物联网开发平台

录音卡

腾讯云

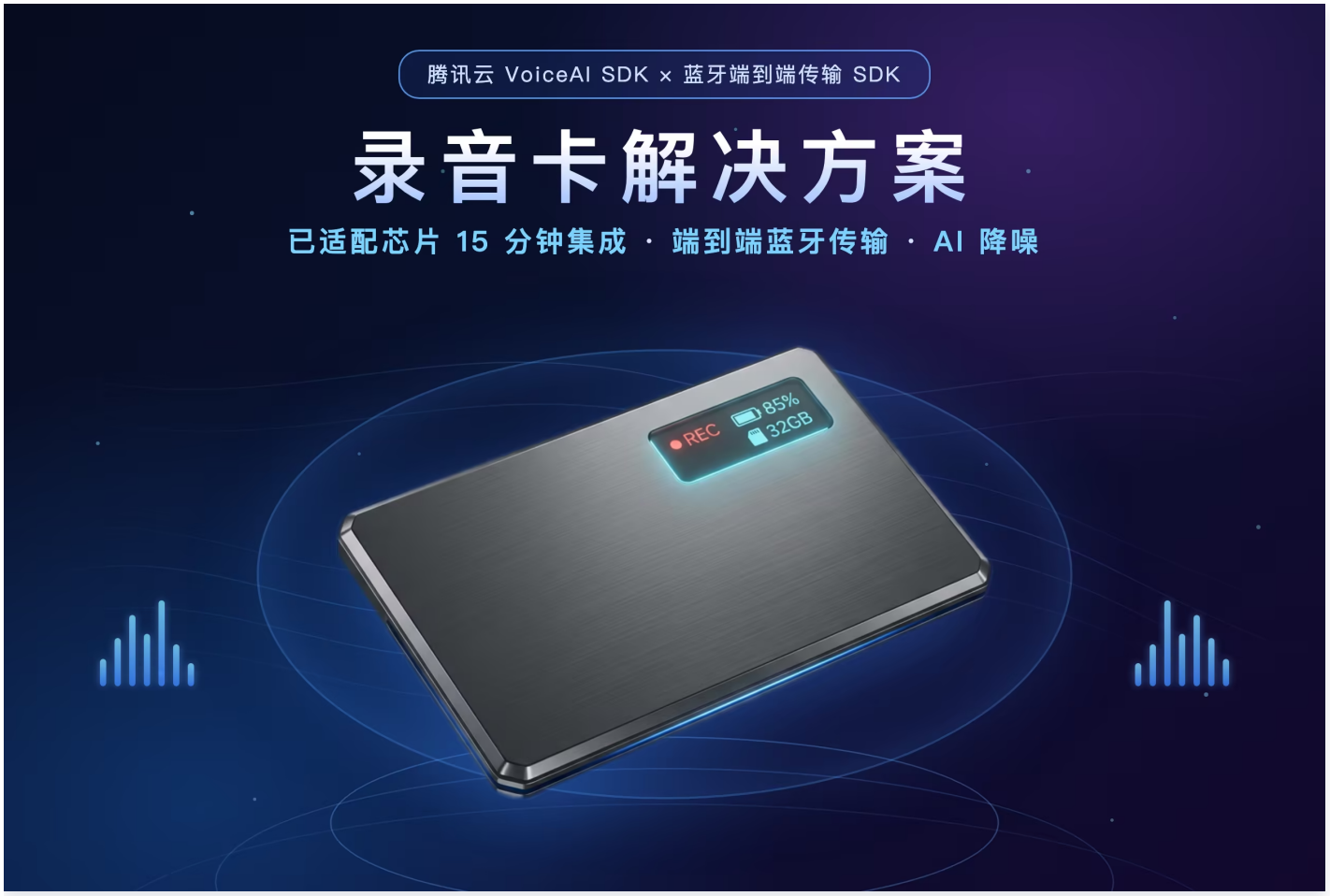
文档目录

录音卡
功能介绍
跑通应用端
跑通设备端

录音卡
功能介绍

最近更新时间：2026-09-09 09:27:01

本方案是面向录音卡类硬件的一站式端到端解决方案：基于腾讯云蓝牙端到端传输 SDK 与 VoiceAI SDK，覆盖「录音设备采集 → 蓝牙链路传输 → App 端 AI 降噪与识别 → ASR 识别 → 结构化文本输出」完整链路。**已适配的芯片15分钟即可完成集成跑通**，输出带发言人和时间轴的结构化文本。



方案优势速览

- **十五分钟集成（已适配芯片）**：针对已完成适配的芯片平台，蓝牙端到端传输 SDK 与 VoiceAI SDK 均提供完整 Demo 与简洁接口，设备端接入蓝牙传输、App 端接入识别链路，15分钟即可完成端到端跑通验证。
- **一整套端到端蓝牙传输方案**：覆盖「录音设备采集 → 蓝牙链路传输 → App 端识别输出」全链路，设备端与端侧 SDK 组合开箱即用，无需自行打通传输与识别两段链路。
- **AI 降噪能力**：VoiceAI SDK 内置端侧 AI 降噪，在识别前抑制环境噪声，保障嘈杂场景下的转写质量。

端到端蓝牙传输方案

录音卡类产品的核心难点在于设备端音频如何稳定、低延迟地到达手机端识别引擎。本方案由腾讯云提供覆盖全链路的 SDK 组合，客户无需自行研发或拼接传输链路：

- **设备端**：蓝牙端到端传输 SDK 在录音设备侧完成音频采集与传输，音频经蓝牙链路实时送达手机。
- **手机端**：App 通过外部 PCM 喂入模式接收设备音频，直接送入 VoiceAI SDK 处理，不依赖手机麦克风采集。
- **全链路对齐**：传输与识别接口已由两个 SDK 组合打通，客户按 Demo 对接即可，无需自行处理链路衔接。

AI 降噪能力

项目集成的 `VoiceAI` SDK 包含客户端 AI 降噪处理能力。其目标不是生成更“悦耳”的声音，而是在音频送入识别引擎前抑制环境噪声、提高语音可懂度，从而减少噪声触发、漏识别和误识别。

能力维度	说明
端侧前处理	降噪位于音频输入与 ASR 之间，可在上云前改善信噪比，减少无效噪声对 VAD 和识别模型的干扰。
AI 模型降噪	SDK 内置多条 AINS 处理路径，包含面向 ASR 和会议场景的处理策略，可按实际声学环境选择。
复杂噪声适配	适用于空调、风扇、键盘、设备运行声等稳态或非稳态背景噪声；实际效果取决于拾音距离、信噪比和设备算力。

识别能力

识别链路基于成熟的 ASR 与说话人分离能力，同时支持录音过程中的流式识别和录音完成后的全量识别：

- **实时流式识别**：按语音片段返回可修订的中间结果和最终结果，支持语音活动检测（VAD），满足边录边看的低时延场景。
- **录音文件识别**：生成高完整度转写结果，返回语句文本、起止毫秒时间和说话人编号，适合会后归档与检索。
- **说话人分离**：实时与录音文件链路均开启，输出「谁在什么时候说了什么」的结构化语句分段。

应用场景

应用场景	核心诉求	方案能力
多人会议	边录边看转写，并在会后快速定位不同发言者的观点。	实时 ASR + 实时说话人分离 + 带时间轴的录音文件识别。
采访与调研	完整保留长时间对话，减少人工听写和区分采访对象的成本。	长录音分片转写 + 语句级时间戳 + 说话人编号。
嘈杂办公环境	降低空调、风扇、键盘等背景噪声对识别效果的影响。	端侧 AI 降噪 + VAD + ASR 友好型音频处理。

功能优势

优势	说明
十五分钟集成	针对已适配芯片，设备端蓝牙传输 SDK 与 App 端 VoiceAI SDK 均提供完整 Demo 与封装接口，15分钟即可完成端到端跑通。
端到端蓝牙传输方案	覆盖设备采集、蓝牙传输、端侧识别输出的完整链路，设备端与端侧能力组合交付，开箱即用。
实时与离线识别互补	实时 ASR 提供低时延反馈，录音文件识别提供完整文本和精确时间轴，分别服务录中和录后场景。
文本与角色同时结构化	识别结果不仅包含文字，还包含发言人编号、片段状态和时间信息，便于直接进入纪要、检索和大模型分析链路。
AI 降噪能力	VoiceAI SDK 内置 AI 降噪能力，可在识别前抑制环境噪声、改善音频质量，降低真实办公和移动环境中的噪声干扰。

跑通应用端

最近更新时间：2026-09-09 09:27:02

本文介绍如何编译并运行录音卡 Android Demo。完成本文步骤后，您可以验证 Demo 的基础界面、BLE 录音卡绑定、实时录音、离线文件同步，以及云端语音识别能力。

前提条件

环境准备

- 安装 Android Studio 2025 及之后的版本。
- 准备一台 Android 8.0 及以上、支持 BLE 的 arm64-v8a 真机。
- 准备已实名认证的腾讯云账号，用于开通 ASR 服务。
- 手机需开启蓝牙，并授予 App 蓝牙与定位权限（Android 12 及以上为「附近的设备」权限），用于扫描和绑定录音卡。

开通 ASR 服务

请参见 TRTC AI 智能语音，创建应用并获取 TRTC SDKAppID 和 SDK 密钥，在后续运行时需要。

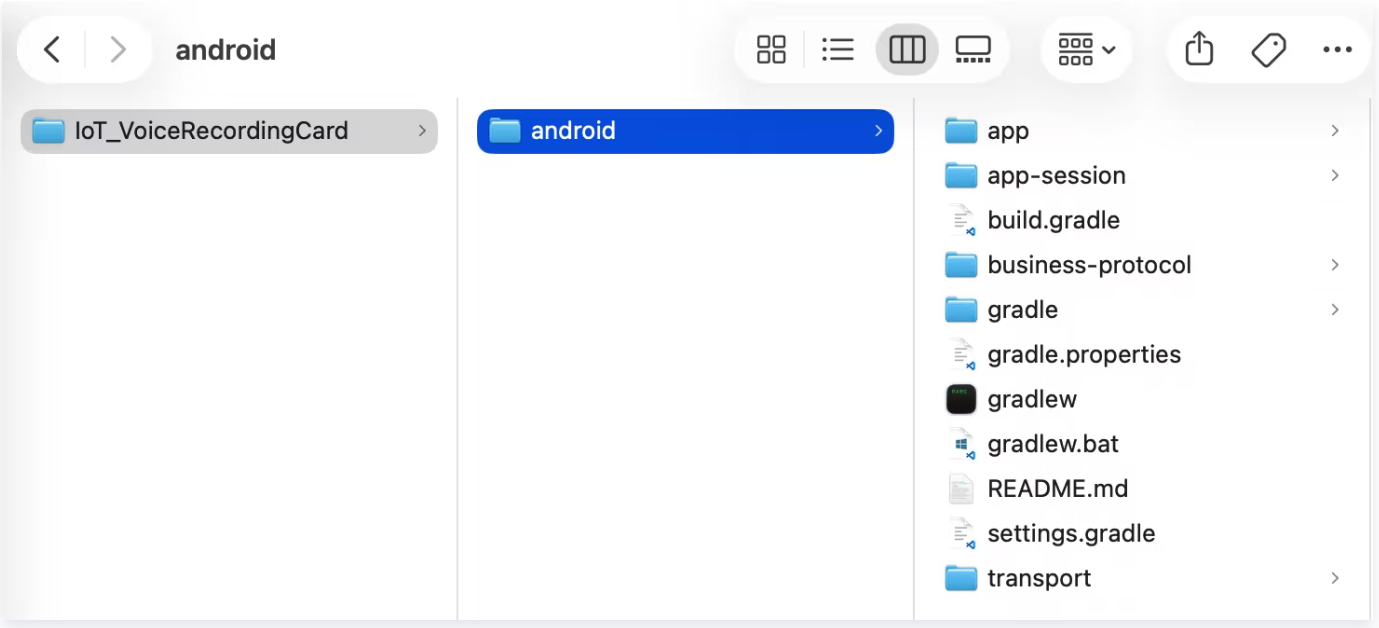
控制台创建应用并获取 SDKAppID 与密钥

下载并配置 Demo

1. 从 GitHub 下载 录音卡 Demo 源码，或者直接在命令行运行以下命令：

```
git clone https://github.com/Tencent-RTC/IoT_VoiceRecordingCard.git
```

2. 通过 Android Studio 打开 `IoT_VoiceRecordingCard/android` 目录：



提示：

Android Studio 推荐配置：Demo 的 Gradle 构建要求 JDK 17，请按下述步骤将 Gradle JDK 改为17，否则首次同步会报错。

1. 打开 Android Studio 设置。
   - **Windows**：File > Settings
   - **Mac**：Android Studio > Preferences
2. 修改 Java 编译器设置。
   - 路径：Build, Execution, Deployment > Build Tools > Gradle
   - 将 Gradle JDK 改为 17（未安装时可直接选择 Android Studio 内嵌的 JetBrains Runtime 17）

3. 将开通服务时获取的 SDKAppID 和 SDK 密钥填入 `android/app/src/main/java/com/recorder/client/trtc/TrtcUserSigConfig.java`：

```
// 填入开通服务时获取的 SDKAppID 和 SDK 密钥
private static final int SDK_APP_ID = 0;
private static final String SECRET_KEY = "";
```

跑通 Demo

步骤1：运行 Demo

1. 开启手机开发者选项与 USB 调试（设置 > 关于手机 > 连续单击版本号7次），连接后允许调试授权。
2. 等待 Gradle 首次同步完成再运行。
3. 选择设备并编译和运行 Demo。

说明：
本 Demo 默认集成了蓝牙能力，因此建议使用真机调试、运行 Demo。

步骤2：扫描并绑定设备

打开 App 扫描周边的录音卡设备，选择目标设备完成绑定：

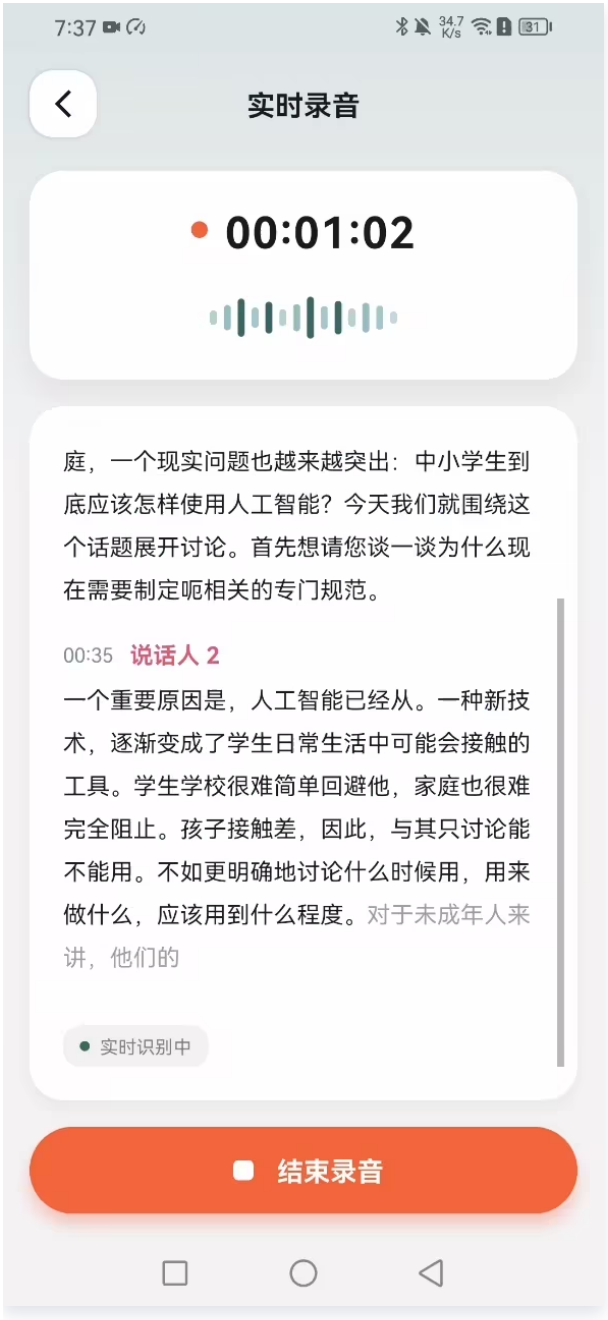
扫描周边的录音卡设备	绑定成功后设备主页

扫描周边的录音卡设备

绑定成功后的设备主页

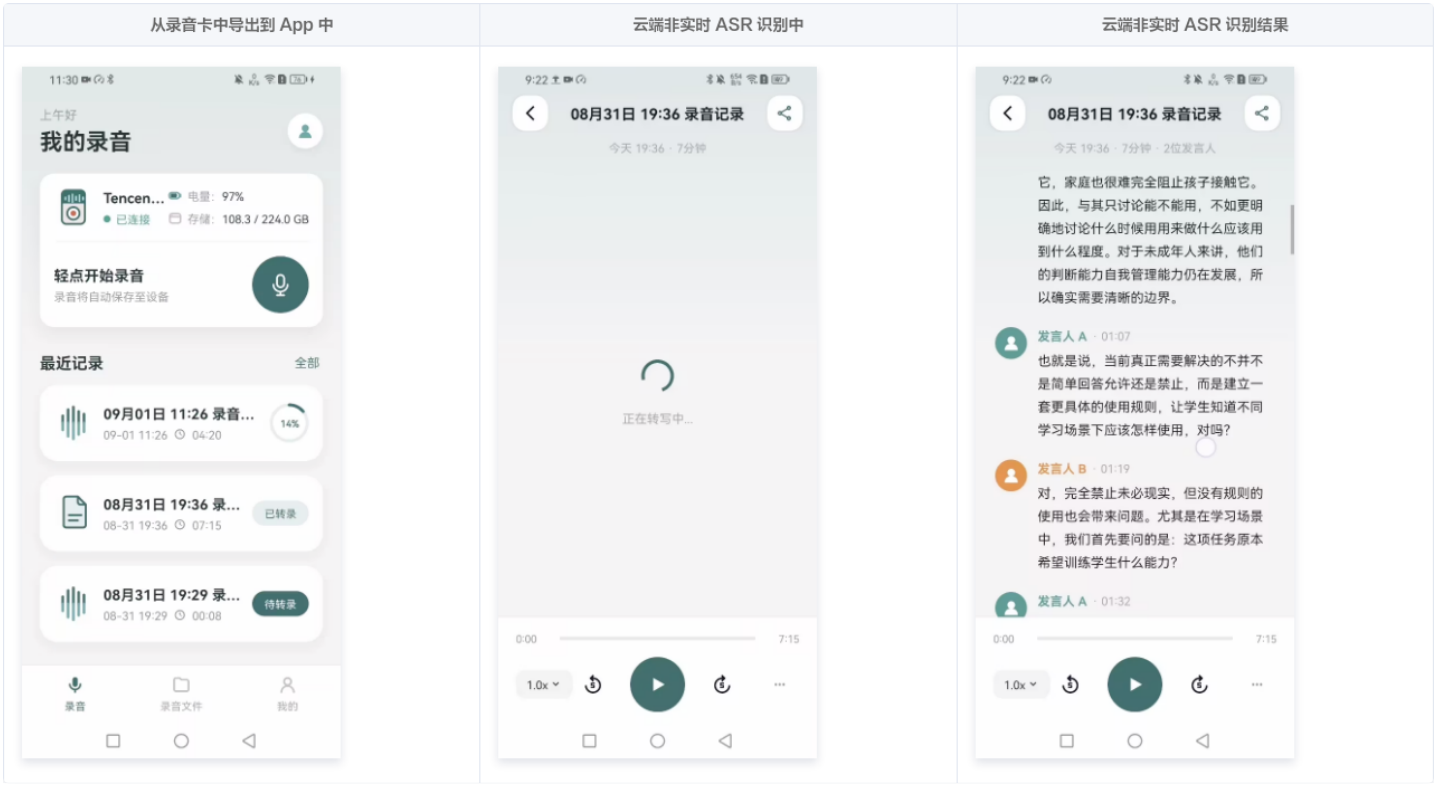
步骤3：实时 ASR 识别

单击设备主页“轻点开始录音”右侧的录音按钮，进入实时 ASR 识别页面：



步骤4：导出录音文件并转录

结束录音后，先将录音文件从录音卡导出到 App，再发起云端非实时 ASR 识别，获得整段录音的完整转写结果：



常见问题

扫描不到录音卡设备

请确认手机蓝牙已开启、App 已获得蓝牙与定位权限（Android 12 及以上为「附近的设备」权限），且录音卡已开机并处于可被发现状态。

Gradle 同步失败

请确认 Gradle JDK 已设置为17（见 下载并配置 Demo 中的提示），并检查网络与代理配置后重新同步。

ASR 无识别结果

请确认 `TrtcUserSigConfig.java` 中的 SDKAppID 与 SDK 密钥填写正确，且对应的应用已在控制台开通 ASR 服务。

后续步骤

跑通 Demo 后，如需将录音与语音识别能力集成到自有 App，请参见 录音卡 Demo 源码仓库 中的集成说明。

跑通设备端

最近更新时间：2026-09-09 09:27:02

本文将介绍如何将录音卡 SDK 集成到您的设备固件中，跑通**录音会话**功能。全文分为三个部分：

- **SDK 下载**：了解支持平台与获取对应平台的 SDK 交付包。
- **SDK 对接指引**：完成 SDK 集成与录音会话能力对接。
- **跑通 Demo**：以 ESP32S3 为例，介绍如何编译烧录，完成设备与 App 联调。

SDK 下载

芯片原厂	芯片型号	SDK 下载地址
乐鑫	ESP32S3	下载

> ⓘ **注意：**
> 如果您的项目使用其它芯片，请填写 芯片适配需求表，我们将尽快适配。
> 芯片需要满足如下要求：
>
> | 前提项 | 要求 |
> | --- | --- |
> | BLE | BLE4.2及以上。 |
> | Flash/RAM | 100KB 以上 Flash，50KB 以上 RAM（SDK 用，不含 BLE 协议栈）。 |
> | 存储介质 | 已挂载的文件系统（如 SD 卡 + FAT），提供录音文件落盘目录。 |
> | 音频能力 | 麦克风采集 + Opus 编码（16kHz / 单声道 / 60ms 帧）。 |
> | 操作系统 | FreeRTOS 或其他 pthread 风格的 RTOS。 |

SDK 对接指引

这部分主要介绍录音卡 API 的对接，方便集成到自己项目中去。
集成方需完成：**调用5个 API、实现3个回调、按固定格式送帧**。

步骤1: 引入录音卡头文件

```
/* 只需要引入这一个头文件即可 */
#include "tc_iot_audio_recorder.h"
```

步骤2: 初始化

```
/* 回调运行在 SDK 内部 message loop 线程：只做入队/置标志，勿阻塞、勿直连硬件 */

static void on_rec_start(const char *params, void *user_data)
{
    /* App 下发开始（或本地 start 被接受）：打开麦克风采集 + Opus 编码器，
     * 之后开始向 SDK 送帧 */
}

static void on_rec_stop(const char *params, void *user_data)
{
    /* App 下发停止（或本地 stop）：请求停止采集，停止送帧 */
}

static void on_rec_message(tc_iot_error_e error_code, const char
*json_msg, void *user_data)
{
    /* 统一事件入口：解析 json_msg 驱动 UI/业务状态，如果没有UI则不需要关注
error_code>0的json_msg */
}

static void on_log(tc_iot_log_level_e level, const char *log)
{
    /* SDK 日志：按需转发到客户日志系统,调试的时候使用，量产只打印Info级别日志即可
*/
}

void recorder_init(void)
{
    tc_iot_audio_recorder_params_s params = {
        .storage_path = "/sdcard/voice",           /* 录音文件落盘目录（SD 卡
挂载点下） */
        .log_level    = TC_IOT_LOG_LEVEL_INFO,     /* 调试阶段可以开Debug */
        .on_log       = on_log,
    };
    tc_iot_audio_recorder_observer_s obs = {
        .on_audio_recorder_start = on_rec_start, // 本地主动触发录音和APP发
起录音都会走这个回调
```

```
        .on_audio_recorder_stop   = on_rec_stop,   // 本地主动触发结束录音或
APP结束录音都会走这个回调
        .on_message               = on_rec_message, // 错误或者事件通知统一走
这个回调
    };
    tc_iot_error_e err = tc_iot_audio_recorder_init(&params, &obs,
NULL);
    if (err != TC_IOT_ERR_SUCCESS) {
        /* 初始化失败 */
    }
    /* init 成功后 SDK 自动开始 BLE 广播，等待 App 连接 */
}
```

步骤3: 设备主动发起录音/结束录音

```
/* 设备本地主动发起录音，如按键开始录音，调用此函数，sdk内部准备好后会通过
on_audio_recorder_start告知，然后才可以推音频流 */
tc_iot_error_e
tc_iot_audio_recorder_start(tc_iot_audio_recorder_operation_cb
callback);
/* 设备本地主动结束录音，如按键结束录音，调用此函数，sdk内部处理完成后会通过
on_audio_recorder_stop告知，此时可以结束推音频流 */
tc_iot_error_e tc_iot_audio_recorder_stop(void);
```

步骤4: 推音频流

当收到 on_audio_recorder_start 的回调后，就可以推送音频。

```
/* 需要在收到on_audio_recorder_start的回调之后才可以推音频流，在这之前推的音频流都
会返回错误，但不影响正常调用 */
  const tc_iot_audio_frame frame = {
    .codec = TC_IOT_AUDIO_CODEC_OPUS, // 当前只接收Opus音频
    .sample_rate = TC_IOT_AUDIO_SAMPLE_RATE_16000,
    .channels = TC_IOT_AUDIO_CHANNEL_MONO,
    .frame_duration_ms = TC_IOT_AUDIO_FRAME_DURATION_MS, // 60ms
    .data = (uint8_t *)opus,
    .data_size = len,
    };
tc_iot_audio_recorder_send_audio(&frame);
```

步骤5: 退出，反初始化

```
/* 停止录音/设备休眠时调用，释放init时的资源，退出SDK */
tc_iot_error_e tc_iot_audio_recorder_deinit(void);
```

跑通 Demo

这节介绍如何跑通录音卡设备端 Demo，我们以 立创实战派 ESP32S3开发版 为例，来介绍录音卡设备端 SDK 的使用。跑通本 Demo 后，您可以验证 BLE 录音卡绑定、实时录音、离线文件同步，以及云端语音识别能力。

前置条件

您已经成功 跑通了应用端 APP，手机已经装好了 APP 并可以正常运行。

下载 Demo

```
git clone https://github.com/Tencent-RTC/IoT_VoiceRecordingCard.git
```

步骤1: 准备 ESP-IDF 编译构建环境

本项目基于乐鑫官方 ESP-IDF 框架开发，版本要求 6.2（或者 master 分支）：

1. 打开 乐鑫官方入门教程 。
2. 按文档安装 ESP-IDF v6.2（master 也可以）及工具链：
   - Windows：下载「ESP-IDF 安装器」一路下一步，完成后桌面会出现 ESP-IDF 终端。
   - macOS / Linux：按文档执行 `install.sh`，之后在终端执行 `. $HOME/esp/esp-idf/export.sh` 激活环境。
3. 验证安装：在 ESP-IDF 终端中执行 `idf.py --version`，应输出 6.2。

> ⓘ **说明：**
> 后续所有命令都在「ESP-IDF 终端」（或已执行 export.sh 的终端）中输入。

步骤2: 编译&烧录

在终端中进入本工程目录 devices/ESP32S3，依次执行：

```
idf.py set-target esp32s3          # 第 1 步：指定芯片型号（仅需一次）
idf.py build                       # 第 2 步：编译（首次会联网下载依赖，需几分
钟）
idf.py -p <串口> flash monitor     # 第 3 步：烧录并打开串口监视器
```

<串口> 的查看方式：

系统	串口	查看方法
macOS	/dev/cu.usbmodem*	ls /dev/cu.usbmodem*
Windows	COMx	设备管理器 → 端口(COM 和 LPT)。
Linux	/dev/ttyACM0	ls /dev/ttyACM*；若无权限执行 sudo usermod -aG dialout $USER 后重新登录。

烧录完成后设备自动重启，串口监视器中可见启动日志（按 **Ctrl+]** 退出监视器）。

步骤3: 上电使用

连接手机 App

1. 用 USB 线给开发板上电，屏幕点亮并显示设备就绪表情。
2. 打开手机 App，在设备列表中选择 **TXVR-XXXX**（XXXX 为屏幕/串口日志中 显示的设备 MAC 地址后两位）。
3. 连接成功后，屏幕表情切换为“已就绪”，即可开始使用。

开始录音

- 手机操作：在 App 中点击“开始录音”，对开发板上的麦克风说话；点击“停止”结束。
- 按键操作：短按板上 BOOT 键 等效开始/停止录音。

录音过程中屏幕显示“聆听中”表情；停止后录音自动上传，文件同时保存在 SD 卡中，可在 App 内回放或下载。