**云存储产品线接口文档**

**V02.00.00**

**公司：深圳市泽宝网络科技有限公司**

**2019年4月29**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 文档版本号 | V02.00.00 | | | 保密等级 | 机密 |
| 概要需求文档版本号 | V02.00.00 | | | | |
| 产品文档版本号 | V02.00.00 | | | | |
| 测试用例文档版本号 |  | | | | |
| 代码版本号 | V2.1.0.0 | | | | |
| 产品线 | VAVA云存储 | | 归属部门/项目 | | IOT |
| 编 制 | dawson | 审 核 |  | 批 准 |  |
| 日 期 | 19-04-29 | 日 期 |  | 日 期 |  |

修改历史

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **日期** | **3说明** | **责任人** |
| 1 | 2019-04-29 | 创建初稿 | dawson |
| 2 | 2019-05-20 | 评审修改 | dawson |
| 3 | 2020-10-12 | V2.0.0版本接口变更 | Dawson |
|  |  |  |  |

目录

[1. 概述 5](#_Toc54009394)

[2. 接口 5](#_Toc54009395)

[2.1. SDK初始化接口 5](#_Toc54009396)

[2.2. 日志初始化接口 5](#_Toc54009397)

[2.2.1. 日志等级枚举(E\_SV\_LOG\_LEVEL)类型定义 5](#_Toc54009398)

[2.2.2. 日志输出方式 6](#_Toc54009399)

[2.3. 设置TOKEN获取接口 6](#_Toc54009400)

[2.4. SDK反初始化接口 6](#_Toc54009401)

[2.5. 连接推流服务器 6](#_Toc54009402)

[2.5.1. 消息回调接口 7](#_Toc54009403)

[2.5.2. 回调通知事件枚举(E\_SV\_EVENT\_TYPE)类型定义 7](#_Toc54009404)

[2.6. 连接推流服务器 8](#_Toc54009405)

[2.7. 连接推流服务器 8](#_Toc54009406)

[2.8. 直播连接推流服务器 9](#_Toc54009407)

[2.9. 发送音视频数据 9](#_Toc54009408)

[2.9.1. 媒体数据编码类型枚举(E\_SV\_MEDIA\_TYPE)类型定义 9](#_Toc54009409)

[2.10. 发送元数据 10](#_Toc54009410)

[2.10.1. 元数据类型枚举(E\_SV\_METADATA\_TYPE)类型定义 10](#_Toc54009411)

[2.10.2. 媒体数据加密类型枚举 10](#_Toc54009412)

[2.11. 推流连接是否正常连接 11](#_Toc54009413)

[2.12. 关闭推流连接 12](#_Toc54009414)

[2.13. 获取推流SDK版本信息 12](#_Toc54009415)

[3. 接入演示代码 12](#_Toc54009416)

# 概述

Cloud Push SDK是泽宝网络科技有限公司(Sunvalley) IOT事业部云存储项目组开发的IOT嵌入式设备推流sdk, 该sdk旨在为能够采集音视频媒体数据的联网设备提供高效稳定地将媒体数据推送到Sunvalley云端，并为用户提供便捷的保存和点播，直播等服务。此SDK将对外开放，以方便其他IOT终端快速便捷地接入Sunvalley并提供的云服务功能。

# 接口

## SDK初始化接口

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Initialize |
| 2 | 接口说明 | 初始化SDK全局变量，在sdk加载后调用 |
| 3 | 参数1[input] | int memory\_pool\_size内存缓冲池大小 |
| 4 | 参数2[input] | int Max\_conn\_cont 最大连接数 |
| 4 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

## 日志初始化接口

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Init\_log |
| 2 | 接口说明 | 初始化日志输出方式、输出等级以及文件日志输出目录 |
| 3 | 参数1 [input] | E\_SV\_LOG\_LEVEL eLogLevel:日志等级，详细定义见2.2.1 |
| 4 | 参数2 [input] | unsigned int nLogFlag：日志输出方式详见2.2.2 |
| 5 | 参数3 [input] | const char\* pLogPath:文件日志输出目录，输出文件日志时此参数不能为NULL |
| 6 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

### 日志等级枚举(E\_SV\_LOG\_LEVEL)类型定义

| **No** | **枚举定义** | **枚举值** | **描述** |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | E\_LOG\_LEVEL\_VERB | 0x01 | 输出最详细日志(级别>= 0x01) |
| 2 | E\_LOG\_LEVEL\_INFO | 0x02 | 输出信息以上级别的日志(级别>= 0x02) |
| 3 | E\_LOG\_LEVEL\_MAIN | 0x03 | 输出主要信息以上级别日志(级别>= 0x03则输出) |
| 4 | E\_LOG\_LEVEL\_WARN | 0x04 | 输出警告以上级别日志(级别>= 0x04) |
| 5 | E\_LOG\_LEVEL\_ERROR | 0x5 | 输出错误级别日志(级别>= 0x05s)s |
| 6 | E\_LOG\_LEVEL\_DISABLE | 0x06 | 不输出日志 |

### 日志输出方式

| **No** | **宏定义** | **枚举值** | **描述** |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | SV\_LOG\_OUTPUT\_MODE\_NONE | 0x00 | 不输出日志 |
| 2 | SV\_LOG\_OUTPUT\_MODE\_CONSOLE | 0x01 | 输出控制台日志 |
| 3 | SV\_LOG\_OUTPUT\_MODE\_FILE | 0x02 | 输出文件日志 |

eg:日志支持同时输出到控制台和文件,设置方式为：

int nLogFlag = SV\_LOG\_OUTPUT\_MODE\_CONSOLE | SV\_LOG\_OUTPUT\_MODE\_FILE;

## 设置TOKEN获取接口

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVPush\_API\_Set\_Token\_Serve\_Interface |
| 2 | 接口说明 | SDK初始化之后设置token获取接口 |
| 3 | 参数[input] | ptoken\_server\_url |
| 4 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

## SDK反初始化接口

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Uninitialize |
| 2 | 接口说明 | 销毁SDK全局变量，在sdk卸载之前调用 |
| 3 | 参数 | 无 |
| 4 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

## 连接推流服务器

| **No** |  | **信息** | **描述** |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 |  | 接口名称 | SVCloudPush\_API\_Connect |
| 2 |  | 接口说明 | 连接推流服务器，开始推流 |
| 3 |  | 参数1 [input] | Const char\* Url: 推流url |
| 4 |  | 参数2[input] | Const char\* ptoken |
| 5 |  | 参数3 [input, output] | event\_callback pcbFun：消息回调通知接口，详见2.4.1 |
| 6 |  | 返回值(int) | 成功返回0，失败返回非0值(-1) |

补充URL规则定义：rtmp://host:port/$appname/$streamname/$time

### 消息回调接口

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | event\_callback |
| 2 | 接口说明 | sdk推流回调消息通知接口，由app调用层设置 |
| 3 | 参数1 [input] | long lcid： SVCloudPush\_API\_Connect返回的connectid |
| 4 | 参数2 [input] | E\_SV\_EVENT\_TYPE eventType：回调事件类型，详见2.4.2 |
| 5 | 参数3 [input] | long wparam：辅助说明参数，通常为错误码 |
| 6 | 参数4 [input] | long lparam：消息值 |
| 4 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

### 回调通知事件枚举(E\_SV\_EVENT\_TYPE)类型定义

| **No** | **枚举定义** | | **枚举值** | **描述** | **wparm** | **lparam** |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |
| 1 | | E\_SV\_EVENT\_TYPE\_NONE | 0x0 | 未知事件类型， | 0 | 0 |
| 2 | | E\_SV\_EVENT\_TYPE\_RTMP\_INVALID\_URL | 0x01 | 无效的推流url | error code | 0 |
| 3 | | E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_DNS\_RESOVLE\_FAILED | 0x02 | dns域名解析失败 | error code | 0 |
| 4 | | E\_SV\_EVENT\_TYPE\_RTMP\_SOCKET\_CONNECT\_FAILED | 0x03 | 推流连接时socket创建连接失败 | error code | 0 |
| 5 | | E\_SV\_EVENT\_TYPE\_RTMP\_HANDSHAKE\_FAILED | 0x4 | rtmp握手失败 | error code | 0 |
| 6 | | E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_APP\_FAILED | 0x05 | Rtmp连接app失败 | error code | 0 |
| 7 | | E\_SV\_EVENT\_TYPE\_RTMP\_PUBLISH\_STREAM\_FAILED | 0x06 | rtmp发布流失败 | error code | 0 |
| 8 | | E\_SV\_EVENT\_TYPE\_RTMP\_SET\_TIMEOUT\_FAILED | 0x07 | rtmp设置超时失败 | error code | 0 |
| 9 | | E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_CLOSE | 0x08 | rtmp推流结束 | 0 | 0 |
| 10 | | E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_SUCCESS | 0x09 | rtmp推流连接成功 | 0 | 0 |
| 11 | | E\_SV\_EVENT\_TYPE\_RTMP\_INVALID\_PARAMETER | 0x50 | 无效的参数 | error code | 0 |
| 12 | | E\_SV\_EVENT\_TYPE\_RTMP\_AUDIO\_ADTS\_DATA\_ERROR | 0x51 | 音频aac数据adts头错误 | error code | 0 |
| 13 | | E\_SV\_EVENT\_TYPE\_RTMP\_VIDEO\_DATA\_ERROR | 0x52 | 视频数据错误 | error code | 0 |
| 14 | | E\_SV\_EVENT\_TYPE\_RTMP\_AUDIO\_DATA\_ERROR | 0x53 | 音频数据错误 | error code | 0 |
| 15 | | E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_VIDEO\_NO\_SPS\_PPS | 0x54 | 视频h264/h265数据没有sps、pps | 0 | 0 |
| 16 | | E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_ERROR\_SPS\_PPS | 0x55 | 视频h264/h265错误的SPS/PPS | 0 | 0 |
| 17 | | E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_ERROR\_SPS | 0x56 | SPS数据错误 | 0 | 0 |
| 18 | | E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_ERROR\_PPS | 0x57 | PPS数据错误 | 0 | 0 |
| 19 | | E\_SV\_EVENT\_TYPE\_RTMP\_RESUME\_MEDIA\_PUSH | 0x58 | Rtmp恢复推流 | 0 | 0 |
| 20 | | E\_SV\_EVENT\_TYPE\_RTMP\_PAUSE\_MEDIA\_PUSH | 0x59 | rtmp暂停推流（网络拥塞，缓冲已满） | 0 | 0 |
| 21 | | E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_THREAD\_EXIT\_WITH\_ERROR | 0x60 | Rtmp socket发送线程因错误退出 | error code | 0 |
| 22 | | E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_ERROR\_VPS | 0x61 | VPS数据错误 | error code | 0 |
| 23 | | E\_SV\_EVENT\_TYPE\_PUSH\_SEGMENT\_END | 0x102 | 分片推流结束 | 0 | 0 |

## 连接推流服务器

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Connect\_By\_D2eviceSN |
| 2 | 接口说明 | 连接推流服务器，开始推流 |
| 3 | 参数1 [input] | Const char\* pclientid: 与设备绑定的clientid |
| 4 | 参数2[input] | Const char\*pclient\_secret:：与设备绑定的客户端密钥 |
| 5 | 参数3[input] | Char\* pdeviceSN：设备端序列号 |
| 6 | 参数3 [input, output] | event\_callback pcbFun：消息回调通知接口，详见2.4.1 |
| 7 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

## 连接推流服务器

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Connect\_By\_Device\_Token |
| 2 | 接口说明 | 根据设备端口令连接推流服务器 |
| 3 | 参数1 [input] | Const char\* pdevice\_token: 设备口令 |
| 4 | 参数2 [input, output] | event\_callbackt pcbFun：消息回调通知接口，详见2.4.1 |
| 5 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

## 直播连接推流服务器

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Live\_Connect |
| 2 | 接口说明 | 根据设备端口令连接推流服务器 |
| 3 | 参数1 [input] | Const char\* ptoken\_url: 设备口令 |
| 4 | 参数2 [input] | Const char\* pdeviceSN：设备SN |
| 4 | 参数3 [input, output] | event\_callbackt pcbFun：消息回调通知接口，详见2.4.1 |
| 5 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

## 发送音视频数据

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Send\_Packet |
| 2 | 接口说明 | 发送音视频数据包 |
| 3 | 参数1 [input] | long lcid： SVCloudPush\_API\_Connect返回的connectid |
| 4 | 参数2 [input] | E\_SV\_MEDIA\_TYPE，媒体数据类型，详见2.5.1 |
| 5 | 参数3 [input] | char\* pData：媒体数据缓存指针 |
| 6 | 参数4 [input] | Int nSize：媒体数据缓存大小 |
| 7 | 参数5 [input] | Unsigned int usTimeStamp：媒体数据时间戳 |
| 4 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

### 媒体数据编码类型枚举(E\_SV\_MEDIA\_TYPE)类型定义

| **No** | **枚举定义** | **枚举值** | **描述** |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | E\_SV\_MEDIA\_TYPE\_UNKNOW | -1 | 未知媒体编码类型 |
| 2 | E\_SV\_MEDIA\_TYPE\_H264 | 0 | 视频-h264编码类型（带起始码） |
| 3 | E\_SV\_MEDIA\_TYPE\_H265 | 1 | 视频-h265编码类型（带起始码） |
| 4 | E\_SV\_MEDIA\_TYPE\_MP3 | 2 | 音频-mp3编码类型 |
| 5 | E\_SV\_MEDIA\_TYPE\_AAC | 3 | 音频-AAC编码类型（带adts头） |
| 6 | E\_SV\_MEDIA\_TYPE\_PRI\_DATA | 4 | 私有媒体数据类型 |

## 发送元数据

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Send\_Metadata |
| 2 | 接口说明 | 发送元数据 |
| 3 | 参数1 [input] | long lcid： SVCloudPush\_API\_Connect返回的connectid |
| 4 | 参数2 [input] | E\_SV\_METADATA\_TYPE，元数据类型，详见2.6.1 |
| 5 | 参数3 [input] | wParam：元数据辅助说明参数 |
| 6 | 参数4 [input] | lParam：元数据消息值 |
| 4 | 返回值(int) | 成功返回0，失败返回非0值(-1) |

### 元数据类型枚举(E\_SV\_METADATA\_TYPE)类型定义

| **No** | **枚举定义** | **枚举值** | **描述** | **wparam** | **lparam** |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 1 | E\_SV\_METADATA \_TYPE\_NONE | -1 | 未知元数据类型 | 0 | 0 |
| 2 | E\_SV\_METADATA \_TYPE\_STREAM\_START | 0 | 流开始 | Tigger type | 流触发本地时间戳取地址(int64\_t\*) |
| 3 | E\_SV\_METADATA \_TYPE\_STREAM\_END | 1 | 流结束 | 告警时间（毫秒/ms） | 流停止本地时间戳取地址(int64\_t\*) |
| 4 | E\_SV\_METADATA \_TYPE\_VIDEO\_ENCRYPT | 2 | 视频加密 | 加密方式，详见2.10.2定义 | 加密密钥指针 |
| 5 | E\_SV\_METADATA \_TYPE\_AUDIO\_ENCRYPT | 3 | 音频加密 | 加密方式，详见2.10.2定义 | 加密密钥指针 |
| 6 | E\_SV\_METADATA \_TYPE\_STREAM\_BITRATE | 4 | 媒体码流的码率 | 音频码率 | 视频码率 |
| 7 | E\_SV\_METADATA \_TYPE\_STREAM\_DROP\_FRAMEss | 5 | 丢帧数 | 音频丢帧数 | 视频丢帧数 |
| 8 | E\_SV\_METADATA \_TYPE\_STREAM\_DISCONNECT | 6 | app与camera失去连接 | error code | 0 |
| 9 | E\_SV\_METADATA \_TYPE\_TIGGER\_TYPE\_CHANGE | 7 | 触发类型改变消息 | 改变之前的触发类型 | 改变之后的触发类型 |

### 媒体数据加密类型枚举

| **No** | **枚举定义** | **枚举值** | **描述** |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | E\_SV\_MEDIA \_ENCRYPT\_NONE | 0 | 不加密 |
| 2 | E\_SV\_MEDIA \_ENCRYPT\_AES\_KEYFRAME | 1 | AES加密关键帧/I帧 |
| 3 | E\_SV\_MEDIA \_ENCRYPT\_ALL\_FRAME | 2 | AES加密所有帧 |

推流触发类型枚举

| **No** | **枚举定义** | **枚举值** | **描述** |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | E\_IPC\_TIGGER\_TYPE\_UNKNOW | -1 | 未知的推流触发类型 |
| 2 | E\_IPC\_TIGGER\_TYPE\_HUMAN | 0 | 人脸识别触发推流 |
| 3 | E\_IPC\_TIGGER\_TYPE\_DEFAULT | 1 | 默认的触发类型（IPC HS002默认为移动侦测触发，HS003/HS004默认为人脸触发） |
| 4 | E\_IPC\_TIGGER\_TYPE\_FACE | 2 | 人脸识别触发 |
| 5 | E\_IPC\_TIGGER\_TYPE\_HUMAN\_AND\_FACE | 3 | 人形加人脸触发 |
| 6 | E\_IPC\_TIGGER\_TYPE\_MOVE\_DETECT | 4 | 移动侦测触发 |
| 7 | E\_IPC\_TIGGER\_TYPE\_RESERVED | 5 | 5-99为保留触发类型，以便后续扩展 |
| 8 | E\_IPC\_TIGGER\_TYPE\_LIVE | 100 | 调试工具触发直播 |
| 9 | E\_IPC\_TIGGER\_TYPE\_IOS\_APP | 101 | IOS app触发直播 |
| 10 | E\_IPC\_TIGGER\_TYPE\_ANDROID\_APP | 102 | Android app触发直播 |
| 11 | E\_IPC\_TIGGER\_TYPE\_ALEXA | 103 | Alexa触发直播 |
| 12 | E\_IPC\_TIGGER\_TYPE\_WEB | 104 | Web触发直播 |
| 13 | E\_IPC\_TIGGER\_TYPE\_GOOGLE\_ASSISTANT | 105 | Google assintant触发直播 |

## 推流连接是否正常连接

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Is\_Connect |
| 2 | 接口说明 | 推流是否正常连接 |
| 3 | 参数1 [input] | lsong lcid： SVCloudPush\_API\_Conn2ect返回的connectid |
| 4 | 返回值(int) | 连接正常返回1,连接已断开返回0 |

## 关闭推流连接

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Close |
| 2 | 接口说明 | 关闭推流连接 |
| 3 | 参数1 [input] | long lcid： SVCloudPush\_API\_Connect返回的connectid |
| 4 | 返回值(int) | 连接正常返回0，失败返回非0值(-1) |

## 获取推流SDK版本信息

| **No** | **信息** | **描述** |
| --- | --- | --- |
|  |  |  |
| 1 | 接口名称 | SVCloudPush\_API\_Version |
| 2 | 接口说明 | 获取推流SDK版本信息 |
| 3 | 参数1 [output] | char\* szVersionXml：返回当前SDK的版本信息 |
| 4 | 参数2 [input] | int len：szVersionXml的缓存长度 |
| 5 | 返回值(int) | 返回当前版本信息的长度 |

# 接入演示代码

#include <stdio.h>

#include <vector>

#include <signal.h>

#include <assert.h>

#include <time.h>

#include <unistd.h>

#include "../common/isvpush.h"

#define MAX\_CHANNEL\_NUM         20

#define TOKEN\_SERVER\_URL        "http://10.30.0.200:2118/connection/token/get"

#define CHECK\_RESULT(ret) if(ret < 0) {assert(0); return ret;}

#define TEST\_DEVICE\_SN      "P020101000201201012400001"

bool g\_brunning = false;

void sig\_stop(int signo)

{

    printf("sig\_stop: signo = %d\n", signo);

    g\_brunning = false;

    return;

}

static int event\_callback(long lCID, long eventType, long wparam, long lparam)

{

CAutoLock lock(m\_crisec);

std::map<long, CPushTask\*>::iterator it = m\_mconnect\_task\_list.find(lCID);

if(it == m\_mconnect\_task\_list.end())

{

printf("Invalid connect task id:%ld eventType:%ld, wparam:%ld, lparam:%ld\n", lCID, eventType, wparam, lparam);

//assert(0);

return -1;

}

CPushTask\* ptask = it->second;

switch(eventType)

{

case E\_SV\_EVENT\_TYPE\_RTMP\_INVALID\_URL:

case E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_DNS\_RESOVLE\_FAILED:

case E\_SV\_EVENT\_TYPE\_RTMP\_SOCKET\_CONNECT\_FAILED:

case E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_HANDSHAKE\_FAILED:

case E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_APP\_FAILED:

case E\_SV\_EVENT\_TYPE\_RTMP\_PUBLISH\_STREAM\_FAILED:

case E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_SET\_TIMEOUT\_FAILED:

case E\_SV\_EVENT\_TYPE\_RTMP\_INVALID\_PARAMETER:

{

printf("lcid:%ld connect error, errorcode:%d\n", lCID, eventType);

ptask->m\_bconnected = false;

return 0;

}

case E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_CLOSE:

{

printf("lcid:%ld connection close\n", lCID);

ptask->m\_bconnected = false;

return 0;

}

case E\_SV\_EVENT\_TYPE\_RTMP\_CONNECT\_SUCCESS:

{

printf("lcid:%ld connection success\n", lCID);

return 0;

}

case E\_SV\_EVENT\_TYPE\_RTMP\_AUDIO\_ADTS\_DATA\_ERROR:

case E\_SV\_EVENT\_TYPE\_RTMP\_VIDEO\_START\_CODE\_ERROR:

case E\_SV\_EVENT\_TYPE\_RTMP\_RECV\_AUDIO\_DATA\_ERROR:

case E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_VIDEO\_NO\_SPS\_PPS:

case E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_ERROR\_SPS\_PPS:

case E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_ERROR\_SPS:

case E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_ERROR\_PPS:

{

printf("lcid:%ld connection error, errorcode:%d\n", lCID, eventType);

return 0;

}

case E\_SV\_EVENT\_TYPE\_RTMP\_RESUME\_MEDIA\_PUSH:

{

printf("lcid:%ld connection resume media push\n", lCID);

return 0;

}

case E\_SV\_EVENT\_TYPE\_RTMP\_PAUSE\_MEDIA\_PUSH:

{

printf("lcid:%ld connection pause media push\n", lCID);

return 0;

}

case E\_SV\_EVENT\_TYPE\_RTMP\_SEND\_THREAD\_EXIT\_WITH\_ERROR:

{

printf("lcid:%ld connection error, push thread exit\n", lCID);

ptask->m\_bconnected = false;

return 0;

}

case E\_SV\_EVENT\_TYPE\_ECHOSHOW\_NOTIFY\_LIVE\_OPEN:

{

g\_brunning = false;

return 0;

}

case E\_SV\_EVENT\_TYPE\_ECHOSHOW\_NOTIFY\_LIVE\_CLOSE:

{

g\_brunning = false;

return 0;

}

default:

{

printf("lcid:%ld connection unknown error, errorcode:%ld\n", lCID, eventType);

return 0;

}

}

}

int main(int argc, char\*\* args)

{

    char ver[256];

    struct timeval tv;

    int ret = 0;

    long lcid = -1;

    char\* pkeystring = "vavakey";

    char\* pvideodata = NULL;

    int video\_data\_len = 0;

    char\* paudiodata = NULL;

    int audio\_data\_len = 0;

    int64\_t pts = 0;

    int keyflag = 0;

    int encflag = 0;

    // get sdk version

SVPush\_API\_Version(ver, 256);

printf(ver);

printf("\n sunvalley cloud storage push sdk example\n");

    //sdk initialize

int ret = SVPush\_API\_Initialize(1024\*1024, MAX\_CHANNEL\_NUM);

CHECK\_RESULT(ret);

ret = SVPush\_API\_Init\_log((E\_SV\_LOG\_LEVEL)E\_LOG\_LEVEL\_MAIN, SV\_LOG\_OUTPUT\_MODE\_CONSOLE | SV\_LOG\_OUTPUT\_MODE\_FILE, "./log/");

CHECK\_RESULT(ret);

ret = SVPush\_API\_Set\_Token\_Server\_Interface(TOKEN\_SERVER\_URL);

CHECK\_RESULT(ret);

signal(SIGINT, sig\_stop);

    signal(SIGQUIT, sig\_stop);

    signal(SIGTERM, sig\_stop);

    // connect to media server

    lcid = SVPush\_API\_Connect\_By\_DeviceSN(NULL, NULL, TEST\_DEVICE\_SN, event\_callback);

    if(lcid < 0)

    {

        printf("push sdk connect media server failed, lcid:%ld, TEST\_DEVICE\_SN:%s\n", lcid, TEST\_DEVICE\_SN);

        assert(0);

        return -1;

    }

    gettimeofday(&tv, NULL);

    unsigned long long untsamp = ((unsigned long long)tv.tv\_sec /\*+3600\*16\*/) \* 1000 + tv.tv\_usec / 1000;

    ret = SVPush\_API\_Send\_Metadata(lcid, E\_SV\_METADATA\_TYPE\_VIDEO\_ENCRYPT, E\_SV\_MEDIA\_ENCRYPT\_AES\_KEY\_FRAME, (long)(pkeystring));

    CHECK\_RESULT(ret);

    ret = SVPush\_API\_Send\_Metadata(lcid, E\_SV\_METADATA\_TYPE\_AUDIO\_ENCRYPT, E\_SV\_MEDIA\_ENCRYPT\_AES\_ALL\_FRAME, (long)(pkeystring));

    CHECK\_RESULT(ret);

    ret = SVPush\_API\_Send\_Metadata(lcid, E\_SV\_METADATA\_TYPE\_STREAM\_START, E\_IPC\_TIGGER\_TYPE\_FACE, (long)(&untsamp));

    CHECK\_RESULT(ret);

do

{

        encflag = 0;

        // read h264 data from file(annexb format)

        // ...

        //send h264 data packet

 ret = SVPush\_API\_Send\_Packet(lcid, E\_SV\_MEDIA\_TYPE\_H264, pvideodata, video\_data\_len, pts, keyflag, encflag);

        // read aac packet data from file

        // ...

        //send aac data packet(with adts header)

        keyflag = 1;

 ret = SVPush\_API\_Send\_Packet(lcid, E\_SV\_MEDIA\_TYPE\_AAC, paudiodata, audio\_data\_len, pts, keyflag, encflag);

        usleep(30000);

} while (g\_brunning);

    printf("after while\n");

    gettimeofday(&tv, NULL);

    untsamp = ((unsigned long long)tv.tv\_sec /\*+3600\*16\*/) \* 1000 + tv.tv\_usec / 1000;

    // segment end alarm time 10000ms

    ret = SVPush\_API\_Send\_Metadata(lcid, E\_SV\_METADATA\_TYPE\_STREAM\_END, 10000, (long)(&untsamp));

    CHECK\_RESULT(ret);

SVPush\_API\_UnInitialize();

return 0;