

TVS Device SDK Linux 版本接入说明

拟制： kangrong

日期： 2017 年 8 月

审核：

日期：

深圳腾讯计算机系统有限公司

版权所有 不得复制

版本修订记录

[illegible]

目录

TVS Device SDK Linux 版本接入说明	1
版本修订记录.....	2
1. 简介.....	5
1.1. 名词解释	5
1.2. 环境依赖	5
1.3. 非通用架构支持	6
2. 入门.....	6
2.1. 使用步骤.....	6
2.1.1. 复制依赖库和头文件.....	6
2.1.2. 初始化.....	6
2.1.3. 回调.....	7
2.2. 输入音频要求	8
2.3. 编码	9
3. 各功能模块入门.....	9
3.1. 语音唤醒	12
3.1.1. 功能.....	12
3.1.2. 回调 cmd.....	12
3.1.3. 交互流程.....	12
3.1.4. 数据格式.....	14
3.2. 在线语音识别	15
3.2.1. 功能.....	15
3.2.2. 回调 cmd.....	15
3.2.3. 交互流程.....	16
3.2.4. 数据格式.....	20
3.3. 在线语义	21
3.3.1. 功能.....	21
3.3.2. 回调 cmd.....	21
3.3.3. 交互流程.....	22

3.3.4. 数据格式.....	23
3.4. 在线语音合成.....	24
3.4.1. 功能.....	24
3.4.2. 回调 cmd.....	24
3.4.3. 交互流程.....	25
3.4.4. 数据格式.....	26
4. 附录.....	27
4.1. 账号系统.....	错误!未定义书签。
4.1.1. 功能.....	错误!未定义书签。
4.1.2. 接口.....	错误!未定义书签。
4.2. 明确意图说明.....	27
4.2.1. 功能.....	27
4.2.2. 回调 cmd.....	27
4.2.3. 交互流程.....	27
4.2.4. 数据格式.....	28
4.2.5. 明确意图构造方法.....	28
4.3. 手机与设备通信模块.....	错误!未定义书签。
4.3.1. 功能.....	错误!未定义书签。
4.3.2. 交互流程.....	错误!未定义书签。
4.3.3. 数据格式.....	错误!未定义书签。
5. FAQ.....	31

1. 简介

本文档介绍功能的使用流程，不介绍接口的详细信息。接口详细信息请见 API 文档。若开发者选择下载的 SDK 不包含所有功能，第 3 章只需看相应小节即可。

1.1. 名词解释

TVS DEVICE SDK：腾讯叮当提供的语音唤醒、语音识别、语义识别、语音合成技术解决方案的软件开发工具包。

语音唤醒：智能硬件/应用在休眠状态下通过个性化语音唤醒词被唤醒。

语音识别：将语音转换为对应的文本。

语义识别：将文本转换成结构化的实体、领域、意图、服务数据。

语音合成：将文本转换为语音音频流。

1.2. 环境依赖

Linux x86_64 PC 版本 SDK 的编译环境如下：

gcc 版本	5.4
glibc 版本	2.23
glibcxx 版本	3.4.21
操作系统	ubuntu 16.04

使用低版本 glibc 或者 glibcxx 无法正常编译/运行。

Linux x86_64 SDK 依赖这些函数库：libz、libm、libgfortran（唤醒需要）、pthread，请确保机器上安装了 gfortran , zlib。

Ubuntu 可以输入命令安装：

```
apt-get install zlib1g-dev
```

```
apt-get install gfortran（如果需要唤醒功能）
```

1.3.非通用架构支持

如果开发者需要特定嵌入式 Linux 系统（Linux 系统音箱、电视盒子、遥控器等）版本的 SDK，开发者需要联系我们，并提供相应的交叉编译链，由我们编译相应版本的 SDK。

对交叉编译链的要求如下：

1. gcc 版本 4.6 以上 。
3. 最好包含 libz 库。

2. 入门

2.1.使用步骤

2.1.1. 复制依赖库和头文件

把 libs 文件夹内 so 动态库复制到编译器可以找到的目录，否则编译时会提示找不到符号的错误。

把 include 下头文件复制到项目中。

2.1.2. 初始化

首先，需要初始化 SDK。

初始化函数在 include/aisdk_common_api.h 内声明。

```
/**
 * @brief 初始化函数
 * @param in folderPath 配置路径
 * @param in appKey 应用的appkey, 需要从平台申请
 * @param in accessToken 应用的access token
 * @return <em>0</em>:ok others:fail
 * @note
 */
AISDK_API_EXPORTS int aisdkInit(const char* folderPath, const char* appKey, const char* accessToken);
```

folderPath 为 SDK 的工作目录，该目录应当是可读写的，SDK 在运行过程中产生的日志和数据会生成到这个目录中，如果不可读写，将会影响语义和语音识别效果。SDK 需要的配置文件 config.conf（在 SDK 包的 res 目录下有该配置文件的通用模板），需要在初始化之前预先放置在该目录中，如果没有放置，则 SDK

将使用默认配置。其它需要的资源（例如唤醒模型）建议也放在该目录中。

appKey 和 accessToken 是腾讯叮当智能语音平台分配给厂商的，在初始化时需要传给 SDK。

最终目录结构如下：

folderPath:

config.conf（非必须）

```
kangrong@kangrong-VirtualBox: /mnt/AndroidSh/IVAClient/outsd
总用量 4
drwxrwxrwx 1 root root    0 9月 14 11:04 ./
drwxrwxrwx 1 root root 4096 9月 13 20:25 ../
-rwxrwxrwx 1 root root    0 9月 12 19:10 config.conf*
```

其次，应当设置回调函数指针，关于回调将在下一节说明。

2.1.3. 回调

SDK 与上层主要通过异步交互。上层注册一个回调函数，接收 SDK 的异步的回调。注册回调的接口如下：

```
/**
 * @brief 设置回调函数
 * @param in callbackPtr 回调函数指针
 * @return
 * @note
 *
 */
AISDK_API_EXPORTS void aisdkSetCallback(AISDK_CALLBACK callbackPtr);
```

回调函数的原型为：

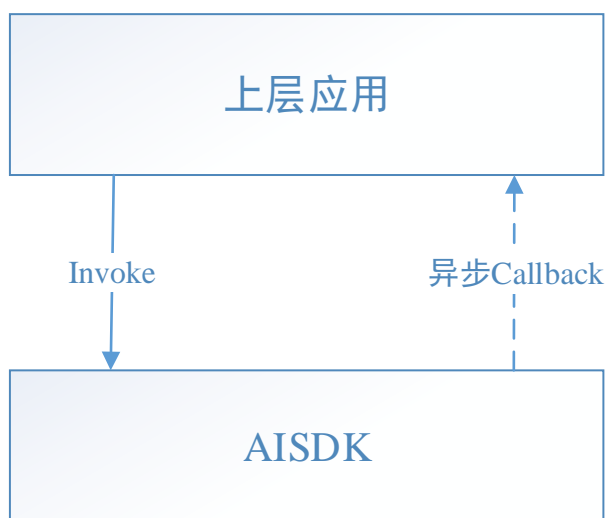
```
/**
 * @define 设置回调函数原型
 * @param cmd 当前的指令，指示参数 data 的内容解析方式和含义，定义由各模块指定
(AISDK_CMD_*开头的常量)
 * @param data 数据存储区域
```

```

* @param len data 的长度
* @param userData 传回用户自定义数据，此数据是在用户发起请求是传入的
* @param userDataLen 用户自定义数据长度
* @param extraData 附加数据，如果没有返回 NULL
* @param extraDataLen 附加数据长度
* 回调命令
* @see 参见各接口头文件的 AISDK_CMD_*常量定义
*
*/
void callback(int cmd, char* data, int dataLen, void* userData, int userDataLen, void *extraData,
int extraDataLen);

```

回调带有 cmd、data 等参数。cmd 用来区分不同类型的回调，data 用于回传结果。例如 cmd= AISDK_CMD_SEMANTIC_RESULT 表示语义请求已经有结果，data 为返回的语义 JSON 数据。注意，在回调方法中的实现 **不能执行耗时的操作**，否则会阻塞 SDK 的后续流程。



在本文中，虚线表示调用是异步的。

2.2.输入音频要求

SDK 的唤醒识别和在线语音识别对输入音频的要求是一致的，要求如下表所示：

项目	要求
音频格式	PCM 格式
采样精度	16 位
采样率	16000Hz
声道	1 声道（单声道）
字节序	小端
单包建议大小	1600 字节

2.3. 编码

默认情况下，传入 SDK 的文本参数和 SDK 返回的文本结果，为 UTF-8 编码。

3. 各功能模块入门

SDK 提供了语音唤醒、在线语音识别、在线语义识别、在线语音合成的功能。

一个集成语音唤醒功能的音箱应用使用 SDK 的流程可以如下所述：

- 调用初始化接口初始化 SDK，并设置回调函数。
- 调用语音唤醒的启动接口开始一次语音唤醒识别流程，调用语音唤醒的输入接口实时输入录音。直到收到识别到唤醒词的回调，此时应停止向 SDK 输入录音，调用语音唤醒的取消接口结束本次唤醒识别流程。
- 如果收到识别到唤醒词的回调，表明音箱已经被唤醒。音箱可以播放一个提示音提醒用户音箱现在可以接收用户说话。
- 调用语音识别的开始接口 `aisdkStartOnlineVoice2Text()` 开启一次语音识别，收到语音识别开始的回调后，调用 `aisdkInputOnlineVoice2TextAudioData()` 向在线语音识别模块实时输入录音，等收到语音识别结果的回调后，录音全部转成了文字，此时就完成了一次语音识别流程。此时应继续监听用户的唤醒，或者根据需要进行连续会话。
- 通过语义识别的接口请求语义，将上一步识别到的文字转换为用户的意图。待收到语义结果的回调后，解析得到的语义服务 JSON 结果。
- 音箱解析语义结果，假如语义的结果是 music 领域（音乐）的，意图为 play（播放），如果有播报语，或者自己组装了一个播报语（如“好的，现在为你

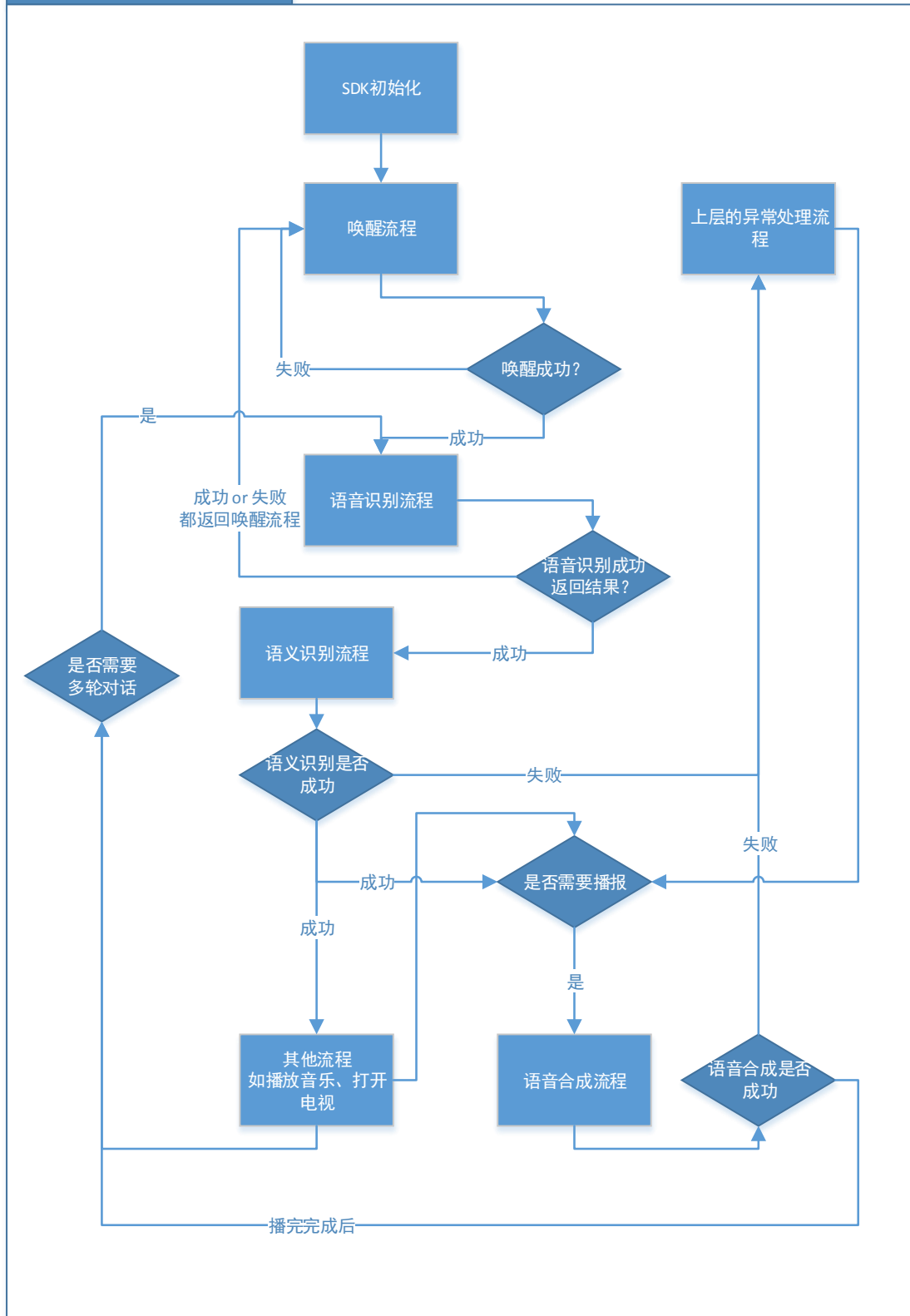
播放{歌手名字}的歌”），那么通过在线语音合成的接口，请求合成的音频进行播放。然后开始播放音乐。

附：如果语义结果返回的 `session=false`，表示需要多轮交互。

多轮交互：与用户进行交互的过程中，如果语义引擎觉得信息不全，需要用户进一步提供信息，需要重新开启一轮交互。例如，用户说“定一个闹钟”，语义引擎没有识别到时间信息，所以返回的语义数据中 `session=false`，下一次的交互，都将会当成定闹钟的交互，此时建议先提醒用户“你要定什么时间的闹钟”，然后开启在线识别录音。

大致流程如下图所示。

一个可能的SDK使用方式



下面介绍每个功能模块的使用。

3.1.语音唤醒

3.1.1.功能

语音唤醒的接口，可以识别音频流中的唤醒语音。上层应用可以根据唤醒结果进行下一步的操作。

SDK 不能进行录音，所以需要上层应用把录音数据实时传入 SDK。

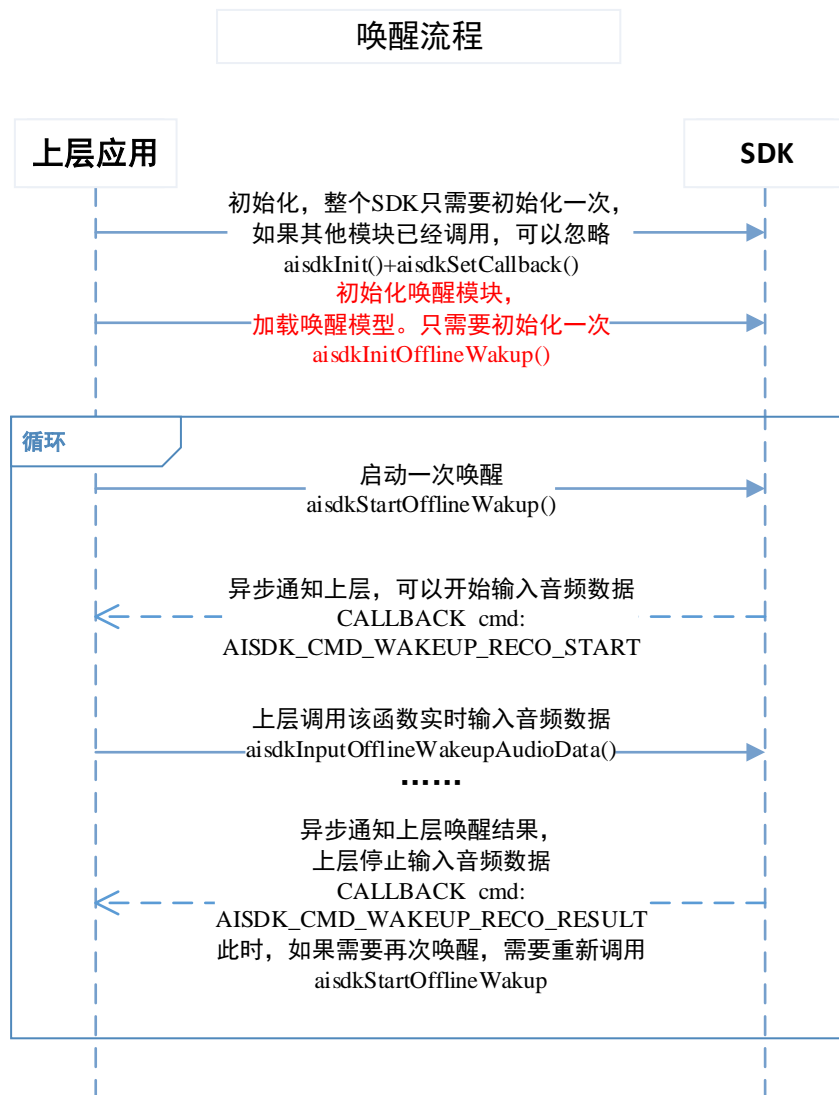
默认唤醒词：叮当叮当。

3.1.2.回调 cmd

CMD	说明
AI SDK_CMD_WAKEUP_RECO_START	开始语音唤醒流程，上层可以开始输入语音数据。
AI SDK_CMD_WAKEUP_RECO_RESULT	语音唤醒流程结束,返回结果。 语音唤醒流程结束。
AI SDK_CMD_WAKEUP_RECO_ERROR	语音唤醒出错。语音唤醒流程结束。
AI SDK_CMD_WAKEUP_RECO_CANCELED	唤醒识别流程被调用方取消。

3.1.3.交互流程

3.1.3.1. 正常流程



首先调用 SDK 初始化的接口 `aisdkInit()`和 `aisdkSetCallback()`, SDK 在应用使用中只需要初始化一次, 如果其他模块或者本模块已初始化过 SDK, 就不需要再次初始化。然后初始化语音唤醒模块。

上层应用（如带有语音唤醒功能的音箱）在启动完毕后可以启动一次唤醒流程（`aisdkStartOfflineWakeup()`）。收到 `cmd=AISDK_CMD_WAKEUP_RECO_START` 后, 上层可以开始录音并实时向语音唤醒模块输入语音数据, 语音唤醒模块会检测语音数据流中的唤醒语音。当检测到唤醒语音, SDK 将会发出 `cmd=AISDK_CMD_WAKEUP_RECO_RESULT` 的回调, 并带有唤醒结果。这一次的唤醒流程结束。

此时, 上层应该停止向唤醒模块输入录音, 建议先进行语音识别的流程, 语音识别流程完毕后重新开启一次唤醒流程（`aisdkStartOfflineWakeup()`）继续监听

用户的唤醒输入。

当上层应用调用 `aisdkCancelOfflineWakeup` 时就取消本次唤醒识别流程，此时 SDK 会发出 `cmd=AISDK_CMD_WAKEUP_RECO_CANCELED` 回调。

3.1.3.2. 异常处理

开始唤醒流程后，发生引擎错误会产生 `cmd=AISDK_CMD_WAKEUP_RECO_ERROR` 的回调，上层应当停止输入音频。可以尝试重新开启一次唤醒（`aisdkStartOfflineWakeup()`）。

3.1.4. 数据格式

3.1.4.1. AISDK_CMD_WAKEUP_RECO_RESULT 回调

数据格式为 JSON，如下：

```
{
  "rc":0,
  "result":{
    "code":0, //0 表示识别到唤醒词，1 表示取消
    "data":"ding1 dang1" //识别到的唤醒词
  }
}
```

正常情况下，code 有以下几种

值	常量	说明
0	AISDK_RESULT_CODE_WAKEUP_OK	识别到唤醒词
1	AISDK_RESULT_CODE_WAKEUP_CANCELED	取消唤醒，当调用 <code>aisdkCancelOfflineWakeup</code> 时会有此情况。取消后，语音唤醒流程结束。

3.1.4.2. AISDK_CMD_WAKEUP_RECO_ERROR 回调

数据格式为 JSON，如下：

```
{
  "rc":1, //返回码
  "error":{
    "code":错误码,
    "message":"错误信息"
  }
}
```

code 有以下几种：

值	常量	说明
7000	AI SDK_ERROR_WAKEUP_RECO_FAILED	唤醒引擎错误

3.2. 在线语音识别

3.2.1. 功能

语音识别的接口，可以流式识别语音中的文本。

SDK 不能进行录音，所以需要上层把录音的数据实时传过来。语音识别有两种模式：自动模式（默认情况）和手动模式。

自动模式即在线语音识别模块会自动检测语音输入的结束（静音时间 500ms）并返回语音识别结果。这是默认的模式。

手动模式即在线语音识别模块不会自动结束，上层主动调用 stop 才会结束并返回结果。手动模式比较适合通过按键启动和结束语音识别的场景。

3.2.2. 回调 cmd

CMD	说明
AI SDK_CMD_ONLINE_RECO_START	本次语音识别已经启动，上层可以开始输入录音数据。

AI SDK_CMD_ONLINE_RECO_SPEECH_START	语音检测到开始端点。
AI SDK_CMD_ONLINE_RECO_SPEECH_END	语音检测到结束端点，上层停止输入录音数据
AI SDK_CMD_ONLINE_RECO_RESULT	返回在线识别结果，本次语音识别结束。
AI SDK_CMD_ONLINE_RECO_INTERMEDIATE_RESULT	上报在线识别的中间结果。一句话没有说完的时候，返回的部分识别文字。识别过程中多次回调。
AI SDK_CMD_ONLINE_RECO_DATA_VOLUME	上报输入音频数据的音量值（or 能量值）。可以用来显示实时录音的语音音量水平。取值范围是 0-25。每次输入语音数据包都会回调。
AI SDK_CMD_ONLINE_RECO_ERROR	错误发生。本次语音识别结束，上层停止输入录音数据。
AI SDK_CMD_ONLINE_RECO_TIMEOUT	超时：在 10s 内没有检测到语音开始端点。本次语音识别结束，上层停止输入录音数据。
AI SDK_CMD_ONLINE_RECO_CANCELED	取消：当上层调用 aisdkCancelOnlineVoice2Text() 时，SDK 会发出这个回调。本次语音识别结束，上层停止输入录音数据。

3.2.3. 交互流程

3.2.3.1. 自动结束

默认情况下，SDK 自动识别语音的结束点并返回识别结果，交互方式如下：



SDK 初始化在整个应用使用中，只需要调用一次，若其他功能模块或者本模块已经调用过，不需要再次调用。

3.2.3.2. 手动结束

SDK 也支持手动模式，不自动结束语音识别，以实现**按键形式**的控制。

使用方法是：在调用 `aisdkStartOnlineVoice2Text` 时，传入 `AISDK_FLAG_ONLINE_RECO_MANUAL_MODE`（如用户按下按键开始）。上层要结束语音输入的时候调用 `aisdkStopOnlineVoice2Text`（如用户松开按键结束录音）。

那么交互流程如下：



3.2.3.3. 取消识别

语音识别提供了取消接口：aisdkCancelOnlineVoice2Text()

在调用 aisdkStartOnlineVoice2Text 后，可以调用 aisdkCancelOnlineVoice2Text 取消整个识别流程。可能的使用场景是：1. 某个业务要打断当前的语音识别流

程。

3.2.4. 数据格式

3.2.4.1. AISDK_CMD_ONLINE_RECO_INTERMEDIATE_RESULT 回调

数据格式为 JSON，如下：

```
{
  "rc":0,
  "result":{
    "code":0,
    "data":"识别到部分语音结果"
  }
}
```

3.2.4.2. AISDK_CMD_ONLINE_RECO_RESULT 回调

数据格式为 JSON，如下：

```
{
  "rc":0,
  "result":{
    "code":0,
    "data":"识别到的结果"
  }
}
```

例如

```
{"rc":0,"result":{"code":0,"data":"给我放一首周杰伦的歌曲"}}
```

正常情况下，code 如下：

值	常量	说明
0	AISDK_RESULT_CODE_ONLINE_OK	正常

3.2.4.3. AISDK_CMD_ONLINE_RECO_ERROR 回调

数据格式为 JSON，如下：

```
{
  "rc":1,
  "error":{
    "code":错误码,
    "message":"错误信息"
  }
}
```

错误码有以下几种：

值	常量	说明
5	AISDK_ERROR_COMMON_NETWORK_FAIL	网络请求发送失败。
6	AISDK_ERROR_COMMON_NETWORK_RESPONSE_FAIL	网络请求回包失败。
7	AISDK_ERROR_COMMON_NETWORK_TIMEOUT	网络请求超时。
10	AISDK_ERROR_COMMON_SERVICE_RESPONSE	后台服务异常

3.3.在线语义

3.3.1. 功能

语义模块可以将文本识别为的 domain 和 intent、语义实体，并返回对应的服务数据。例如把“我想听周杰伦的歌曲”识别为 domain 为 music、intent 为 play，带有的语义实体是歌手名字为“周杰伦”，服务数据为周杰伦的歌曲列表。

SDK 只会返回 JSON 数据，不做具体逻辑，如播放音乐需要由上层用用实现。

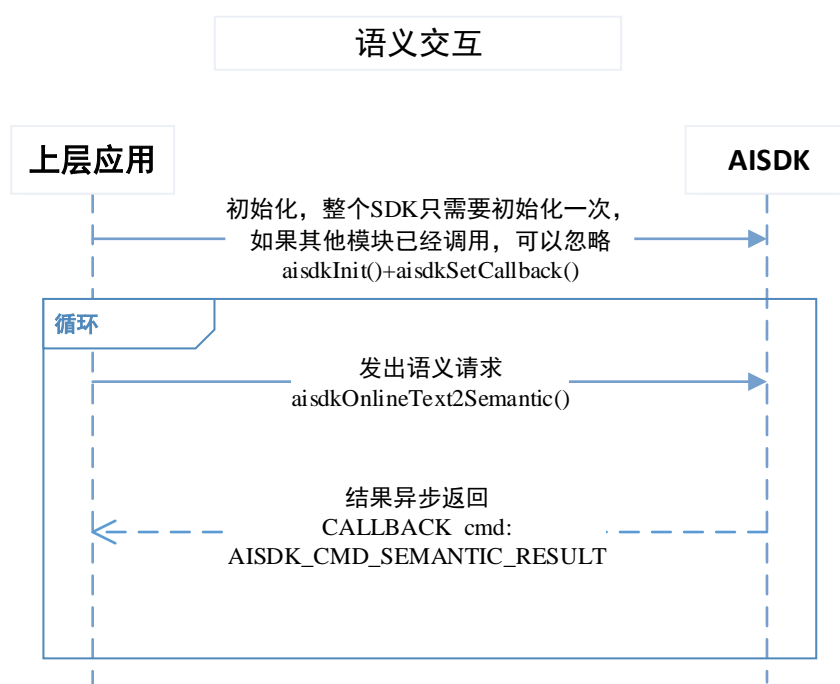
3.3.2. 回调 cmd

CMD	说明
AI SDK_CMD_SEMANTIC_RESULT	返回在线语义识别结果。语义识别流程结束。
AI SDK_CMD_MEDIA_SEMANTIC_RESULT	返回请求音乐、FM 信息的结果。语义识别流程结束。
AI SDK_CMD_SEMANTIC_ERROR	在线语义识别错误。语义识别流程结束。
AI SDK_CMD_MEDIA_SEMANTIC_ERROR	请求音乐、FM 错误。语义识别流程结束。

3.3.3. 交互流程

3.3.3.1. 请求语义

请求语义的交互流程如下：



当返回的语义结果的 `session==false`，需要进行多轮交互。

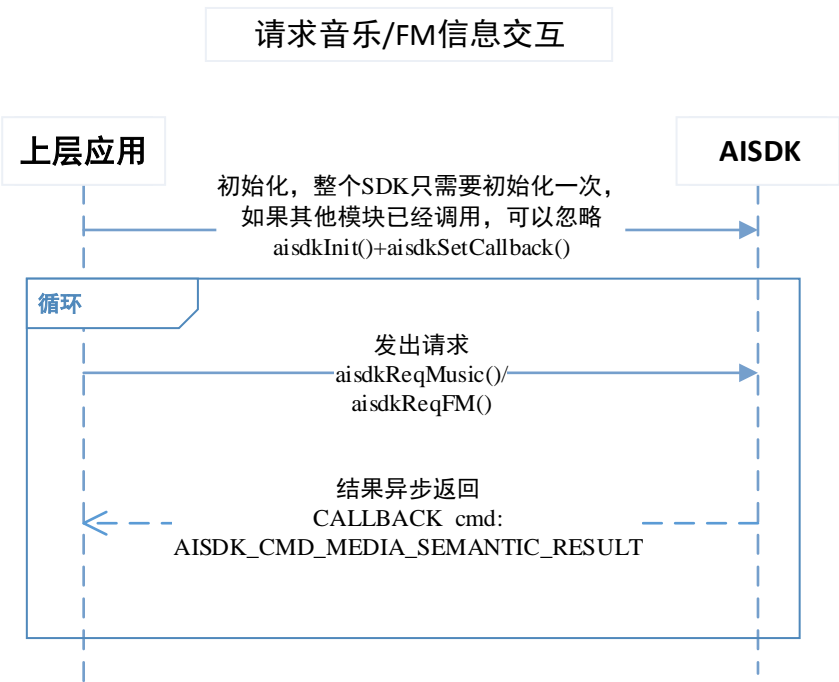
3.3.3.2. 功能性接口

除了常规的请求语义的接口，语义识别模块还提供了一些接口来实现特殊功

能。

在 FM/音乐领域，语义返回的音乐播放地址，可能会有变化。所以，在播放音乐或者 FM 前，建议使用音乐/FM 的 ID 重新请求一下音乐/FM 的播放地址。

请求单个音乐或者 FM 信息的流程如下：



3.3.4. 数据格式

3.3.4.1. AISDK_CMD_SEMANTIC_RESULT 回调

/AISDK_CMD_MEDIA_SEMANTIC_RESULT 回调

返回的是语义结果，数据格式为 JSON，如下：

```
{
  "rc":0, //返回码，0 表示正常
  .....//其他字段
}
```

具体格式可以参考《TVS Device SDK 语义结构说明书》

3.3.4.2. AISDK_CMD_SEMANTIC_ERROR

回调

/AISDK_CMD_MEDIA_SEMANTIC_ERROR 回调

数据格式为 JSON，如下：

```
{
  "rc":1,
  "error":{
    "code":错误码,
    "message":"错误信息"
  }
}
```

错误码有以下几种：

值	常量	说明
5	AISDK_ERROR_COMMON_NETWORK_FAIL	网络请求发送失败。请检查网络状态。
10	AISDK_ERROR_COMMOM_SERVICE_RESP	后台服务异常

3.4.在线语音合成

3.4.1.功能

语音合成的接口，可以将文本转换语音数据。SDK 没有播放音频的功能，上层自行处理语音数据。

3.4.2.回调 cmd

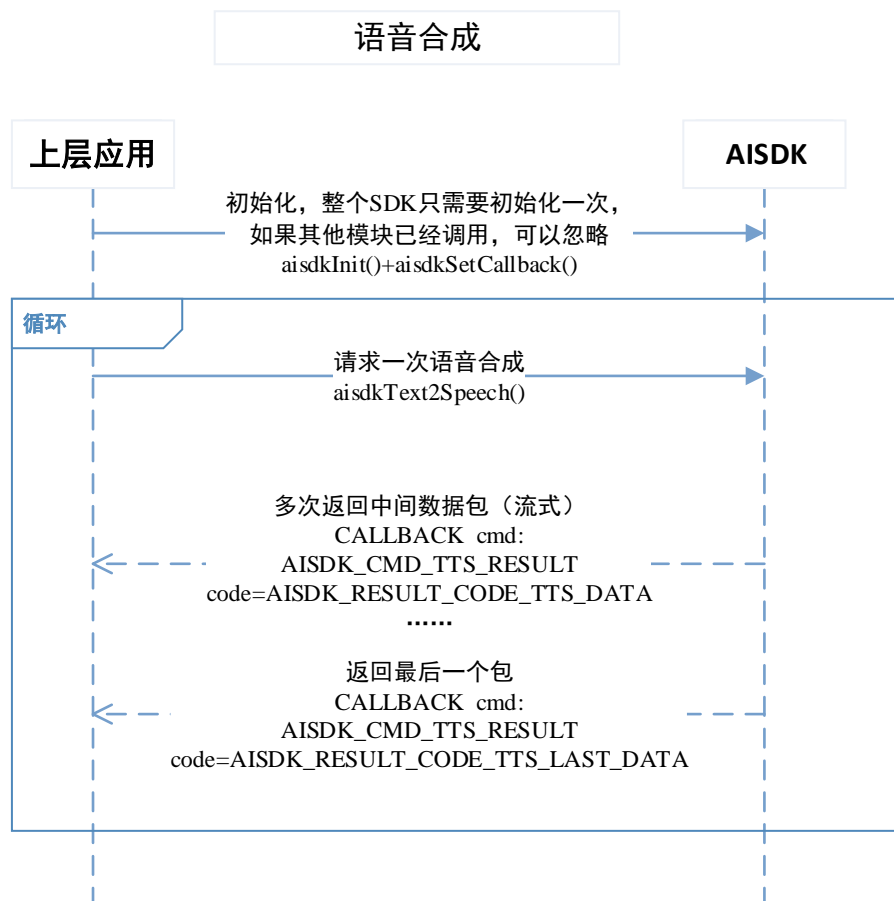
CMD	说明
-----	----

AI SDK_CMD_TTS_RESULT	返回语音合成处理结果。语音合成流程结束。
AI SDK_CMD_TTS_ERROR	语音合成错误。语音合成流程结束。

3.4.3. 交互流程

长文本的语音合成请求的结果分多次返回（采用流式合成以减少延时），最后一个返回的语音数据包的code为AI SDK_RESULT_CODE_TTS_LAST_DATA。

短文本的语音合成请求的结果一次性返回，语音数据包的code为AI SDK_RESULT_CODE_TTS_LAST_DATA。



3.4.4. 数据格式

3.4.4.1. AISDK_CMD_TTS_RESULT 回调

数据格式为 JSON，如下：

```
{
  "rc":0,
  "result":{
    "code":code, //标识是否是最后一个包
    "data":"bin2str()方法编码的音频数据" //音频
  }
}
```

正常情况下，code 有以下几种

值	常量	说明
0	AISDK_RESULT_CODE_TTS_DATA	表示 data 中的数据不是最后一个音频数据包
1	AISDK_RESULT_CODE_TTS_LAST_DATA	表示 data 中的数据是最后一个音频数据包。语音合成完成。

注意：音频流数据从 extraData 中返回。

3.4.4.2. AISDK_CMD_TTS_ERROR 回调

数据格式为 JSON，如下：

```
{
  "rc":1,
  "error":{
    "code":错误码,
    "message":"错误信息"
  }
}
```

错误码如下：

值	常量	说明
5	AISDK_ERROR_COMMON_NETWORK_FAIL	网络请求发送失败
6	AISDK_ERROR_COMMON_NETWORK_RESPONSE_FAIL	网络请求回包失败
7	AISDK_ERROR_COMMON_NETWORK_TIMEOUT	网络请求超时
10	AISDK_ERROR_COMMON_SERVICE_RESP	后台服务异常

4. 附录

4.1. 明确意图说明

4.1.1. 功能

当开发者已经知道语义，想去直接获得服务数据时，可以调用 `aisdkComplexSemantic2Semantic` 函数。

4.1.2. 回调 cmd

CMD	说明
AISDK_CMD_COMPLEX_SEMANTIC_RESULT	返回数据，解析方法与在线语义相同。
AISDK_CMD_COMPLEX_SEMANTIC_ERROR	明确意图出错

4.1.3. 交互流程

交互流程与在线语义相同

4.1.4. 数据格式

返回数据格式与在线语义相同

4.1.5. 明确意图构造方法

```
aisdkComplexSemantic2Semantic(const char* semanticJson, int len, void *userData,
int userDataLen);
```

其中 semanticJson 为 json 结构字符串。

总体结构如下：

字段	类型	说明
service	String	领域
operation	String	意图
query	String	查询字符串，可以填空串，一般后台忽略这个字段
session_complete	Bool	回话是否结束，默认可填 false
slots	JSONArray	实体列表。可不填。与《SDK 语义格式.pdf》所列的 semantic/slots 一致。 注意：每个意图所需要的 slots 都不一样。

播放歌曲示例：

```
{
    "query": "",
    "service": "music",
```

```
"operation": "play"
"session_complete": false
}
```

没有指定实体，后台将会随机返回歌曲。

播放歌曲（指定周杰伦）：

```
{"query": "", "service": "music", "operation": "play", "slots": [{"name": "singer", "type":
"sys.music.singer", "slot_struct": 1, "values": [{"text": "周 杰 伦
", }]}, {"session_complete": false}
```

slots 内填了周杰伦这个实体。

查询上海天气：

```
{"query": "", "service": "weather", "operation": "general_search", "slots": [{"name":
"location", "type": "sys.geo.county", "slot_struct": 2, "values": [{"original_text": "上海
", "country": "", "province": "", "city": "上 海 ", "district": "", "town": "", "street":
"", "longitude": 0, "latitude": 0, "vLBSKeyData": {"type": "Buffer", "data": []}, "title":
"", "village": "", "residual": "", "source": 0}], "prompt": {"show_text": "", "speak_text":
"", "prompt_type": 0, "slot_name": "", "slot_type": ""}}]}
```

slots 内填了上海这个位置实体。

4.2. 状态上报接口说明

4.2.1. 功能

终端有些状态的变化需要通知后台，比如闹钟响铃，后台收到后可以更新云端的闹钟列表，甚至同步到其他端。这种情况可以调用 `aisdk_report_online.h` 头文件提供的 `aisdkReportClientState` 函数。

4.2.2. 函数原型

```
void aisdkReportClientState(const char* sJsonData, void* userData, int
userDataLen);
```

参数	说明
sJsonData	要上报的内容，JSON 格式，各类型的上报对应的数据会有不同，下面介绍
userData	用户数据，在回调时传回
userDataLen	用户数据长度，在回调时传回

4.2.3. 上报类型和数据说明

```
(1) 闹钟状态上报
{
    "businessType":6,    // 标识闹钟状态上报
    "calendarList":[    // 闹钟状态列表，每次最多 10 个，超过请分批上报
        {
            "report_type":1, // 参见 E_REMINDER_REPORT_TYPE
            "event_type":1, // 参见 E_REMINDER_EVENT_TYPE
            "calendar_type":0, // 参见 E_REPEAT_TYPE
            "calendar_id":1522379260112, // 闹钟唯一 ID 服务端下发
            "calendar_time":1522379320, // 事件发生时间 与 calendar_id 对应，均有服务端下发
            "calendar_note":"喝水" // 提醒事件
        }
    ]
}

取值说明：
// 提醒类型 闹钟/提醒
enum E_REMINDER_EVENT_TYPE{
```

```

E_REMINDER_ALARM = 0, // 闹钟
E_REMINDER_REMINDER = 1 // 提醒
};
// 上报事件类型
enum E_REMINDER_REPORT_TYPE{
    E_REMINDER_REPORT_NO_SIGN = 0, // 无意义 服务接收到该类型时 会丢弃请求
    E_REMINDER_REPORT_SET_SECCCESS = 1, // 闹钟设置成功
    E_REMINDER_REPORT_SET_FAILURE = 2, // 闹钟设置失败
    E_REMINDER_REPORT_DELETE_SECCCESS = 3, // 闹钟删除成功
    E_REMINDER_REPORT_DELETE_FAILURE = 4, // 闹钟删除失败
    E_REMINDER_REPORT_START = 5, // 闹钟开始响铃
    E_REMINDER_REPORT_STOP = 6, // 闹钟停止响铃
};
// 闹钟重复类型
enum E_REPEAT_TYPE{
    E_REPEAT_ONCE = 0, // 一次性闹钟
    E_REPEAT_DAY = 1, // 按天重复
    E_REPEAT_WEEK = 2, // 按周重复
    E_REPEAT_MONTH = 3, // 按月重复
    E_REPEAT_WORKDAY = 4, // 按工作日重复
    E_REPEAT_WEEKEND = 5, // 按节假日重复
    E_REPEAT_SELF = 6, // 无效 服务接收到该类型时 会丢弃请求
    E_REPEAT_HOUR = 7, // 无效 服务接收到该类型时 会丢弃请求
    E_REPEAT_EXCEPT = 8 // 无效 服务接收到该类型时 会丢弃请求
};

```

5. FAQ

1. 语音识别返回结果慢是什么原因？

语音识别对网络速度依赖比较大，可以检查下网络状态。