值	成功返回 UVS_OK，否则失败

33、uvs_get_video_encode_config

说明	获取视频编码参数
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，可以支持多个流同时编码，每个流都有对应的索引号； uvs_video_codec_e *videoCodec 参数类型为[OUT]，对应的视频编码器类型； uvs_encode_config_t *config 参数类型为[OUT]，对应的视频编码参数。
返回值	成功返回 UVS_OK，否则失败

34、uvs_request_video_key_frame

说明	请求编码视频关键帧
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号。
返回值	成功返回 UVS_OK，否则失败

35、uvs_get_video_encode_status

说明	获取视频编码状态信息，例如编码的帧数和帧率
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号； uvs_encode_status_t *info 参数类型为[OUT]，存放编码状态信息。
返回值	成功返回 UVS_OK，否则失败

36、uvs_start_encode

说明	开始音视频流编码，在录像或推流之前必须先启动音视频编码
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，值为 UVS_STREAM_INDEX_ALL 时表示所有的编码流。
返回值	成功返回 UVS_OK，否则失败

37、uvs_stop_encode

说明	停止音视频流编码
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，值为 UVS_STREAM_INDEX_ALL 时表示所有的编码流。
返回值	成功返回 UVS_OK，否则失败

38、uvs_set_record_metadata

说明	设置录像文件信息

参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号； const char *szTitle 录像标题； const char *szComment 录像说明； const char *szArtist 演员； const char *szGenre 类别； const char *szComposer 作者。
返回值	成功返回 UVS_OK，否则失败

39、uvs_start_record

说明	开始录像
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号； const char *szSaveFile1 录像文件 1，录像文件名可以根据时间格式自动生成，字符串的时间格式以%开头，定义如下： %M 月份； %D 按照月份的天数； %H 24 小时制的时钟； %h 12 小时制的时钟； %m 分钟； %s 秒钟； %Y 年份； %t 12 小时制 AM/PM； %N 月份名； %n 月份名缩写； %W 按照星期的名称； %w 按照星期的名称缩写； %i 编码流索引号 streamIndex； %I 编码流索引号，从 1 开始（streamIndex + 1）； %% %号； const char *szSaveFile2 录像文件 2，生成备份的录像文件，不能与 szSaveFile1 相同，为 NULL 表示不生成备份录像； BOOL bRecAudio 是否录音频； BOOL bRepairSupport 是否支持录像中断修复； UINT uKBData 单个录像文件大小（Kilo Bytes），超过自动开始新录像，为 0 表示手动停止录像； UINT uMilliSec 单个录像文件时间（Milliseconds），超过自动开始新录像，为 0 表示手动停止录像。 如果同时设置录像大小和时间的限制，则达到任一条件时都会开始新的录像。
返回值	成功返回 UVS_OK，否则失败

40、uvs_get_record_filename

说明	获取当前的录像文件名
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号； uvs_filename_t fileName[] 参数类型为[OUT]，录像文件名； int *fileNum 参数类型为[IN] [OUT]，数组 fileName 大小，返回实际个数。
返回值	成功返回 UVS_OK，否则失败

41、uvs_pause_record

说明	暂停录像
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号； BOOL bRecResume 恢复或暂停录像； BOOL bRequestKeyFrame 是否请求编码关键帧。
返回值	成功返回 UVS_OK，否则失败

42、uvs_stop_record

说明	停止录像
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，值为 UVS_STREAM_INDEX_ALL 时表示所有的编码流。
返回值	成功返回 UVS_OK，否则失败

43、uvs_start_preview

说明	启动视频预览
参数	uvsobj_handle obj 对应的设备句柄； HWND hwnd 视频显示的窗口句柄； uvs_disp_type_e dispType 显示渲染器类型，uvs_disp_type_ddraw directdraw，uvs_disp_type_d3d direct3d。
返回值	成功返回 UVS_OK，否则失败

44、uvs_get_preview_freeze

说明	获取预览状态
参数	uvsobj_handle obj 对应的设备句柄； BOOL *bFreeze 参数类型为[OUT]，为 TRUE 表示显示暂停。
返回值	成功返回 UVS_OK，否则失败

45、uvs_set_preview_freeze

说明	设置预览状态
参数	uvsobj_handle obj 对应的设备句柄； BOOL bFreeze 暂停或恢复视频预览。
返回值	成功返回 UVS_OK，否则失败

46、uvs_set_preview_rect

说明	设置预览显示位置
参数	uvsobj_handle obj 对应的设备句柄； LPRECT wndRect 视频显示在窗口中的位置，参考坐标为窗口大小（0, 0, wndWidth, wndHeight），NULL 表示整个窗口； LPRECT cropRect 视频显示裁剪区域，参考坐标为视频大小（0, 0, videoWidth, videoHeight），NULL 表示不裁剪； COLORREF bkColor 窗口空白区域填充的背景色。
返回值	成功返回 UVS_OK，否则失败

47、uvs_stop_preview

说明	停止预览
参数	uvsobj_handle obj 对应的设备句柄。
返回值	成功返回 UVS_OK，否则失败

48、uvs_set_time_osd

说明	视频叠加时间 OSD
参数	uvsobj_handle obj 对应的设备句柄； int osdIndex OSD 对应的索引号，在多个 OSD 同时显示的情况下，按照索引号从低到高依次叠加，索引号索引号按照范围分为 3 类： 1、[uvs_osd_index_preview_enc, uvs_time_osd_index_preview_enc]表示预览和编码都叠加； 2、[uvs_osd_index_preview_only , uvs_time_osd_index_preview_only]仅预览叠加 OSD，编码不叠加 OSD； 3、[uvs_osd_index_frame_preview, uvs_time_osd_index_frame_preview]由 uvs_frame_preview 使用。 4、uvs_time_osd_index_preview_enc、uvs_time_osd_index_preview_only 以及 uvs_time_osd_index_frame_preview 有特殊意义，表示对应的时间 OSD 索引号。 int x 叠加 OSD 的水平起始坐标，参考坐标为视频原始大小(0, 0, width, height)； int y 叠加 OSD 的垂直起始坐标，参考坐标为视频原始大小(0, 0, width, height)； const char *szTimeMode 时间显示格式，例如 UVS_TIMEMODE_YMD_24H 或 UVS_TIMEMODE_MDY_12H 格式，字符串的时间格式以%开头，定义如下： %M 月份； %D 按照月份的天数； %H 24 小时制的时钟； %h 12 小时制的时钟； %m 分钟； %s 秒钟； %Y 年份； %t 12 小时制 AM/PM；

	%N 月份名； %n 月份名缩写； %W 按照星期的名称； %w 按照星期的名称缩写； %% %号； const uvs_font_info_t *info 字体格式，该结构定义如下： char szFontName[UVS_NAME_LEN] 字体名，默认 Arial； int fontPointSize 字体大小； uvs_font_style_e fontStyle 字体风格； uvs_string_format_e stringFormat 文字格式； COLORREF textColor 字体颜色； COLORREF textBkColor 字体背景色； int textOpacity 文字透明度； int textBkOpacity 背景透明度。
返回值	成功返回 UVS_OK，否则失败

49、uvs_set_text_osd

说明	视频叠加文本 OSD
参数	int width 文本显示宽度，为 0 表示默认宽度； int height 文本显示高度，为 0 表示默认高度； const char *szText 文本显示内容。 其他参数见 uvs_set_time_osd 的相关说明。
返回值	成功返回 UVS_OK，否则失败

50、uvs_set_image_osd

说明	视频叠加图片 OSD
参数	int width 图片显示宽度，为 0 表示图片原始宽度； int height 图片显示高度，为 0 表示图片原始高度； int imgOpacity 图片叠加透明度； const char *szFileName 图片文件路径。 其他参数见 uvs_set_time_osd 的相关说明。
返回值	成功返回 UVS_OK，否则失败

51、uvs_set_rect_osd

说明	视频叠加矩形
参数	int width 矩形宽度； int height 矩形高度； 结构 const uvs_style_info_t *info 定义如下： int lineSize 线条宽度； uvs_dash_style_e dashStyle 线条样式； COLORREF edgeColor 线条颜色；

	COLORREF fillColor 填充颜色； int edgeOpacity 线条叠加透明度； int fillOpacity 填充透明度； 其他参数见 uvs_set_time_osd 的相关说明。
返回值	成功返回 UVS_OK，否则失败

52、uvs_set_ellipse_osd

说明	视频叠加椭圆形
参数	int width 椭圆形宽度； int height 椭圆形高度； 结构 const uvs_style_info_t *info 定义如下： int lineSize 线条宽度； uvs_dash_style_e dashStyle 线条样式； COLORREF edgeColor 线条颜色； COLORREF fillColor 填充颜色； int edgeOpacity 线条叠加透明度； int fillOpacity 填充透明度； 其他参数见 uvs_set_time_osd 的相关说明。
返回值	成功返回 UVS_OK，否则失败

53、uvs_set_line_osd

说明	视频叠加直线
参数	int x0 直线起点的水平坐标； int y0 直线起点的垂直坐标； int x1 直线终点的水平坐标； int y1 直线终点的垂直坐标； 结构 const uvs_line_info_t *info 定义如下： int lineSize 线条宽度； uvs_dash_style_e dashStyle 线条样式，包括实线、线段、点、线段和点组合等样式； COLORREF lineColor 线条颜色； int lineOpacity 线条叠加透明度。 其他参数见 uvs_set_time_osd 的相关说明。
返回值	成功返回 UVS_OK，否则失败

54、uvs_osd_control

说明	视频 OSD 控制
参数	uvs_osd_cmd_e cmd OSD 控制命令，需要配合参数 param0 和 param1 使用，不使用的参数将被忽略，可以为 NULL，具体的命令定义如下： uvs_osd_enable 使能 OSD，param0 为 0 表示禁用，为 1 表示启用； uvs_osd_delete 删除 OSD；

	uvs_osd_delete_all 删除所有 OSD； uvs_osd_get_count 获取不同类型的 OSD 个数，param0 返回编码和预览类型的 OSD 个数，param1 返回仅预览的 OSD 个数； uvs_osd_get_size 获取 OSD 的大小，param0 返回宽度，param1 返回高度； uvs_osd_get_position 获取 OSD 的起始位置，param0 返回水平位置，param1 返回垂直位置； uvs_osd_set_position 设置 OSD 的起始位置，param0 为水平位置，param1 为垂直位置； uvs_osd_get_font_size 获取字体大小，param0 返回字体大小； uvs_osd_set_font_size 设置字体大小，param0 为字体大小； uvs_osd_get_color 获取颜色，param0 为前景色，param1 为背景色； uvs_osd_set_color 设置颜色，param0 为前景色，param1 为背景色； uvs_osd_get_opacity 获取透明度，param0 为前景透明度，param1 为背景透明度； uvs_osd_set_opacity 设置透明度，param0 为前景透明度，param1 为背景透明度； uvs_osd_get_clip_position 获取 OSD 裁剪位置，也就是说可以控制 OSD 显示的内容，param0 为水平位置，param1 为垂直位置； uvs_osd_set_clip_position 设置 OSD 裁剪位置，param0 为水平位置，param1 为垂直位置； uvs_osd_get_clip_size 获取 OSD 裁剪尺寸，param0 为宽度，param1 为高度； uvs_osd_set_clip_size 设置 OSD 裁剪尺寸，param0 为宽度，param1 为高度，参数为 0 表示不裁剪 OSD； uvs_osd_get_blink 获取 OSD 显示参数，param0 为显示时间（milliseconds），param1 为消隐时间； uvs_osd_set_blink 设置 OSD 显示参数，param0 为显示时间（milliseconds），param1 为消隐时间，每个周期都包含一次 OSD 的显示和消隐，OSD 将按照周期自动循环，直到该参数被重新设置为 0。 其他参数见 uvs_set_time_osd 的相关说明。
返回值	成功返回 UVS_OK，否则失败

55、uvs_create_frame_preview

说明	创建视频预览，直接显示数据。
参数	HWND hwnd 视频显示的窗口句柄； uvs_disp_type_e dispType 显示渲染器类型，uvs_disp_type_ddraw directdraw，uvs_disp_type_d3d direct3d。
返回值	成功返回 UVS_OK，否则失败

56、uvs_frame_preview

说明	开始视频预览

参数	uvsobj_handle handle 通过接口 uvs_create_frame_preview 返回的句柄； const uvs_frame_info_t *info 视频帧数据。
返回值	成功返回 UVS_OK，否则失败

57、uvs_set_frame_preview_rect

说明	设置视频预览位置
参数	uvsobj_handle handle 通过接口 uvs_create_frame_preview 返回的句柄； LPRECT wndRect 视频显示在窗口中的位置，参考坐标为窗口大小（0, 0, wndWidth, wndHeight），NULL 表示整个窗口； LPRECT cropRect 视频显示裁剪区域，参考坐标为视频大小（0, 0, videoWidth, videoHeight），NULL 表示不裁剪； COLORREF bkColor 窗口空白区域填充的背景色。
返回值	成功返回 UVS_OK，否则失败

58、uvs_destroy_frame_preview

说明	结束视频预览
参数	uvsobj_handle handle 通过接口 uvs_create_frame_preview 返回的句柄。
返回值	成功返回 UVS_OK，否则失败

59、uvs_set_video_raw_frame_callback

说明	设置视频帧数据回调函数
参数	uvsobj_handle obj 对应的设备句柄； pUVSFrameCallback pCB 回调函数指针，采集到视频时将被调用，注意不要进行耗时的操作以免影响性能，值为 NULL 时表示回调停止； void *pUserData 用户数据指针，在回调函数被调用时传递。
返回值	成功返回 UVS_OK，否则失败

60、uvs_set_video_enc_frame_callback

说明	设置视频编码数据回调函数
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号； pUVSFrameCallback pCB 回调函数指针，视频数据编码时将被调用，因此必须先启动视频编码，注意不要进行耗时的操作以免影响性能，值为 NULL 时表示回调停止； void *pUserData 用户数据指针，在回调函数被调用时传递。
返回值	成功返回 UVS_OK，否则失败

61、uvs_set_video_draw_callback

说明	设置视频预览回调函数
参数	uvsobj_handle obj 对应的设备句柄； pUVSDrawCallback pCB 回调函数指针，视频帧数据显示时将被调用，注意不要进行耗时的操作以免影响性能，值为 NULL 时表示回调停止； void *pUserData 用户数据指针，在回调函数被调用时传递。

返回值	成功返回 UVS_OK，否则失败

62、uvs_convert_video_frame

说明	视频数据转换，可以进行裁剪、色彩空间、旋转、缩放转换
参数	const uvs_frame_convert_t *frameConvert 视频转换设置，结构定义如下： uvs_frame_type_e frameType 最终的帧格式，可以是 YUV/RGB 格式； BOOL bMirror 是否水平翻转； BOOL bFlip 是否垂直翻转； int cropLeft 视频裁剪起点的水平坐标； int cropTop 视频裁剪起点的垂直坐标； int cropWidth 视频裁剪宽度，为 0 表示不裁剪； int cropHeight 视频裁剪高度，为 0 表示不裁剪； int scaleWidth 视频缩放宽度，为 0 表示不缩放； int scaleHeight 视频缩放高度，为 0 表示不缩放； uvs_scale_usage_e scaleUsage 视频缩放模式； uvs_rotate_mode_e rotateMode 旋转模式； 视频转换按照以下顺序进行： Video Crop->Mirror/Rotate->Scale->Color Space Convert const uvs_frame_info_t *src 源数据帧信息； uvs_frame_info_t *dst 转换后的数据帧，用户可以指定数据存储方式： videoDataStride[0] 指定数据的对齐大小，为 0 表示按照 4 字节对齐； frameData 指定保存的数据缓冲区，NULL 表示使用内部缓冲，由于内部缓冲区是固定的，使用内部缓冲时不能释放对应的指针，如果需要同时转换多个视频数据，用户需要为每个数据自行分配内存空间，因为每次转换都会导致内部缓冲的数据被更新； frameDataLen 用户数据缓冲大小，用户自行分配内存时必须指定大小。
返回值	成功返回 UVS_OK，否则失败

63、uvs_save_video_frame

说明	视频帧保存为文件
参数	const char *szFileName 需要保存的文件名，文件名后缀表示对应的文件类型，支持的类型如下： .bmp bmp 位图文件； .jpg JPG 压缩格式； .png PNG 格式； const uvs_frame_convert_t *frameConvert 数据转换格式，参见 uvs_convert_video_frame 相关说明； const uvs_frame_info_t *info 视频帧数据； UINT jpgQuality JPG 保存质量，取值为 0 ~ 100，值越大质量越好。
返回值	成功返回 UVS_OK，否则失败

64、uvs_save_video_snapshot

说明	截图保存为文件
参数	uvsobj_handle obj 对应的设备句柄； const uvs_frame_convert_t *frameConvert 数据转换格式，参见 uvs_convert_video_frame 相关说明； const char *szFileName 需要保存的文件名，参见 uvs_save_video_frame 的相关说明； UINT jpgQuality JPG 保存质量，参见 uvs_save_video_frame 的相关说明。 该接口的实现等同于 uvs_copy_video_frame 加 uvs_save_video_frame。
返回值	成功返回 UVS_OK，否则失败

65、uvs_lock_video_raw_frame

说明	锁定视频帧数据
参数	uvsobj_handle obj 对应的设备句柄； uvs_frame_info_t *info 锁定的视频帧数据； UINT waitMilliSec 以毫秒为单位的等待时间，等待超时将返回，为 0 表示始终等待直至获得数据，如果当前无视频信号将会导致阻塞。 视频帧锁定成功后需及时解锁，注意不要进行耗时的操作以免影响性能，存在耗时的操作请使用 uvs_copy_video_frame 拷贝数据。
返回值	成功返回 UVS_OK，否则失败

66、uvs_unlock_video_raw_frame

说明	解锁视频帧数据，锁定数据成功后必须解锁
参数	uvsobj_handle obj 对应的设备句柄；
返回值	成功返回 UVS_OK，否则失败

67、uvs_copy_video_frame

说明	拷贝视频帧数据
参数	uvsobj_handle obj 对应的设备句柄； const uvs_frame_convert_t *frameConvert 数据转换格式，参见 uvs_convert_video_frame 相关说明； uvs_frame_info_t *info 拷贝的视频帧数据，用户可以指定数据存储方式： videoDataStride[0] 指定数据的对齐大小，为 0 表示按照 4 字节对齐； frameData 指定保存的数据缓冲区，NULL 表示使用内部缓冲，由于内部缓冲区是固定的，使用内部缓冲时不能释放对应的指针，如果需要同时转换多个视频数据，用户需要为每个数据自行分配内存空间，因为每次转换都会导致内部缓冲的数据被更新； frameDataLen 用户数据缓冲大小，用户自行分配内存时必须指定大小。 UINT waitMilliSec 以毫秒为单位的等待时间，等待超时将返回，为 0 表示

	始终等待直至获得数据，如果当前无视频信号将会导致阻塞。
返回值	成功返回 UVS_OK，否则失败

68、uvs_set_audio_raw_frame_callback

说明	设置音频帧数据回调
参数	uvsobj_handle obj 对应的设备句柄； pUVSFrameCallback pCB 回调函数指针，采集到音频时将被调用，注意不要进行耗时的操作以免影响性能，值为 NULL 时表示回调停止； void *pUserData 用户数据指针，在回调函数被调用时传递。
返回值	成功返回 UVS_OK，否则失败

69、uvs_set_audio_enc_frame_callback

说明	设置音频编码数据回调
参数	uvsobj_handle obj 对应的设备句柄； pUVSFrameCallback pCB 回调函数指针，音频数据编码时将被调用，因此必须先启动音频编码，注意不要进行耗时的操作以免影响性能，值为 NULL 时表示回调停止； void *pUserData 用户数据指针，在回调函数被调用时传递。
返回值	成功返回 UVS_OK，否则失败

70、uvs_lock_audio_raw_frame

说明	锁定音频帧数据
参数	uvsobj_handle obj 对应的设备句柄； uvs_frame_info_t *info 锁定的音频帧数据； UINT waitMilliSec 以毫秒为单位的等待时间，等待超时将返回，为 0 表示始终等待直至获得数据。 音频帧锁定成功后需及时解锁，注意不要进行耗时的操作以免影响性能，存在耗时的操作请使用 uvs_copy_audio_raw_frame 拷贝数据。
返回值	成功返回 UVS_OK，否则失败

71、uvs_unlock_audio_raw_frame

说明	解锁音频帧数据，锁定数据成功后必须解锁
参数	uvsobj_handle obj 对应的设备句柄；
返回值	成功返回 UVS_OK，否则失败

72、uvs_copy_audio_raw_frame

说明	拷贝音频帧数据
参数	uvsobj_handle obj 对应的设备句柄； uvs_frame_info_t *info 拷贝的音频帧数据，用户需要分配数据缓冲区，参数 frameData 指定保存的数据缓冲区地址，参数 frameDataLen 为用户数据缓冲大小，建议大小为 64K 字节； UINT waitMilliSec 以毫秒为单位的等待时间，等待超时将返回，为 0 表示始终等待直至获得数据。

返回值	成功返回 UVS_OK，否则失败

73、uvs_media_server_rtsp_create

说明	创建 rtsp 服务，启动 rtsp 媒体流之前必须先创建服务
参数	USHORT rtspPort rtsp 服务端口号，通常使用 554 或 8554； USHORT httpTunnerPort http 透传端口，rtsp 接收端也需要同步设置，为 0 表示不使用； const char *szUserName rtsp 权限控制，用户名，为 NULL 表示不使用； const char *szPassword 权限控制，账户密码，为 NULL 表示不使用。
返回值	成功返回 UVS_OK，否则失败，例如 rtsp 端口被占用或端口未关闭，可以等待一段时间重试。

74、uvs_media_server_rtsp_destroy

说明	结束 rtsp 服务
参数	USHORT rtspPort rtsp 服务的端口号。
返回值	成功返回 UVS_OK，否则失败

75、uvs_media_stream_rtsp_start

说明	启动 rtsp 流，必须先调用 uvs_media_server_rtsp_create 创建 rtsp 服务
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，每个设备可启动多个编码流； USHORT rtspPort rtsp 服务端口号； BOOL bHasAudio 是否启动音频； const char *szSessionName rtsp 会话名称，将决定 rtsp url； rtp_multicast_t *rtpMulticast rtsp 组播设置，定义如下： ipAddress 组播地址； ipPort 组播端口号。
返回值	成功返回 UVS_OK，否则失败

76、uvs_media_stream_rtsp_stop

说明	停止 rtsp 流
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，每个设备可启动多个编码流；
返回值	成功返回 UVS_OK，否则失败

77、uvs_media_stream_rtsp_get_url

说明	获取 rtsp 流的 url 信息
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，每个设备可启动多个编码流； uvs_url_info_t *urlInfo 参数类型为[OUT]，返回 URL 信息。
返回值	成功返回 UVS_OK，否则失败

78、uvs_media_file_rtsp_start

说明	启动 rtsp 流媒体文件，必须先调用 uvs_media_server_rtsp_create 创建 rtsp 服务
参数	const char *szFileName MP4 录像文件路径； USHORT rtspPort rtsp 服务端口号； BOOL bHasAudio 是否启动音频； const char *szSessionName rtsp 会话名称，将决定 rtsp url。
返回值	成功返回 UVS_OK，否则失败

79、uvs_media_file_rtsp_stop

说明	停止 rtsp 流媒体文件
参数	const char *szFileName MP4 录像文件路径。
返回值	成功返回 UVS_OK，否则失败

80、uvs_media_file_rtsp_get_url

说明	获取 rtsp 文件流的 url 信息
参数	const char *szFileName 录像文件路径； uvs_url_info_t *urlInfo 参数类型为[OUT]，返回 URL 信息。
返回值	成功返回 UVS_OK，否则失败

81、uvs_media_stream_rtmp_send

说明	启动 rtmp 推流
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，每个设备可启动多个编码流； const char *rtmpUrl rtmp 推流的 url； BOOL bHasAudio 是否启动音频。
返回值	成功返回 UVS_OK，否则失败

82、uvs_media_stream_rtmp_stop

说明	停止 rtmp 推流
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，每个设备可启动多个编码流。
返回值	成功返回 UVS_OK，否则失败

83、uvs_media_stream_rtmp_get_send_url

说明	获取 rtmp 推流的 url 信息
参数	uvsobj_handle obj 对应的设备句柄； int streamIndex 编码流索引号，每个设备可启动多个编码流； uvs_url_info_t *urlInfo 参数类型为[OUT]，返回 URL 信息。
返回值	成功返回 UVS_OK，否则失败

84、uvs_media_file_rtmp_send

说明	启动录像文件 rtmp 推流
参数	const char *szFileName MP4 录像文件路径；

	const char *rtmpUrl rtmp 推流的 url； BOOL bHasAudio 是否启动音频。
返回值	成功返回 UVS_OK，否则失败

85、uvs_media_file_rtmp_get_progress

说明	获取录像文件推流的进度
参数	const char *szFileName MP4 录像文件路径；
返回值	返回推流的进度[0, 100]，轮询该接口直到 100 表示成功完成，否则失败

86、uvs_media_file_rtmp_stop

说明	停止 rtmp 推流
参数	const char *szFileName MP4 录像文件路径。
返回值	成功返回 UVS_OK，否则失败

87、uvs_media_file_rtmp_get_send_url

说明	获取 rtmp 推流的 url 信息
参数	const char *szFileName MP4 录像文件路； uvs_url_info_t *urlInfo 参数类型为[OUT]，返回 URL 信息。
返回值	成功返回 UVS_OK，否则失败

88、uvs_media_stream_ndi_create

说明	创建 NDI 推流
参数	uvsobj_handle obj 对应的设备句柄； const char *szSourceName NDI source 名字； const char *szGroupName NDI Group 名字； const char *szXMLMetadata NDI 流 xml 元数据（字符串为 UTF8 格式）； BOOL bAsync 是否需要创建发送线程的异步模式。
返回值	成功返回 UVS_OK，否则失败

89、uvs_media_stream_ndi_config

说明	实时 NDI 推流配置
参数	uvsobj_handle obj 对应的设备句柄； const uvs_frame_convert_t *frameConvert 视频帧转换，参见 uvs_convert_video_frame 相关说明； int skipFrameRate 帧间隔控制，0 表示发送所有视频帧，1 表示间隔 1 帧发送（1/2），2 表示间隔 2 帧发送（1/3）...
返回值	成功返回 UVS_OK，否则失败

90、uvs_media_stream_ndi_start

说明	启动实时 NDI 推流
参数	uvsobj_handle obj 对应的设备句柄； BOOL bHasAudio 是否启动音频流。
返回值	成功返回 UVS_OK，否则失败

91、uvs_media_stream_ndi_stop

说明	停止 NDI 推流
参数	uvsobj_handle obj 对应的设备句柄。
返回值	成功返回 UVS_OK，否则失败

92、uvs_media_stream_ndi_destroy

说明	结束 NDI 推流
参数	uvsobj_handle obj 对应的设备句柄。
返回值	成功返回 UVS_OK，否则失败

93、uvs_mp4_file_get_status

说明	获取 MP4 录像文件状态
参数	const char *szFileName MP4 录像文件路径； uvs_mp4_file_status_t *status 参数类型为[OUT]，返回对应的文件信息： uvs_audio_codec_e audioCodec 音频编码类型，MP3 或 AAC； uvs_video_codec_e videoCodec 视频编码类型，H.264、265 等； int audioDuration 音频流时长（毫秒）； int videoDuration 视频流时长（毫秒）； int videoWidth 视频宽度； int videoHeight 视频高度； float videoFrameRate 视频帧率。
返回值	成功返回 UVS_OK，否则失败

94、uvs_mp4_file_repair

说明	修复中断的 MP4 录像文件，另存为新录像文件
参数	const char *szFileName MP4 录像文件路径； const char *szOutputFile 保存修复的 MP4 新录像文件路径。
返回值	成功返回 UVS_OK，否则失败

95、uvs_mp4_file_repair_cancel

说明	取消录像文件修复
参数	无
返回值	成功返回 UVS_OK，否则失败

96、uvs_mp4_file_get_repair_progress

说明	获取 MP4 录像文件修复的进度
参数	int *progress 参数类型为[OUT]，返回修复的进度[0, 100]，轮询该接口直到 100 表示成功完成，否则失败。
返回值	成功返回 UVS_OK，否则失败

97、uvs_mp4_file_cut

说明	剪切 MP4 录像文件，另存为新录像文件

参数	const char *szFileName MP4 录像文件路径； int beginMilliSec 文件剪切的起始时间位置（毫秒）； int endMilliSec 文件剪切的结束时间位置（毫秒）； const char *szOutputFile 保存 MP4 新录像文件的路径。
返回值	成功返回 UVS_OK，否则失败

98、uvs_mp4_file_get_cut_progress

说明	获取录像文件剪切进度
参数	int *progress 参数类型为[OUT]，返回修复的进度[0, 100]，轮询该接口直到 100 表示成功完成，否则失败。
返回值	成功返回 UVS_OK，否则失败

99、uvs_mp4_file_add_title

说明	为录像文件叠加 Title，包括文字或图片，另存为新录像文件
参数	const char *szFileName MP4 录像文件路径； int beginMilliSec 叠加的起始时间位置（毫秒）； const uvs_mp4_file_ title_t title[] 叠加 Title 的信息，定义如下： int x, y 叠加 Title 的起点坐标； int width, height 叠加 Title 的宽度和高度； int imageOpacity 叠加图片的透明度[0, 255]； uvs_font_info_t textFont 文字显示的字体信息，参见 uvs_set_time_osd 的相关说明； const char *szTitleImageFile 叠加图片的文件路径，可以是 bmp、jpg 或 png 文件，为 NULL 表示不叠加图片； const char *szTitleText 叠加文本内容，为 NULL 表示不叠加文本，szTitleImageFile 和 szTitleText 不能同时为 NULL。 int titleNum title 数组的个数； int titleMilliSec title 持续的时长（毫秒）； const char *szOutputFile 保存 MP4 新录像文件的路径。
返回值	成功返回 UVS_OK，否则失败

100、uvs_mp4_file_get_title_progress

说明	获取 Title 叠加的进度
参数	int *progress 参数类型为[OUT]，返回修复的进度[0, 100]，轮询该接口直到 100 表示成功完成，否则失败。
返回值	成功返回 UVS_OK，否则失败

101、uvs_mp4_file_merge

说明	合并 MP4 录像文件
参数	const char *szFileName[] 需要合并的录像文件路径数组； int fileNum 录像文件数组的个数； const char *szOutputFile 保存 MP4 新录像文件的路径。
返回值	成功返回 UVS_OK，否则失败

102、uvs_mp4_file_get_merge_progress

说明	获取录像文件合并的进度
参数	int *progress 参数类型为[OUT]，返回修复的进度[0, 100]，轮询该接口直到 100 表示成功完成，否则失败。
返回值	成功返回 UVS_OK，否则失败