实时互动-工业能源版

远端设备（观看端）SDK

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

远端设备（观看端） SDK
基本介绍
SDK API 调用流程
C/C++ 远端设备 SDK API
Android Java API
远端设备配置说明
错误码及排查

远端设备（观看端） SDK
基本介绍

最近更新时间：2025-04-03 17:56:12

适用范围

本 SDK 用于远端设备（视频观看/操控端）接入, 适用于 Linux 和 Windows 64 位系统，提供 so、dll 动态链接库形式接入。

主要功能

本 SDK 面向远程实时操控场景提供低时延音视频通话和控制/状态数据传输，主要功能如下：

视频渲染

在指定句柄的窗口进行视频渲染，并能够自动响应窗口的缩放事件。

视频解码

支持 H264、H265 和 AV1 等视频格式解码。

视频传输

支持 H264、H265 和 AV1 等编码视频流的传输，具备抗弱网和低延迟传输能力。

二进制数据传输

支持二进制数据的透传，可向通话方传输二进制数据。

推流管理

SDK 可以根据观看需求来动态管理对端推流状态。

视频延时和链路状态报告

SDK 可以监测视频延时和视频状态，并通过回调接口报告对应的状态数据。

基本概念

设备 ID

分为现场设备 ID 和远端设备 ID，用于标识设备身份，设备 ID 具有唯一性。如果网络中有相同设备 ID 的设备连网，会出现 MQTT 信令反复被踢下线又重连的情况，影响正常使用。

视频流 ID

用于标识现场设备的视频流编号，编号从0开始递增，与 JSON 配置中的 stream_config 数组元素编号一致。远端设备拉取视频流时，会通过现场设备 ID + 视频流 ID 来指定要拉取的视频流。

会话权限

用于管理远端设备是否能看到现场设备的在线状态以及观看现场设备的视频. 没有目标现场设备会话权限的远端设备是无法与该现场设备建立视频会话。会话权限通过控制台或者云 API 的权限管理功能进行管理。

操控权限

用于管理远端设备的控制权限。拥有权限的远端设备才能对目标现场设备进行控制数据下发和语音互动, 否则只能进行视频观看. 操控权限通过SDK Permission相关接口进行请求和管理。

接收句柄（conn_fd）

用于标识远端设备接收视频流的视频流接收器，编号从0开始递增；视频流接收器，用于桥接现场设备视频流和本地视频渲染窗口，拉取视频流时，需要指定对应的视频流接收器来收流；如下是收取视频流的过程。
现场设备视频流 >> 远端设备视频流接收器 >> 视频渲染窗口

视频窗口句柄

用于标识视频窗口，属于指向视频窗口的指针，例如 win32 的 HWND。通过视频窗口句柄，SDK 可以通过在指定视频窗口上进行画面渲染。

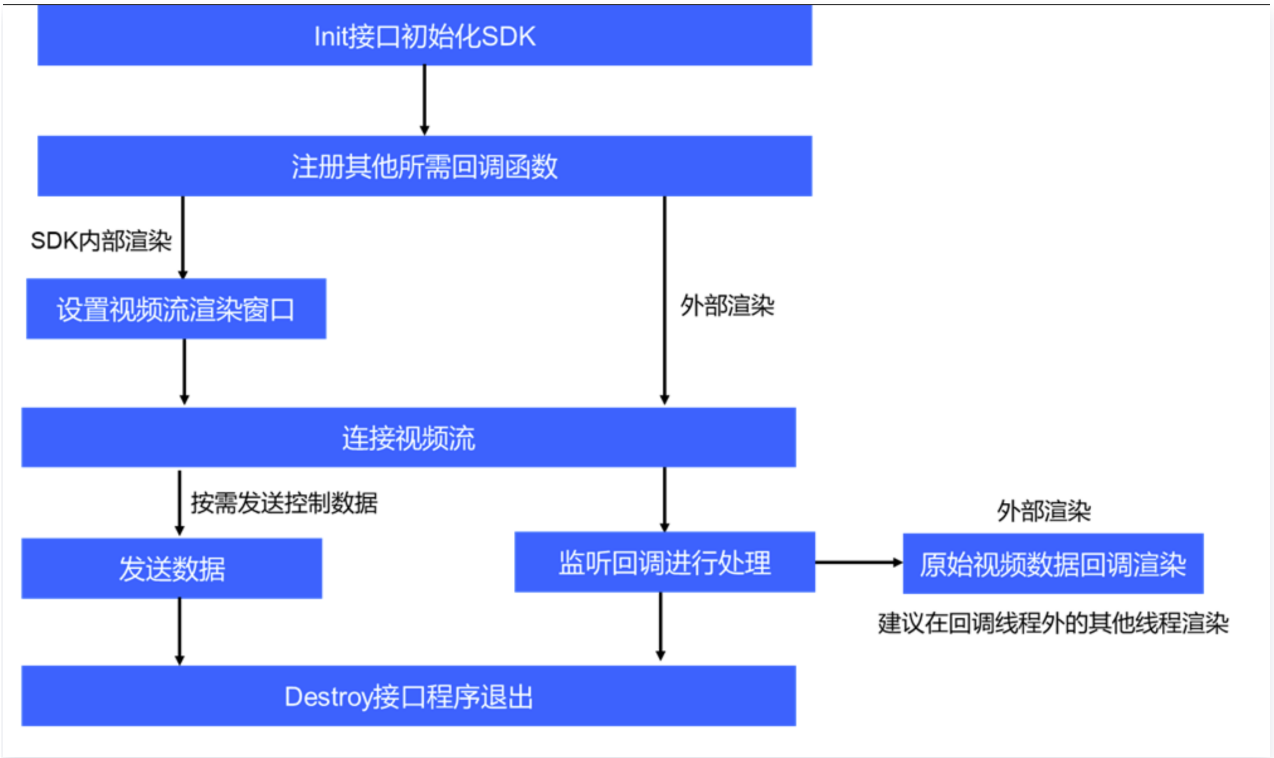
配置文件

包含 SDK 配置 JSON 字符串的文件，默认文件名为 config.json。SDK 可以基于配置文件或者配置 JSON 字符串初始化。

SDK API 调用流程

最近更新时间：2025-04-03 17:56:12

SDK API 调用流程



调用步骤建议

> **注意：**
> signalStae 信令回调注册需要在初始化函数之前调用，其他回调函数可以在初始化之后注册。

步骤1：加载配置文件初始化

注册 signalState 信令回调接口，通过 Json 文件加载或输入 Json 字符串加载等方式，调用 Init 接口初始化 SDK。

步骤2：注册所需的回调函数

注册所需的回调函数。根据需求注册控制消息回调、延迟信息回调、日志回调、连接状态回调、远端设备视频链路信息回调、现场设备链路信息回调、原始视频数据回调接口等函数，处理对应的状态信息和事件。

步骤3：设置视频流渲染窗口

如果使用内置渲染，调用设置渲染窗口接口，绑定视频接收器 fd 和要渲染的视频窗口句柄。

步骤4：连接视频流

调用 connect 接口，连接指定现场设备 ID 的指定编号视频流 。当需要切换或断开视频时，可调用 disconnect 接口或 connect 接口。

步骤5：按需调用发送函数

当有外部二进制数据发送需求时，在所需发送时刻，调用对应的发送接口输入要发送的二进制控制数据。

步骤6：按需退出

当程序退出时，调用销毁接口销毁 SDK，C# 退出需要主动调用销毁接口。

Sample 代码示意

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "trro_remote.h"

//该代码sample未包含UI渲染部分，仅用于视频连接过程参考
int main() {
    //注册signal回调
    TRRO_registerSignalStateCallback(nullptr, [](void *context,
SignalState state) {
        if(state == kTrroReady) {
            //服务器连接成功
            printf("init success \n");
        }
        if (state == kTrroAuthFailed) {
            //鉴权失败
            printf("device_id or password is incorrect\n");
        }
        if (state == kTrroKickout) {
            //被其他用户剔除
            printf("the device is kicked by server, may be there is
another device using the same device id\n");
        }
    });

    // 可按需替换为其他初始化函数，如使用配置字符串初始化
    int ret = TRRO_initJsonPath("./config.json");
    if(TRRO_SUCCED != ret) {
        if (ret == -TRRO_SIGNAL_CONNECT_OUTTIME) {
```

```
            printf("init process: wait for connecting\n");
        } else {
            printf("init fail ret %d\n", ret);
        }
    }

    //按需注册回调
    //注册视频连接状态回调
    TRRO_registerOnState(this, [](void* context, const char* gwid, int
stream_id, int conn_fd, int state) {
      printf("###### onstate gwid %s, streamid %d  conn_fd %d  state
%d\n", gwid, stream_id, conn_fd, state);
    });
    //注册媒体传输状态回调
    TRRO_registerOnMediaState(this, [](void* context, int windows_id,
int fps, int bps, int rtt, long long decd, int jitter,
        long long packets_lost, long long packets_received, int stun) {
           printf("onMediaState window id %d fps %d bps %d rtt %d decd
%lld rb %d lost %lld rec %lld stun %d\n",windows_id, fps, bps, rtt,
decd, jitter, packets_lost, packets_received, stun);
    });
    //注册延迟信息回调
    TRRO_registerAllLatencyCallback(this, [](void* context, const char*
gwid, int windows_id, long long latency1, long long latency2, long long
gateway_time, int vcct) {
        printf("allLatency %s, %d, latency1 %lld latency2 %lld video
time %lld, vcct %d\n", gwid, windows_id, latency1, latency2,
gateway_time, vcct);
    });

    //注册原始视频数据回调，可用于外部渲染
    TRRO_registerRemoteFrameCallback(this, [](void * context, const char
* gwid, int stream_id, int conn_id, const char* data, int width, int
height, long long videotime) {
       //可以在这里调用渲染函数播放视频数据，但不要阻塞处理，处理延迟大请异步调用其他
线程处理，否会导致视频卡顿
       //showMyVideo(gwid, conn_id, (unsigned char*)data, width,
height);
    });

    //视频接收句柄数组
    int confds[8] = {0,1,2,3,4,5,6,7};
    //渲染窗口句柄数组
```

```
    WindowIdType window_hwnds[8] = { nullptr};
    //SDK内部渲染设置目标渲染窗口win32 HWND窗口句柄即可，外部渲染可设置空窗口句柄
(nullptr)，通过实现showMyVideo进行渲染
    TRRO_setWindows(confds, window_hwnds, 8);
    //连接视频流，假设每个现场设备需要连接4个视频流
    int stream_num = 4;
    int stream_indexs[4] = {0,1,2,3};
    //连接现场设备1的4个视频流0-3，使用视频接收句柄 0-3
    //公有云connect接口中field_deviceId的格式为  projectId/fieldDeviceId
    TRRO_connect("field_deviceId1", "", stream_num , &stream_indexs[0],
&confds[0]);
    //连接现场设备2的4个视频流0-3，使用视频接收句柄 4-7
    TRRO_connect("field_deviceId2", "", stream_num , &stream_indexs[0],
&confds[4]);
    //等待退出
    sleep(10000);
    //断开视频流
    TRRO_disconnectAll();
    //退出SDK,释放资源
    TRRO_destroy();

    return 0;
}
```

C/C++ 远端设备 SDK API

最近更新时间：2025-06-19 10:11:22

API 描述

适用于 Windows/Linux 64 位系统。头文件引用 trro_remote.h，API 可参考头文件中接口注释。下面列举常用接口描述。

API 概览

初始化及销毁接口

函数列表	描述
TRRO_initJsonPath	使用 JSON 配置文件路径初始化 SDK
TRRO_initJson	使用 JSON 字符串初始化 SDK
TRRO_destroy	销毁 SDK

现场设备查询接口

函数列表	描述
TRRO_getGwList	获取有会话权限的在线现场设备列表
TRRO_getGwInfo	获取目标现场设备信息，需要有对应设备会话权限

视频操作接口

函数列表	描述
TRRO_setWindows	设置视频流渲染窗口
TRRO_connect	连接视频流（depreciated），请用新接口 TRRO_connnectField
TRRO_connectField	连接视频流（新）
TRRO_disconnect	关闭指定视频连接
TRRO_disconnectAll	关闭所有视频连接
TRRO_SetRenderConfig	更新视频渲染配置，可开启超分、夜视增强、旋转、镜像等

版权所有：腾讯云计算（北京）有限责任公司

TRRO_fieldDeviceEncodeConfig	更新现场设备目标视频流编码参数
TRRO_startMediaPush	开启视频流转推
TRRO_stopMediaPush	关闭视频流转推

音视频输入输出接口

函数列表	描述
TRRO_registerVideotimeCallback	注册视频帧采集时间戳回调（depreciated），请用帧回调获取时间戳
TRRO_registerRemoteFrameCallback	注册原始视频图像帧回调（默认YUVI420格式)
TRRO_registerRemoteEncodedFrameCallback	注册编码视频图像帧回调
TRRO_useExternalDecoder	使用外部视频解码，打开会关闭原始视频图像回调，只回调编码视频帧。
TRRO_externAudioData	外部输入音频数据（PCM 数据）
TRRO_registerRemoteMixAuidoFrameCallback	注册音频数据回调函数（PCM 数据）

消息类接口

函数列表	描述
TRRO_sendControlData	向现场设备发送控制数据
TRRO_registerReportDataCallback	监听现场设备发送来的数据

权限控制接口

函数列表	描述
TRRO_requestPermission	请求控制权限
TRRO_registerOnOperationPermissionState	注册现场设备权限状态回调

状态回调类接口

函数列表	描述

TRRO_registerLatencyCallback	注册时延回调（depreciated），请用新接口AllLantencyCallback
TRRO_registerAllLatencyCallback	注册时延回调（新）
TRRO_registerSignalStateCallback	注册信令服务连接状态回调
TRRO_registerOnState	注册视频流连接状态回调
TRRO_registerOnMediaStat	注册视频流传输状态回调（depreciated），请用新接口OnMediaStateInfo
TRRO_registerOnMediaStateInfo	注册视频流传输状态回调（新）
TRRO_registerOnAudioMediaStateInfo	注册音频流传输状态回调
TRRO_registerFieldSideNetworkState	注册连接的现场设备网络状态回调

音频控制接口

函数列表	描述
TRRO_audioMute	静音现场设备

诊断类接口

函数列表	描述
TRRO_registerLogCallback	注册日志回调
TRRO_registerOnDiagReport	注册诊断信息回调接口

图像转换接口

函数列表	描述
TRRO_I4202ARGB	I420转 ARGB

软件信息接口

函数列表	描述
TRRO_getVersion	获取版本号

初始化接口

使用说明：可根据配置输入类型（文件/字符串），选择对应初始化接口初始化，只需使用一个。

使用 JSON 配置文件路径初始化

```
/*
* @name :  TRRO_initJsonPath
* @brief : SDK 初始化，读取指定路径文件初始化， 接口为同步阻塞模式，等待信令连接成功后返回
* @input : jsonPath   JSON文件路径
* @return : 成功 1 失败 <= 0
*/
int TRRO_initJsonPath(const char* jsonPath);
```

参数	含义
jsonPath	JSON 配置文件路径
返回值	• 成功：1 • 失败：<= 0

使用 json 字符串初始化

```
/*
* @name :  TRRO_initJson
* @brief : SDK 初始化，读取输入JSON字符串初始化， 接口为同步阻塞模式，等待信令连接成功后返回
* @input : json   JSON 配置字符串
* @return : 成功 1 失败 <= 0
*/
int TRRO_initJson(const char* json);
```

参数	含义
json	配置信息 JSON 字符串
返回值	• 成功：1 • 失败：<= 0

销毁 SDK

使用说明：可通过该接口销毁 SDK，释放资源，可用于程序退出时需要主动释放底层资源场景（例如 C# 程序）。

```
//销毁 SDK
extern "C" TRRO_EXPORT void TRRO_destroy();
```

现场设备查询接口

获取在线现场设备列表

使用说明：可通过该接口获取到远端设备有会话权限的在线现场设备列表，可根据实际需要使用。当有权限的设备数量过多时，建议通过服务端 API 获取。

```
/*
* @name :  TRRO_getGwList
* @brief : 获取在线现场设备设备列表，接口为同步阻塞模式
* @input : void
* @return : JSON 字符串
*           {"ret":0,"msg":"suc","gateways":
[{"deviceID":"xx","name":"xx","type":"gateway","status":"connected","str
eams":4,"timestamp":1682231838673,"version":"xx"}],"count":1}
*/
const char* TRRO_getGwList();
```

参数	含义

返回值	现场设备列表 JSON 字符串
	{
	"ret":0,
	"msg":"suc",
	"gateways “:[
	{"deviceID":"xx","name":"xx","type":"gateway","status":"connected
	","streams":4,"timestamp":1682231838673,"version":"xx"}
	],
	"count":1
	}

获取在线现场设备信息

使用说明：获取目标现场设备信息，需要有对应设备会话权限。当设备不在线或无权限时返回 null。

```
/*
* @name :  TRRO_getGwList
```

```
* @brief : 获取对应网关的版本，接口为同步阻塞模式.
* @input : gwid 设备id 公有云含projectID前缀，形式如ProjectId/FieldDeviceId
* @return : json字符串， 使用方不需要释放指针， 获取到值后立即拷贝。
*            {"ret":0, "msg": "suc",
"deviceID":"xx","name":"xx","type":"gateway","status":"connected","strea
ms":4,"timestamp":1682231838673,"version":"xx","sdk_mode":"server2","abi
lity":0,"licenseExpire":1745666980}
*/
extern "C" TRRO_EXPORT const char* TRRO_getGwInfo(const char* gwid);
```

参数	含义
gwid	目标现场设备 ID，公有云形式例如 ProjectId/FieldDeviceId

返回值	现场设备列表 JSON 字符串
	{
	"ret":0,
	"msg":"suc",
	"deviceID":"xx",
	"name":"xx",
	"type":"gateway",
	"status":"connected",
	"streams":4,
	"timestamp":1682231838673,
	"version":"xx",
	"sdk_mode":"server2",
	"ability":0,
	"licenseExpire":1745666980
	}

视频操作接口

连接视频流（depreciated）

使用说明：可通过该接口对现场设备视频流进行连接。

```
/*
* @name :  TRRO_connect
* @brief : 发起视频连接，可多次调用连接不同流，异步模式，根据 onState 状态回调确认
视频连接成功
* @input : gwid          目标连接的现场设备 ID， 公有云传设备ID时需要带上
projectid前缀，如projectid/fieldDeviceId
*          record_config:
```

```
*                          优先使用录制配置，当默认录制配置命名规则无法满足需求时使
用，JSON 字符串，需要对每一路进行配置 eg:"
{\"file_names\": [{\"file\":\"test\", \"duration\":15}, {\"file\":\"test
01\", \"duration\":15}]}"  ps:file 文件名 duration 分片时长单位秒
*          streams_num 要拉取现场设备视频流的个数， 与conn_fds和streams_num数
组长度匹配， 值的范围 1 到 现场设备支持的device_streams数量
*          streams_id  现场设备视频流的 ID 数组， 现场设备视频流 ID 从0开始， 最
大值为现场设备支持的 device_streams数量 -1
*          conn_fds    conn_fd数组，conn_fd为接收视频流的句柄标识，自行编号，取
值从0开始，最大值为 max_streams -1，max_streams在远端设备配置文件配置
* @return : 成功 1 失败 <= 0
*/
int TRRO_connect(const char* gwid, const char* record_config, int stream
s_num, int* streams_id, int* conn_fds);
```

参数	含义
gwid	现场设备 ID，公有云传设备 ID 时需要带上 projectid 前缀，例如 projectid/fieldDeviceId
record_config	视频录制配置，建议当默认录制配置无法满足需求时使用，JSON 字符串，需要对每一路进行配置，例如： `{ "file_names": [{ "file": "test_%D_%T", "duration": 15 }, { "file": "test01_%D_%T", "duration": 15 }] }` 参数： • file：文件名, 支持%D自动补充日期，%T自动补充时间 • duration：切片时长（单位：秒）
streams_num	要连接视频流的数量
streams_id	要连接的现场设备视频流 ID 数组

conn_fds	接收视频流使用的接收视频流句柄数组
返回值	• 成功：1 • 失败：<= 0

连接视频流（new）

使用说明：可通过该接口对现场设备视频流进行连接，可支持跨 SDK 模式兼容拉流。打开兼容模式可能会影响秒开，建议在有明确需求时打开。

```
/*
* @name :  TRRO_connectField
* @brief : 发起视频连接，可多次调用连接不同流，异步模式，根据onState状态回调确认视频连接成功
* @input : gwid            目标连接的现场设备ID，公有云传设备ID时需要带上projectid前缀，如projectid/fieldDeviceId
*          streams_num     要拉取现场设备视频流的个数， 与conn_fds和streams_num数组长度匹配， 值的范围 1 到 现场设备支持的device_streams数量
*          streams_id      现场设备视频流的ID数组， 现场设备视频流ID从0开始， 最大值为现场设备支持的device_streams数量 -1
*          conn_fds        conn_fd数组，conn_fd为接收视频流的句柄标识，自行编号，取值从0开始，最大值为max_streams -1，max_streams在远端设备配置文件配置
*          compatible      是否开启版本兼容： 设置0，根据配置文件的“sdk_mode“字段来决定连接模式；设置为1，该接口会获取网关版本信息从而根据网关模式自动适配模式连接（由于需要获取版本信息会影响秒开）。默认值为0
*          record_config: 优先使用录制配置，当默认录制配置命名规则无法满足需求时使用，
                           json字符串，需要对每一路进行配置 eg:"{\"file_names\": [{\"file\":\"test\", \"duration\":15},{\"file\":\"test01\", \"duration\":15}]}"
                           ps:file 文件名 duration 分片时长单位秒; 不使用时默认为NULL
* @return : 成功 1 失败 <= 0
*/
extern "C" TRRO_EXPORT int TRRO_connectField(const char* gwid, int streams_num, int* streams_id, int* conn_fds, int compatible = 0, const char* record_config = NULL);
```

参数	含义

gwid	现场设备 ID，公有云传设备 ID 时需要带上 projectid 前缀，例如 projectid/fieldDeviceId
streams_num	要连接视频流的数量
streams_id	要连接的现场设备视频流 ID 数组
conn_fds	接收视频流使用的接收视频流句柄数组
compatible	是否开启兼容模式， 0 不开启， 1开启。开启兼容模式，可跨 SDK 模式拉流，但会增加首帧延迟，建议有明确需求时使用。
record_config	视频录制配置，建议当默认录制配置无法满足需求时使用，JSON 字符串，需要对每一路进行配置，例如： `{ "file_names": [{ "file": "test_%D_%T", "duration": 15 }, { "file": "test01_%D_%T", "duration": 15 }] }` 参数： • file：文件名, 支持%D自动补充日期，%T自动补充时间 • duration：切片时长（单位：秒）
返回值	• 成功：1 • 失败：<= 0

设置视频流渲染窗口

使用说明：可通过该接口设置渲染窗口，内部渲染时使用，外部渲染时设置空指针即可。

```
/*
* @name :  TRRO_setWindows
* @brief : 设置接收流句柄对应的显示窗口句柄
* @input : conn_fds 接收视频流句柄数组
*          windows  显示窗口句柄数组， 显示窗口句柄为显示窗体的句柄/指针，如
Windows 平台的 HWND;
```

```
*                    外部渲染时，数组中的显示窗口句柄配置为 nullptr
*          num       设置显示窗口数量，与 conn_fds 和 windows 数组长度匹配
* @return :void
*/
void TRRO_setWindows(int* conn_fds, WindowIdType * windows, int num);
```

参数	含义
conn_fds	接收视频流句柄数组
windows	显示窗口句柄数组，外部渲染时可设置为空指针数组
num	与 conn_fds 和 windows 数组长度一致

关闭指定视频连接

使用说明：可通过该接口断开视频连接。

```
/*
* @name :  TRRO_disconnect
* @brief : 关闭conn_fds 对应的视频连接
* @input : conn_fds 要关闭视频连接对应的视频接收句柄数组
*          fd_num conn_fd数组长度
* @return : 成功 1 失败 <= 0
*/
int TRRO_disconnect(int* conn_fds, int fd_num);
```

参数	含义
conn_fd	视频接收句柄数组
fd_num	数组长度
返回值	成功 1 失败 <= 0

关闭所有视频连接

```
/*
* @name :  TRRO_disconnectAll
* @brief : 关闭所有视频连接
* @input : void
* @return : 成功 1 失败 <= 0
```

```
*/
int TRRO_disconnectAll();
```

更新视频流渲染配置

使用说明：设置接收流句柄对应的内部渲染模式，仅内部渲染时使用。可设置拉伸模式、旋转、镜像、超分、夜视增强等功能

```
/*
* @name :  TRRO_SetRenderConfig
* @brief : 设置接收流句柄对应的内部渲染模式，仅内部渲染时使用
* @input : conn_fds 要设置渲染模式的接收视频流句柄数组
*          rotation 旋转角度 数组 rotation 为正顺时针旋转，rotation为负 逆时针
旋转
*          scale 分辨率匹配模式数组 0 窗口拉伸， 1 保持原比例
*          config 预留渲染配置数组 0x00000XYZ X为1~2时开启夜视不同等级 推荐为2，
Y为1时为开启超分模式，Z为1时为开启镜像渲染
*          num  配置的内部渲染流数量，与输入数组长度匹配
* @return : 成功 1 失败 <= 0
*/
extern "C" TRRO_EXPORT int TRRO_SetRenderConfig(int *conn_fds, int
*rotation, int *scale, int *config, int num);
```

参数	含义
conn_fds	要设置渲染模式的接收视频流句柄数组
rotation	旋转角度，数组 rotation 为正顺时针旋转，rotation为负逆时针旋转， 缺省为0.
scale	分辨率匹配模式数组 0 窗口拉伸， 1 保持原比例，缺省为0.
config	渲染配置数组，数组元素为32位int型，形如0x00000XYZ，缺省为0. X为1~3时开启夜视增强，对应不同强度，推荐强度为2，Y为1时为开启超分模式，Z为1时为开启镜像渲染；
num	配置的内部渲染流数量，与输入数组长度匹配

更新现场设备编码参数

使用说明：设置接收流句柄对应的内部渲染模式，仅内部渲染时使用。

```
/*
* @name :  TRRO_fieldDeviceEncodeConfig
```

```
* @brief : 更新现场设备目标视频流编码参数
* @input : gwid 现场设备id
*          streams_id 目标视频流编号
*          encode_config 待更新的编码参数，JSON 格式字符串，缺省字段将保持当前值
不进行更新。 下面是更新支持的编码参数示意
*  {
*       "fps": 30,
*       "encode_width": 1920,
*       "encode_height": 1080,
*       "bps": 3000,
*       "min_fps": 30,
*       "min_bps": 1800,
*       "force_min": 0,
*       "min_width": 1920
*  }
*/
extern "C" TRRO_EXPORT int TRRO_fieldDeviceEncodeConfig(const char*
gwid, int streams_id, const char* encode_config);
```

参数	含义
gwid	现场设备 ID
streams_id	目标视频流编号

encode_config	待更新的编码参数，JSON 格式字符串，缺省字段将保持当前值不进行更新。 下面是更新支持的编码参数示意
	{
	"fps": 30,
	"encode_width": 1920,
	"encode_height": 1080,
	"bps": 3000,
	"min_fps": 30,
	"min_bps": 1800,
	"force_min": 0,
	"min_width": 1920
	}

开启视频流转推

使用说明：将对应视频接收句柄接收的视频流转推到指定RTMP URL的直播/存储服务。当远端对视频流disconnect时，会自动结束转推。

```
/*
* @name :  TRRO_startMediaPush
* @brief : 开启rtmp/rtsp转推，调用TRRO_stopMediaPush停止，当调用
connect/disconnect函数后conn_fd连接的视频流发生切流或断开时，也会自动停止推流，需
要重新start才能推流
* @input : conn_fd      接收视频流的句柄标识
*          path         推流地址
* @return : 成功 1 失败 <= 0
*/
extern "C" TRRO_EXPORT int TRRO_startMediaPush(int conn_fd, const char*
path);
```

参数	含义
conn_fd	视频接收句柄
path	视频流转推的目标 RTMP URL，以 rtmp://开头

停止视频流转推

使用说明：停止对应视频接收句柄接收的视频流转推。

```
/*
* @name :  TRRO_stopMediaPush
* @brief : 停止rtmp/rtsp转推
* @input : conn_fd      接收视频流的句柄标识
* @return : 成功 1 失败 <= 0
*/
extern "C" TRRO_EXPORT int TRRO_stopMediaPush(int conn_fd);
```

参数	含义
conn_fd	视频接收句柄

音视频输入输出接口

注册视频帧时间戳回调函数

使用说明：可通过该接口获取现场设备 SDK 采集当前远端接收到视频帧时的时间戳，对应现场设备系统时钟。

```
extern "C" TRRO_EXPORT void TRRO_registerVideotimeCallback(void*
context, TRRO_onVideotimeReport * callback);
/*
* @name :  TRRO_onVideotimeReport (depreciated)
* @brief : 视频帧采集时间戳回调，旧接口，建议使用 onAllLatencyReport 接口获取
* @input : context      回调上下文指针，返回注册回调函数时传入的 context
*          gwid         视频来源现场设备 ID
*          conn_fd        对应视频流接收句柄
*          videotime    当前视频帧的现场设备采集时间戳,对应系统时钟
* @return : void
*/
typedef void STD_CALL TRRO_onVideotimeReport(void* context, const char*
gwid, int conn_fd, long long videotime);
```

视频原始图像回调

使用说明：可通过该接口获取现场设备发送的原始图像视频帧，默认为YUVI420格式。

> **注意：**
> 请勿在回调线程长时间阻塞，若需要长时间处理建议转移到其他线程处理，否则会引起视频卡顿或延迟变大。

```
extern "C" TRRO_EXPORT void TRRO_registerRemoteFrameCallback(void*
context, TRRO_onRemoteFrameData * callback, int frame_type = 0);

/*
* @name :  TRRO_onRemoteFrameData
* @brief : 原始视频图像帧回调
* @input : context      回调上下文指针，返回注册回调函数时传入的 context
*          gwid         视频帧的来源现场设备 ID
*          stream_id    视频帧的来源现场设备流 ID
*          conn_fd      接收视频流的句柄标识
*          data         视频图像帧数据，数据格式为YUVI420，可通过I4202ARGB接口转
ARGB
*          width        视频帧宽
*          height       视频帧高
*          videotime    当前视频帧的现场设备采集时间戳
* @return : void
*/
typedef void STD_CALL TRRO_onRemoteFrameData(void* context, const char*
gwid, int stream_id, int conn_fd, const char* data, int width, int
height, long long videotime);
```

参数	含义
context	上下文指针
gwid	该视频图像来源的现场设备 ID
stream_id	该视频图像对应的现场设备视频流ID
conn_fd	该视频图像对应的远端设备视频流接收句柄
data	视频图像数据
width	视频图像宽
height	视频图像高
videotime	现场设备 SDK 采集该视频图像时的时间戳，对应 SteadClock 时钟

视频编码帧回调

使用说明：可通过该接口获取现场设备发送的编码视频帧。

> **注意：**
> 请勿在回调线程长时间阻塞，若需要长时间处理建议转移到其他线程处理，否则会引起视频卡顿或延迟变大。

```
extern "C" TRRO_EXPORT void
TRRO_registerRemoteEncodedFrameCallback(void* context,
TRRO_onRemoteEncodedFrameData* callback);

enum TrroCodec {
  Trro_H264 = 0,
  Trro_H265 = 1,
  Trro_AV1 = 2,
};


/*
 * @name :  TRRO_onRemoteEncodedFrameData
 * @brief : 编码视频图像帧回调
 * @input : context        回调上下文指针，返回注册回调函数时传入的 context
 *          gwid           视频帧的来源现场设备 ID
```

```
 *          stream_id      视频帧的来源现场设备流 ID
 *          conn_fd        接收视频流的句柄标识
 *          data           编码的视频图像帧数据
 *          len            编码的视频图像帧的长度
 *          trro_codec     编码的 codec 类型
 *          is_key_frame   是否是关键帧
 *          videotime      当前视频帧的现场设备采集时间戳
 * @return : void
 */
typedef void STD_CALL TRRO_onRemoteEncodedFrameData(void* context, const
char* gwid, int stream_id, int conn_fd,
                                                    const char* data,
int len, TrroCodec trro_codec, bool is_key_frame,
                                                    long long
videotime);
```

参数	含义
context	上下文指针
gwid	该视频帧来源的现场设备 ID
stream_id	该视频帧对应的现场设备视频流ID
conn_fd	该视频帧对应的远端设备视频流接收句柄
data	视频帧数据
len	视频帧数据大小
TrroCodec	视频帧编码格式 Trro_H264 = 0, Trro_H265 = 1, Trro_AV1 = 2,
is_key_frame	是否为关键帧，true 为关键帧， false不是关键帧；
videotime	现场设备 SDK 采集该视频帧时的时间戳，对应 SteadClock 时钟

视频编码帧回调

使用说明：可通过该接口获取现场设备发送的编码视频帧。

> **注意：**

> 请勿在回调线程长时间阻塞，若需要长时间处理建议转移到其他线程处理，否则会引起视频卡顿或延迟变大。

```
extern "C" TRRO_EXPORT void
TRRO_registerRemoteEncodedFrameCallback(void* context,
TRRO_onRemoteEncodedFrameData* callback);

enum TrroCodec {
  Trro_H264 = 0,
  Trro_H265 = 1,
  Trro_AV1 = 2,
};


/*
 * @name :  TRRO_onRemoteEncodedFrameData
 * @brief : 编码视频图像帧回调
 * @input : context        回调上下文指针，返回注册回调函数时传入的 context
 *          gwid           视频帧的来源现场设备 ID
 *          stream_id      视频帧的来源现场设备流 ID
 *          conn_fd        接收视频流的句柄标识
 *          data           编码的视频图像帧数据
 *          len            编码的视频图像帧的长度
 *          trro_codec     编码的 codec 类型
 *          is_key_frame   是否是关键帧
 *          videotime      当前视频帧的现场设备采集时间戳,对应现场设备SteadClock
而不是系统时钟
 * @return : void
 */
typedef void STD_CALL TRRO_onRemoteEncodedFrameData(void* context, const
char* gwid, int stream_id, int conn_fd,
                                                const char* data,
int len, TrroCodec trro_codec, bool is_key_frame,
                                                long long
videotime);
```

使用外部视频解码

使用说明：此接口用于关闭 SDK 内部视频解码，转而使用外部视频解码。

> **说明：**

开启外部解码时，原始视频图像将不会回调，只会回调编码数据。

```
//是否使用外部解码(只针对视频)

extern "C" TRRO_EXPORT int TRRO_useExternalDecoder(bool external);
```

参数	含义
external	true 使用外部解码 false 使用内部解码

输入外部音频流（PCM 数据）

说明：
配置中需要“audio_external”节点值修改为1。

使用说明：此接口用于提供用户输入音频流进行传输，输入为 PCM 数据，16位采样。

```
/* @name  TRRO_externAudioData(Experimental)
* @brief 外部音频数据输入(pcm数据), 16位音频采样
* @param[in]   data          源数据
* @param[in]   data_size     数据大小
* @param[in]   channel       音频声道数
* @param[in]   sample_rate   音频采样率
* @return 1 for success, other failed
*/
extern "C" TRRO_EXPORT int TRRO_externAudioData(const char* gwid, const
char* data, int data_size, int channel, int sample_rate);
```

参数	含义
gwid	要发送的现场设备 ID
data	音频流数据
data_size	数据大小
channel	声道数量 • 1：单声道

	• 2：双声道
sample_rate	采样频率，支持8k/16k/24k/32k/44.1k/48kHz等采样率，例如48k采样率输入为48000

注册外部音频回调函数（PCM 数据）

> **说明：**
> 配置中需要“audio_play”节点值修改为"outside"（关闭SDK内部音频播放），且打开音频接收配置"audio_receive":1。

使用说明：此接口给上层应用提供音频数据，用于应用自行播放音频。每次回调音频数据长度对应10ms。

```
/**
* @name  TRRO_onRemoteMixAuidoFrame
* @brief 对端多路混音后音频原始数据回调
* @param[in] context      回调上下文指针，返回注册回调函数时传入的context
* @param[in] data         音频PCM数据 10ms 16bits，小端对齐
* @param[in] length       数据长度
* @param[in] channel      音频声道数目，如单声道为1，双声道为2
* @param[in] sample_rate    音频采样率
* @return : void
*/

typedef void TRRO_onRemoteMixAuidoFrame(void* context, const char* data,
int length, int channel, int sample_rate);

/**
* @name TRRO_registerRemoteMixAuidoFrameCallback
* @brief 注册远端混音音频数据回调函数
* @param[in]    context              上下文
* @param[in]    callback             回调函数
* @return 1 for success, other failed
*/
extern "C" TRRO_EXPORT int
TRRO_registerRemoteMixAuidoFrameCallback(void* context,
TRRO_onRemoteMixAuidoFrame * callback);
```

消息类接口

向现场设备发送控制数据

使用说明：可通过该接口给现场设备发送数据。目前限制单次发送数据大小1KB字节，频率限制100次/秒。每秒总大小小于100KB。 发消息前需要拥有对应现场设备的 master 控制权限。

```
/*
* @name :  TRRO_sendControlData
* @brief : 向现场设备发送控制数据
* @input : gwid      目标现场设备 ID
*          msg       发送消息，二进制透传
*          len       消息长度
*          qos       消息传输qos  0:不可靠传输 1:可靠传输
* @return : 成功 1 失败 <= 0
*/
extern "C" TRRO_EXPORT int TRRO_sendControlData(const char* gwid, const
char* msg, int len, int qos = 0);
```

参数	含义
gwid	现场设备 ID
msg	发送二进制数据
len	消息长度
qos	发送 qos： • 0：超低延迟高频传输， 适合高频远控消息，大小限制1KB • 1：可靠传输，网络不好时消息可能会堵塞，适合低频重要消息，大小限制10KB
返回值	成功：1 失败：<= 0

注册现场设备消息回调函数

使用说明：可通过该接口接收现场设备发送的数据。

```
extern "C" TRRO_EXPORT void TRRO_registerReportDataCallback(void*
context, TRRO_onReportData * callback);

/*
* @name :  TRRO_onReportData
* @brief : 接收来自现场设备信息回调
```

```
* @input : context      回调上下文指针，返回注册回调函数时传入的 context
*          gwid         消息来源现场设备 ID
*          msg          消息，二进制透传
*          len          消息长度
*          qos          消息来源传输 qos， 0:不可靠传输，1:可靠传输
* @return : void
*/
typedef void STD_CALL TRRO_onReportData(void* context, const char* gwid,
const char* msg, int len, int qos);
```

参数	含义
context	上下文指针
gwid	现场设备 ID
msg	接收的二进制数据
len	消息长度
qos	发送 qos： • 0：不可靠传输 • 1：可靠传输

权限控制类接口

请求现场设备控制权限

使用说明：向现场设备请求控制权限，请求成功后才可向对应现场设备发送及接收控制数据。

```
//操控权限
enum TrroPermission {
    kPermissionGuest = 0,
    kPermissionMaster = 1,
};

/*
* @name : TRRO_requestPermission
* @brief : 向网关发出权限请求，网关会反馈TRRO_OnOperationPermissionState 更新
权限
* @input : gwid          目标设备 ID
*          permisson     参考结构体 TrroPermission
* @return : 成功1 失败 <= 0
*/
```

```
extern "C" TRRO_EXPORT int TRRO_requestPermission(const char* gwid, int
permisson);
```

参数	含义
gwid	目标现场设备 ID
permisson	控制权限

现场设备控制权限回调

```
extern "C" TRRO_EXPORT void
TRRO_registerOnOperationPermissionState(void* context,
TRRO_OnOperationPermissionState* callback);


//操控权限
enum TrroPermission {
    kPermissionGuest = 0,
    kPermissionMaster = 1,
};

/*
* #name :   TRRO_OnOperationPermissionState
* @brief :  回调现场设备操控权限状态通知
* @input :  context            回调上下文指针，返回注册回调函数时传入的
context
*           field_devid        来源现场设备 ID
*           self_permission    本设备当前的操控权限，参考 TrroPermission， 0
是 guest，只有观看权限， 1是 master，拥有完全控制权限
*           master_devid       拥有 master 权限的远端设备 ID
* @return : void
*/
typedef void STD_CALL TRRO_OnOperationPermissionState(void* context,
const char* field_devid, int self_permission, const char* master_devid);
```

参数	含义
context	上下文指针

field_devid	现场设备 ID
self_permission	自身对该现场设备拥有的控制权限级别
master_devid	获得该现场设备 master 控制权的远端设备ID，空字符串代表没有远端设备获取了该现场设备的 master 控制权

状态回调类接口

延迟状态回调

使用说明：可用于监听视频延迟、控制延迟等延迟状态。

```
extern "C" TRRO_EXPORT void TRRO_registerLatencyCallback(void* context,
TRRO_onLatencyReport * callback);
extern "C" TRRO_EXPORT void TRRO_registerAllLatencyCallback(void*
context, TRRO_onAllLatencyReport * callback);

/*
* @name :  TRRO_onLatencyReport (depreciated)
* @brief : 视频时延回调，旧接口，建议使用onAllLatencyReport接口获取
* @input : context      回调上下文指针，返回注册回调函数时传入的context
*          gwid         视频来源现场设备id
*          conn_fd      对应视频流接收句柄
*          latency      返回视频传输时延ms
* @return : void
*/
typedef void STD_CALL TRRO_onLatencyReport(void* context, const char*
gwid, int conn_fd, long long latency);


/*
* @name :  TRRO_onAllLatencyReport (Recommended)
* @brief : 时延回调
* @input : context      回调上下文指针，返回注册回调函数时传入的 context
*          gwid         视频来源现场设备 ID
*          conn_fd      对应视频流接收句柄
*          latency1     视频传输时延ms ，可能存在一定误差，与网络波动及上下行时延差
值有关
*          latency2     视频传输+控制下发时延ms
*          videotime    当前视频帧的现场设备采集时间戳，基于现场设备系统时钟
*          rcct         控制信道往返时延ms， 等于状态上报延迟+控制下发延迟
* @return : void
*/
```

```
typedef void STD_CALL TRRO_onAllLatencyReport(void* context, const char*
gwid, int conn_fd, long long latency1, long long latency2, long long
videotime, int rcct);
```

参数	含义
context	上下文指针
gwid	视频流对应的现场设备 ID
conn_fd	视频流对应的远端设备接收句柄
latency1	估计的视频流传输延迟，单位 ms，估计误差与网络波动及上下行时延差值有关
latency2	视频传输 + 控制下发时延 ms
videotime	当前接收视频帧的现场设备采集时间戳
rcct	控制消息通道往返时延 ms, 等于状态上报延迟+控制下发延迟

信令连接状态回调

使用说明：可用于监听与服务端信令连接的状态。

```
extern "C" TRRO_EXPORT int TRRO_registerSignalStateCallback(void
*context, TRRO_onSignalState *callback);

/**
 * SignalState 信令连接状态枚举
 */
enum SignalState {
    kTrroReady = 0, /**< 首次连接建立成功 */
    kTrroLost = 1,  /**< 连接断开，内部会进行自动重连 */
    kTrroReup = 2,  /**< 自动重连成功 */
    kTrroKickout = 3,  /**< 被同名用户剔除 */
    kTrroAuthFailed = 4,  /**< 用户名或者密码错误 */
};

/**
* @name TRRO_onSignalState
* @brief 信令连接状态回调
* @param[in]     context        上下文指针
* @param[in]     state          信令连接状态，参考枚举SignalState
```

```
* @return void
*/
typedef void TRRO_onSignalState(void *context, SignalState state);
```

参数	含义
context	上下文指针
SignalState	信令连接状态 • kTrroReady = 0, 首次连接建立成功 • kTrroLost = 1，连接断开，内部会进行自动重连 • kTrroReup = 2，自动重连成功 • kTrroKickout = 3，被同名用户剔除 • kTrroAuthFailed = 4，用户名或者密码错误

视频连接状态回调

使用说明：可用于监听拉取的视频流连接状态。

```
extern "C" TRRO_EXPORT void TRRO_registerOnState(void* context,
TRRO_OnState * callback);

//视频连接状态
enum TrroState {
    kDisconnect = 0,
    kConnecting = 1,
    kConnected = 2,
    kDisconnecting = 3,
};

/*
* @name :  TRRO_OnState
* @brief : 视频流连接状态回调
* @input : context      回调上下文指针，返回注册回调函数时传入的 context
*          gwid         视频帧的来源现场设备 ID
*          stream_id    视频帧的来源现场设备流 ID
*          conn_fd      接收视频流的句柄标识
*          state        视频流连接状态，对应 TrroState 状态
* @return : void
*/
```

```
typedef void STD_CALL TRRO_OnState(void* context, const char* gwid, int
stream_id, int conn_fd, int state);
```

参数	含义
context	上下文指针
gwid	视频流对应的现场设备 ID
stream_id	视频流对应现场设备上的视频流 ID
conn_fd	视频流对应的远端设备接收句柄
state	视频连接状态 • kDisconnect = 0, 视频连接断开 • kConnecting = 1, 视频连接正在连接中 • kConnected = 2, 视频连接成功 • kDisconnecting = 3, 视频连接正在断开中

远端音视频传输状态回调函数

使用说明：可用于监听远端设备和现场设备的音视频流传输指标。

```
extern "C" TRRO_EXPORT void TRRO_registerOnMediaState(void* context,
TRRO_OnMediaState * callback);
extern "C" TRRO_EXPORT void TRRO_registerOnMediaStateInfo(void* context,
TRRO_OnMediaStateInfo * callback);
extern "C" TRRO_EXPORT void TRRO_registerOnAudioMediaStateInfo(void
*context, TRRO_OnAudioMediaStateInfo * callback);


/* TRRO_OnMediaStateInfo 使用该结构体 包含媒体传输的详细信息
* fps          每秒帧数目
* bps          每秒数据量
* rtt          UDP ping 网络往返时间
* decd         解码耗时ms
* jitter       接收抖动值
* packets_lost 丢包率（0-255），丢包百分比 = packets_lost / 255 * 100%
* packets_received 总接收数目
* stun          穿网模式 0: host, 1: srflx, 2: prflx, 3: relay
* 单位(%) 视频播放卡顿率 = 视频播放的累计卡顿时长 / 视频播放的总时长。
```

```
* lag_k100      视频帧间隔超过100ms视为卡顿，并将其计入视频播放的累计卡顿时长，最终
计算获得的卡顿率。
* lag_k150      视频帧间隔超过150ms视为卡顿，并将其计入视频播放的累计卡顿时长，最终
计算获得的卡顿率。
* duration      视频播放的总时长
*/
struct TrroMediaState {
    int fps;
    int bps;
    int rtt;
    long long decd;
    int jitter;
    long long packets_lost;
    long long packets_received;
    int stun;
    double lag_k100;
    double lag_k150;
    long long duration;
};


/*
* @name :  TRRO_OnMediaState（depreciated）
* @brief : 视频流传输状态回调
* @input : context      回调上下文指针，返回注册回调函数时传入的 context
*          conn_fd      接收视频流的句柄标识
*          fps          每秒帧数目
*          bps          每秒数据量
*          rtt          UDP ping 网络往返时间
*          decd         解帧耗时ms
*          jitter       接收抖动值
*          packets_lost 丢包率（0-255），丢包百分比 = packets_lost / 255 *
100%
*          packets_received 总接收数目
*          stun          穿网模式 0：host，1：srflx，2：prflx，3：relay
* @return : void
*/
typedef void STD_CALL TRRO_OnMediaState(void* context, int conn_fd, int
fps, int bps, int rtt, long long decd, int jitter,
    long long packets_lost, long long packets_received, int stun);


/*
```

```
* @name :  TRRO_OnMediaStateInfo(Recommended)
* @brief : 视频流传输状态回调
* @input : context      回调上下文指针，返回注册回调函数时传入的 context
*          conn_fd      接收视频流的句柄标识
*          mediastate   媒体相关的详细信息
* @return : void
*/
typedef void STD_CALL TRRO_OnMediaStateInfo(void* context, int conn_fd,
TrroMediaState mediastate);
```

参数	含义
context	上下文指针
conn_fd	视频流对应的远端设备接收句柄
TrroMediaState	媒体状态结构体 • fps: 帧率 • bps: 码率，单位bps • rtt：远端设备到服务器的网络rtt，单位ms • jitter：远端设备到服务器的网络抖动 • packets_lost：远端设备丢包率（0-255），丢包百分比=lost/255*100% • packets_received: 接收到包数量 • stun： stun连接模式 • lag_k100：100ms卡顿率 • lag_k150： 150ms卡顿率 • duration： 视频持续时长，单位ms

现场媒体传输状态回调

```
extern "C" TRRO_EXPORT void TRRO_registerFieldSideNetworkState(void*
context, TRRO_onFieldSideMediaState * callback);

/**
* @name : TRRO_onFieldSideMediaState
* @brief : field端网络状态信息，server模式时有意义
* @input :        context            上下文指针
*                 gwid               field端ID
*                 stream_id          field端流ID
*                 fps                field端推送帧率
```

```
*                bps                 field端推送码率
*                rtt                 field端封包来回时间
*                jitter              field端抖动
*                lost                field端丢包率(0-255), ，丢包百分比 =
lost/255 * 100%
* @return : void
*/
typedef void TRRO_onFieldSideMediaState(void* context, const char* gwid,
int stream_id, int fps, int bps, int rtt, int jitter, int lost);
```

参数	含义
context	上下文指针
gwid	现场设备 ID
stream_id	现场设备的视频流 ID
fps	帧率
bps	码率，单位 bps
rtt	现场设备到服务器的网络 rtt，单位 ms
jitter	现场设备到服务器的网络抖动
lost	现场设备丢包率(0-255), ，丢包百分比 = lost/255 * 100%

诊断类接口

诊断报告

使用说明：可用于问题诊断，需在调用 TRRO_connect 接口后调用对目标进行诊断。详情请查看 会话连接诊断 。

```
extern "C" TRRO_EXPORT void TRRO_registerOnDiagReport(void* context,
TRRO_OnDiagReport * callback);


/*
* #name :    TRRO_OnDiagReport
* @brief :   诊断信息回调
* @input :   context       回调上下文指针，返回注册回调函数时传入的 context
*           gwid          发起诊断的目标现场设备 ID
*           type          1 成功 , < 0 相关错误信息
```

```
*            json          回调诊断详细信息 json 格式
* @return : void
*/
typedef void STD_CALL TRRO_OnDiagReport(void* context, const char* gwid,
int type, const char* json);
```

日志回调

使用说明：可用于监听 SDK 回调的日志内容。

```
extern "C" TRRO_EXPORT int TRRO_registerLogCallback(void* context,
TRRO_OnLogData * callback);

//日志等级
enum TrroLogLevel {
    LOG_INFO = 1,
    LOG_WARNING = 2,
    LOG_ERROR = 3,
};

/*
* @name :   TRRO_OnLogData
* @brief :  日志回调
* @input :  context        回调上下文指针，返回注册回调函数时传入的 context
*           msg            日志内容
*           level          日志级别,参考枚举 TrroLogLevel
* @return : void
*/
typedef void STD_CALL TRRO_OnLogData(void* context, const char* msg, int
level);
```

音频控制接口

静音现场设备

使用说明：将目标现场设备静音，仅支持 server 模式。

```
/*
* @name :  TRRO_audioMute
* @brief : Mute/unMute 目标现场设备的音频 仅 Server 模式
* @input : gwid     目标现场设备 ID
*          mute     true 静音， false 取消静音
```

```
* @return : 成功 1 失败 <= 0
*/
extern "C" TRRO_EXPORT int TRRO_audioMute(const char* gwid, bool mute);
```

参数	含义
gwid	目标现场设备 ID
mute	• true：静音 • false：取消静音

图像转换接口

I420转ARGB

使用说明：用于图像数据的格式转换，将 yuvi420格式转为 ARGB，支持分辨率 resize。

```
/*
* @name :  TRRO_I4202ARGB
* @brief : YUV I420 图像格式转 ARGB 图像格式，转换时带 resize 功能
* @input : argb      存放输出A RGB 图像数据的指针，ARGB 的 int 表示:  0XFF(RR)
(GG)(BB)  ,   RR GG BB 为对应的 rgb 的值
*          yuvI420  输入 yuvI420 图像数据指针
*          src_width  输入图像宽度
*          src_height  输入图像高度
*          dst_width  输出图像宽度
*          dst_height  输出图像高度
* @return : 成功 1 失败 <= 0
*/
extern "C" TRRO_EXPORT int TRRO_I4202ARGB(int* argb, char* yuvI420, int
src_width, int src_height, int dst_width, int dst_height);
```

参数	含义
argb	输出的 argb 数据
yuvI420	输入的 yuvi420格式数据
src_width	输入图像宽度

src_height	输入图像高度
dst_width	输出图像宽度
dst_height	输出图像高度

软件信息接口

获取版本号

```
//获取 SDK 版本号
const char * TRRO_getVersion();
```

Android Java API

最近更新时间：2025-06-30 19:17:11

API 描述

Android 平台提供 Native C API（C/C++ 远端设备 SDK API）和 Java API。本文档为 Java API， 适用于 Android 系统，使用时请先引用 Android SDK aar 包。

SDK 集成说明

开发环境要求

- Android Studio 2.0+。
- Android 5.0（SDK API 21）及以上系统。

SDK 引用及依赖配置

1. 将 AAR 文件复制到工程目录下（例如：app/libs）
2. 在 gradle 文件中增加依赖配置：

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.aar']) //添加
SDK 依赖
    implementation 'com.google.code.gson:gson:2.8.9'    //添加gson依赖
}
```

权限配置

```
<uses-permission android:name="android.permission.INTERNET" />
```

混淆配置

```
-keep class com.tencent.trro.**{*;}
```

API 概览

TRRORemote

SDK 主入口。

API 方法	描述
getInstance	获取 SDK 单例。
setListener	注册 SDK 事件监听。
initWithOptions	使用 TrroOptions 配置类启动 SDK。
initWithJson	使用 JSON 字符串配置启动 SDK。
connect	连接现场设备视频流。
getGwList	获取在线现场设备列表（单线程调用，同步阻塞）。
disconnect	关闭指定视频连接。
disconnectAll	关闭所有视频连接。
sendControlData	给现场设备发送控制数据。
externalAudioData	发送自定义音频数据，需要配置开启使用外部音频 "audio_external": 1。
audioMute	现场设备音频设置。
RequestPermission	向现场设备发出权限请求。
fieldDeviceEncodeConfig	更新现场设备目标视频流编码参数。
startRecord	开始录制视频。
stopRecord	停止录制视频。
destroy	释放 SDK 资源。
getSDKVersion	获取 SDK 版本号。

getInstance

获取 SDK 单例：

```
public static TrroRemote getInstance()
```

setListener

注册 SDK 事件监听，可获取网关连接状态、媒体信息等，详细参见本文档 TrroRemoteListener 说明。

```
public void setListener(TrroRemoteListener listener)
```

initWithOptions

使用 TrroOptions 配置类启动 SDK，接口为同步阻塞模式，等待信令连接成功后返回。返回值：成功 1 失败 <= 0，详细错误码参见 错误码及排查 。

```
//基于ApplicationContext和TrroOptions初始化
public int initWithOptions(Context context, TrroOptions trroOptions)
```

参数	描述
context	Android ApplicationContext
TrroConnectParam	初始化配置，参见本文档 TrroOptions 说明。

initWithJson

使用 JSON 配置启动SDK,接口为同步阻塞模式，等待信令连接成功后返回。返回值：成功 1 失败 <= 0，详细错误码参见 错误码及排查 。

```
public int initWithJson(Context context, String jsonStr)
```

参数	描述
context	Android ApplicationContext
jsonStr	初始化配置 JSON 字符串，详细参见 远端设备配置说明 。 **注意：**证书已内置 SDK 中，不需要再配置 certificate 字段。

connect

发起单路视频连接，可多次调用连接不同流。异步模式，根据 onState 状态回调确认视频连接成功，连接多路视频流也可以用下面多路视频流连接接口。

```
public void connect(String gwid, TrroConnectParam connectParam)
```

参数	描述

gwid	现场设备 ID（要拉取视频流的推流设备 ID），公有云格式为 projectId/deviceId，私有云格式为 deviceId。
TrroConnectParam	连接参数，参见本文档中 TrroConnectParam 说明。

connect

发起多路视频连接，异步模式，根据 onState 状态回调确认视频连接成功。

```
public void connect(String gwid, List<TrroConnectParam>
connectParamList)
```

参数	描述
gwid	目标连接的现场设备 ID，需添加 projectid 前缀。 • 格式：projectid/gwid • 例如：ebesyladhdhs8txj/cam_logitech
connectParamList	连接参数列表，配置每一路视频参数。

getGwList

获取在线现场设备设备列表，接口为同步阻塞模式。返回值参见本文档 GateWayList 。

```
public GateWayList getGwList()
```

getGwInfo

获取网关详细信息，接口为同步阻塞模式。返回值网关详细信息对象。

```
public GwInfo getGwInfo(String gwid)
```

disconnect

关闭指定视频连接， 返回值：成功 1 失败 <= 0。

```
public int disconnect(int connFd)
```

参数	描述

connFd	要关闭视频连接对应的接收句柄。

disconnectAll

关闭所有视频连接，返回值：成功 1 失败 <= 0。

```
public int disconnectAll()
```

sendControlData

给现场设备发送控制数据。消息限制为每条大小1KB，每秒<100条，每秒总大小<100KB。

```
public int sendControlData(String gwid, byte[] msg, int qos)
```

参数	描述
gwid	目标连接的现场设备 ID。
msg	消息内容，二进制透传。
qos	消息传输 qos 。 • 0 超低延迟高频传输， 适合高频远控消息，消息大小<1KB。 • 1 可靠传输，网络不好时可能产生消息堆积，适合低频重要消息，消息大小<10KB。

externalAudioData

外部输入音频数据，需要配置"audio_enable": 1 并开启外部音频输入"audio_external": 1，此时SDK音频采集会关闭并使用外部输入的音频数据。

```
public int externalAudioData(String gwid, byte[] data, int channel, int
sampleRate)
```

参数	描述
gwid	目标连接的现场设备 ID。
data	音频 PCM 数据，16bit采样，10ms长度。
channel	音频声道数。
sampleRate	音频采样率。

audioMute

现场设备音频设置, 返回值：成功 1 失败 <= 0。

```
public int audioMute(String gwid, boolean mute)
```

参数	描述
gwid	目标连接的现场设备 ID。
mute	• true：静音。 • false：取消静音。

requestPermission

向网关发出权限请求,返回值：成功 1 失败 <= 0

```
public int requestPermission(String gwid, int permission)
```

参数	描述
gwid	目标连接的现场设备 ID。
permission	• 0 guest，只有观看权。 • 1 master， 完全控制权限。

fieldDeviceEncodeConfig

更新现场设备目标视频流编码参数, 返回值：成功 1 失败 <= 0

```
public int fieldDeviceEncodeConfig(String gwid, int streamId, String
encodeConfig)
```

参数	描述
gwid	目标连接的现场设备 ID。
streamId	目标视频流编号。
encodeConfig	待更新的编码参数 ，JSON 格式字符串，缺省字段将保持当前值不进行更新，下面是更新支持的编码参数示意。

	`{ "fps": 30, "encode_width": 1920, "encode_height": 1080, "bps": 3000, "min_fps": 30, "min_bps": 1800, "force_min": 0, "min_width": 1920 }`

startRecord

开始录制视频。

参数	描述
connFd	视频接收句柄。
filePath	视频储存路径。

stopRecord

停止录制视频。

参数	描述
connFd	视频接收句柄

destroy

释放SDK资源，在程序退出时可以主动释放 SDK 资源，并退出用户登录。

```
public void destroy()
```

getSDKVersion

获取 SDK 版本号信息。

```
public static String getSDKVersion()
```

TrroRemoteListener

SDK 事件监听回调接口。

回调API	描述
onSignalState	信令连接状态回调（与登录状态相关）。
onState	视频流连接状态回调。
onMediaState	视频流传输状态回调。
onAllLatencyCallback	时延回调。
onReportData	接收来自现场设备信息回调。
onFieldSideMediaState	现场设备视频流传输网络状态回调。
onRemoteFrameData	解码后视频图像回调（仅开启实验性参数使用强制软解时）。
onRemoteEncodedFrame	编码帧回调。
onError	错误信息回调。
onOperationPermissionState	回调现场设备操控权限状态通知。
onRemoteMixAudioFrame	接收的音频混流数据回调。

onSignalState

信令连接状态回调

```
void onSignalState(int signalState)
```

参数	描述
signalState	信令连接状态。 • 0 连接建立成功（初始化首次连接成功）。 • 1 连接断开，内部会进行自动重连。 • 2 自动重连成功 。 • 3 被踢下线 （已有相同用户登录）。

	• 4 用户名或者密码错误。

onState

视频流连接状态回调。

```
void onState(String gwid, int streamId, int connFd, int state)
```

参数	描述
gwid	现场设备 ID。
streamId	现场设备视频流 ID。
connFd	视频接收句柄。
state	视频流连接状态。 • 0：未连接 • 1：连接中 • 2：已连接 • 3：正在断开连接

onMediaState

视频流传输状态回调

```
void onMediaState(int connFd, int fps, int bps, int rtt, long decd, int
jitter, long packetsLost, long packetsReceived, int stun)
```

参数	描述
connFd	连接 FD。
fps	每秒帧数。
bps	每秒数据量。
rtt	ping 网络往返时间。
decd	解码耗时，单位：毫秒。
jitter	接收抖动值。
packetsLost	丢包， 丢包率 = packetsLost / 255.0 * 100%。

packetsReceived	总接收包数。
stun	穿网模式。 • 0：host • 1：srflx • 2：prflx • 3：relay

onAllLatencyCallback

时延回调

```
void onAllLatencyCallback(String gwid, int connFd, long latency1, long latency2, long videotime, int rcct)
```

参数	描述
gwid	现场设备 ID。
connFd	视频接收句柄。
latency1	SDK 估计的视频传输时延 ms。
latency2	视频传输 + 消息下发时延 ms，可认为是远程操控相对本地操控的时延。
videotime	当前视频帧的现场设备采集时间戳。
rcct	消息往返时延ms。

onReportData

接收来自现场设备消息回调

```
void onReportData(String gwid, byte[] msg, int len)
```

参数	描述
gwid	现场设备 ID。
msg	消息内容，二进制透传。
qos	消息来源传输 qos。 • 0：低延迟高频传输。

	• 1：可靠传输。

onFieldSideMediaState

现场设备视频流传输状态回调。

```
void onFieldSideMediaState(String gwid, int streamId, int fps, int bps,
int rtt, int jitter, int lost);
```

参数	描述
gwid	现场设备 ID。
streamId	视频流 ID。
fps	每秒帧数。
bps	每秒数据量，单位 kbps。
rtt	网络往返时间 ms。
jitter	网络延迟抖动值。
lost	丢包， 丢包率 = packetsLost / 255.0 * 100%。

onRemoteFrameData

原始视频图像回调，可获取视频 YUV 数据（CPU 内存），用于自定义渲染或者图像处理等场景（仅在开启强制软解时生效）。考虑 Android 平台性能加速优化，一般优先推荐通过 connect 接口的视频输出纹理（GPU内存）来获取原始视频数据。

```
void onRemoteFrameData(String gwid, int streamId, int connFd, ByteBuffer
data, int width,int height, long videoTime);

//需要开启强制软解，connect时不使用视频输出纹理
ExperimentOptions experimentOptions = new
ExperimentOptions.Builder().enableSoftwareDecoder(true).printDecodeTime(
false).build();
TrroRemote.getInstance().callExperimentApi(experimentOptions);
```

参数	描述
gwid	现场设备 ID。

streamId	现场设备流 ID。
connFd	视频接收句柄。
data	视频帧数据，YUV I420格式。
width	视频帧宽。
height	视频帧高。
videoTime	当前视频帧时间戳。

onRemoteEncodedFrame

视频编码数据帧回调。

```
void onRemoteEncodedFrame(String gwid, int streamId, int connFd, byte[]
data, int len,int trroCodec, boolean isKeyFrame, int width, int height,
long videoTime);
```

参数	描述
gwid	现场设备 ID。
streamId	现场设备流 ID。
connFd	视频接收句柄。
data	编码的视频图像帧数据。
len	编码的视频图像帧的长度。
trroCodec	编码的 codec 类型。 • 0：H264 • 1：H265
isKeyFrame	是否是关键帧。
width	视频帧宽（硬解有效）。
height	视频帧高（硬解有效）。
videoTime	当前视频帧时间戳。

onError

错误信息回调。

```
void onError(int errorCode, String errorMsg)
```

参数	描述
errorCode	错误码。
errorMsg	错误信息描述。

onOperationPermissionState

回调现场设备操控权限状态通知，只有拥有 master 权限时才能发送消息给对应现场设备。

```
void onOperationPermissionState(String fieldDevId, int selfPermission,
String masterDevId)
```

参数	描述
fieldDevId	来源现场设备 ID。
selfPermission	本设备当前的操控权限 • 0 是 guest，只有观看权限。 • 1是 master，拥有完全控制权限。
masterDevId	拥有 master 权限的远端设备 ID。

onRemoteMixAudioFrame

接收的音频混流后 PCM 数据回调, 需要配置开启音频接收"audio_receive": 1 并使用外部音频播放（SDK 不播放音频）"audio_play": "outside"。

```
void onRemoteMixAudioFrame(byte[] data, int length, int channel, int
sampleRate)
```

参数	描述
data	PCM 音频数据，16bit采样，10ms长度。
length	音频数据长度。
channel	音频声道数。

sampleRate	音频采样率。

TrroOptions

配置初始化参数，字段说明参见 远端设备配置说明 。

TrroConnectParam

视频流连接配置参数。

属性	描述
streamId	现场设备视频流的 ID 数组， 现场设备视频流 ID 从0开始， 最大值为现场设备支持的 device_streams 数量 -1。
connFd	接收视频流的句柄标识，自行编号，取值从0开始。
videoView	视频流输出到视频流显示控件。
previewTexture	视频流输出到 SurfaceTexture，可用于离屏渲染，优先建议使用 Surface。
surface	视频流输出到 Android Surface 实例，可用于自定义渲染等。

> **注意：**
> videoView、previewTexture 和 surface 同时只能设置一个。如果都不设置则 SDK 只回调对应视频流的视频帧数据，不会进行渲染（含离屏渲染）。

TXCloudVideoView

视频流显示控件，可设置隐藏或展示。

API	描述
setFillMode	设置视频画面填充模式。 • 0：填充（画面可能会被拉伸裁剪） • 1：适应（显示完整画面，可能会有黑边） 默认值：1
useTextureView	设置视频显示使用 TextureView 或 SurfaceView。 • true：使用 TextureView • false：使用 SurfaceView 默认值：false

GateWayList

现场设备列表。

属性	描述
ret	0 获取成功。
msg	获取结果描述。
count	在线设备总数。
gateways	在线设备列表，GateWay 数组。

GateWay

现场设备。

属性	描述
name	设备名称。
type	设备类型。
status	设备状态，ready 在线等待 / connected 会话中。
streams	视频流数量。
timestamp	时间戳。
version	设备 SDK 版本。

示例代码

SDK 初始化

```
//采用trroOptions初始化，也可采用json接口初始化（注意android平台不需要
certificate和output_path配置）
TrroOptions trroOptions = new TrroOptions.Builder()
                        .deviceId(mDeviceId)
                        .projectid(mProjectId)
                        .password(mPassword)
                        //日志级别，0：关闭，1，输出部分运行日志，2：输出详细运行
日志
                        .logEnable(2)
                        .cloudMode("public")
                        .deviceName(mDeviceName)
                        .setMaxStreams(8).build();
```

```
//启动SDK，阻塞接口，建议在工作线程中调用
TrroRemote.getInstance().initWithOptions(getApplicationContext(),
trroOptions);
```

连接视频流

```
//在layout文件中，配置视频流显示控制
<com.tencent.trro.rtmp.ui.TXCloudVideoView
        android:id="@+id/txcvv_1"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
/>

//第1路视频连接信息配置
TrroConnectParam connectParam1 = new TrroConnectParam.Builder()
                .connFd(0)  //connFd，分配一个唯一ID，建议从0开始
                .streamId(0) //现场设备视频流的ID数组， 现场设备视频流ID从0开始
                .videoView(txCloudVideoView1) //视频流显示控件
                .build();
//第2路视频连接信息配置
TrroConnectParam connectParam2 = new TrroConnectParam.Builder()
                .connFd(1)
                .streamId(1)
                .videoView(txCloudVideoView2)
                .build();

List<TrroConnectParam> trroConnectParamList = new ArrayList<>();
       trroConnectParamList.add(connectParam1);
       trroConnectParamList.add(connectParam2);
//调用connect方法连接现场设备视频流

TrroRemote.getInstance().connect(mGateWayId,
trroConnectParamList);
```

事件回调监听

```
//在Activity生命周期函数中设置监听器，比如onStart方法设置监听，在onStop方法取消监
听
TrroRemote.getInstance().setListener(this);
```

资源释放

```
//在退出时释放资源，比如Activity的onDestroy方法中调用
TrroRemote.getInstance().destroy();
```

远端设备配置说明

最近更新时间：2025-04-14 10:46:02

JSON 配置文件样例

以下是基本 JSON 配置举例。可使用 配置生成工具（选择远端设备）生成配置。

```
配置文件简述
文件名：config.json
文件位置：${workspace}/config.json
文件类型：json
注意："//"后注释在使用时要删除。最小节点如下
{
    "device_id":"dev1",  //修改为控制台中创建的远端设备ID
    "device_name":"vin99",
    "max_streams":8,
    "cloud_mode":"public",
    "certificate":"./device.pem",
    "projectid" : "xxxxx", //修改为控制台中创建的项目ID
    "password": "xxxxx" //修改为控制台中创建的密码
}
```

JSON 配置节点含义

节点名	类型	含义	默认值	备注	参数范围
device_id	字符串型（String）	设备 ID，用于注册到服务器的唯一编码	-	必选	-
device_name	字符串型（String）	设备名称，用于辨识设备信息	-	必选	-
max_streams	数字型（Number）	远端设备连接的最大流数目	-	必选	无，配套 Demo 建议 >=8
projectid	字符串型（String）	项目 ID	-	公有云必填	-

cloud_mode	字符串型（String）	云模式	-	必填	• public：公有云 • private：私有云
password	字符串型（String）	设备密钥	-	公有云必填	-
certificate	字符串型（String）	服务器公钥证书	-	公有云必填，device.pem 文件路径	-
force_login	数字型（Number）	是否强制登录	0	同名 ID 登录是否剔除之前的设备登录	• 0：已有相同 ID 登录时，当前设备登录被剔除 • 1： 已有相同 ID 登录时，当前设备会强制登录，将之前设备剔除
server_ip	字符串型（String）	TRRO 服务端 IP	-	公有云无需填写， 私有化必填	-
server_port	数字型（Number）	TRRO 服务端端口号	-	公有云无需填写，私有化必填	-
log_enable	布尔型（Boolean）	日志开关	1	可选	• 0：不使能 • 1：info日志 • 2：info+debug日志
sdk_mode	字符串型（String）	流传输模式	"server2"	可选	公有云需用 server2
rtc_server_ip	字符串型（String）	媒体服务端 IP	-	公有云无需指定，私有化必填	-
rtc_server_port	数字型（Number	媒体服务端端口号	-	公有云无需指定，私有化必填	-

	）				
audio_enable	数字型（Number）	音频采集使能	0	可选	• 0：不使能 • 1：使能
audio_external	数字型（Number）	外部音频输入	0	可选，使用外部音频输入替代SDK音频采集时启用	• 0：不使能 • 1：使能
audio_receive	数字型（Number）	音频接收使能	1	可选	• 0：不使能 • 1：使能
audio_play	数字型（Number）	音频播放设备	-	可选，用于指定播放设备；使用音频回调进行自定义播放时配置“outside”	参见 语音对讲
auto_permission	数字型（Number）	是否自动请求权限	1	可选	• 0：应用请求权限 • 1：自动请求权限
play_mode	数字型（Number）	观看模式	0	可选	• 0：超低时延观看 • 1：流畅观看
decode_hw_acc	数字型（Number）	是否硬件解码优先	0	可选	• 0：软解 • 1：硬解优先
file_path	字符型（String）	保存文件路径	-	可选，配置时开启录制，需保证配置路径已存在	-
min_port	数字型（Number）	视频传输本地端口段最小值	50000	可选	-
max_port	数字型（Number）	视频传输本地端口段最大值	50100	可选	-

错误码及排查

最近更新时间：2024-02-07 10:08:22

错误码的定义位于 trro_remote.h 头文件中。函数返回值为错误码负值，如错误码为16777215，返回值为-16777215。错误码具体含义及排查方法如下表所示：

模块	错误码定义	错误码16进制	错误码10进制	排查方向
通用错误码	TRRO_SUCCED	0x00000001	1	成功返回值
	TRRO_COMMON_ERROR	0x00FFFFFF	16777215	参数输入或初始化状态异常
	TRRO_GWID_NOT_FOUND	0x00000010	15728640	不存在的网关 ID
	TRRO_PARAM_INVALID	0x00F00000	16777215	参数非法
配置文件加载模块（检查配置文件）	TRRO_CONFIG_ERROR	0x01FFFFFF	33554431	通用配置类错误
	TRRO_CONFIG_PARSE_FAILED	0x01000002	16777218	配置文件的格式解析异常，建议使用在线 json 校验工具进行检查。
	TRRO_CONFIG_ILLEGAL	0x01000003	16777219	配置文件的参数非法，建议检查最小节点是否覆盖
	TRRO_CONFIG_UNEXIST	0x01000004	16777220	配置文件不存在
	TRRO_CONFIG_CER_FAILED	0x01000005	16777221	公有云模式下证书错误
初始化模块（检查配置文件和接口入参是否正常）	TRRO_INIT_ERROR	0x02FFFFFF	50331647	通用初始化失败，未知原因
	TRRO_INIT_INPUT_ILLEGAL	0x02000002	33554434	输入参数非法，需检查入参是否正常
	TRRO_INIT_PARSE_fAILED	0x02000003	33554435	初始化参数解析失败，检查配置文件最小节点是否正常

	TRRO_INIT_CREAT_MEDIAMODE_FAILED	0x02000004	33554436	媒体模块创建失败
	TRRO_INIT_LICENSE_CHECK_FAILED	0x02000005	33554437	LICENSE 校验失败
	TRRO_INIT_INVALUDE_INPUT	0x02000006	33554438	接口调用入参非法
	TRRO_INIT_MQTT_ABNORMAL	0x02000007	33554439	始化 mqtt 异常
	TRRO_INIT_REPEAT	0x02000008	33554440	重复初始化
信令服务器模块（检查服务器连接是否正常）	TRRO_SIGNAL_ERROR	0x03FFFFFF	67108863	信令模块异常
	TRRO_SIGNAL_REGIST_FAILED	0x03000002	50331650	信令注册失败
	TRRO_SIGNAL_STATUS_ABNORMAL	0x03000003	50331651	连接 MQTT 信令服务异常
	TRRO_SIGNAL_MESSAGE_FAILED	0x03000004	50331652	消息发送失败
	TRRO_SIGNAL_CONNECT_OUTTIME	0x03000005	50331653	连接 MQTT 服务超时
流传输模块（检查网络和防火墙）	TRRO_CONNECT_ERROR	0x05FFFFFF	100663295	连接异常
	TRRO_CONNECT_OUTTIME	0x05000002	83886082	超时断连
	TRRO_CONNECT_RESET	0x05000003	83886083	connect 的时候如果之前有未断开的连接需要重置
诊断工具模块	TRRO_DIAG_STARTED	0x06000000	100663296	诊断工具正在启动中
	TRRO_DIAG_TIMEOUT	0x06000001	100663297	诊断工具获取推流信息超时
	TRRO_DIAG_FAILE	0x06000	10066329	诊断工具处理失败

	D	002	8	
权限配置模块	TRRO_UNAUTHORIZED	0x07000000	117440512	网关未授权无法发送消息
消息通道	TRRO_MESSAGE_BYTE_EXCEED	0x08000002	134217730	发送字节数超出限制