ALIBABA CLOUD

阿里云

智能语音交互

设备端语音交互SDK

文档版本：20201014

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.接口说明

随着智能硬件的迅速发展，以及对语音交互的迫切需求，我们设计了具有全链路语音能力的NUI（Natural User Interaction）SDK。SDK聚合了端&云语音核心算法能力，包括远场信号处理、语音唤醒、语音识别、语义理解及语音合成等语音交互全链路模块。通过十分简单的接口，快速完成对产品的语音赋能。

> 注意
>
> 目前提供的prebuilt SDK仅适用于炬芯ATS3605D芯片定制Linux系统软件环境，使用双路数据及一路参考声道（总共三路）作为输入，算法效果仅在特定设备上有效，其他芯片支持敬请期待。

功能简介

需要说明的是，NUI SDK不同于智能语音交互通用SDK（一句话识别、实时识别、语音合成、长文本语音合成），其主要用在如智能音箱、儿童教育故事机、语音IoT家电等需要远近场语音交互的智能硬件设备端。和智能语音交互通用SDK比，NUI SDK提供了一套完整的端到端远场语音解决方案。

设备端语音交互SDK特性

- 远场信号处理

  在远场语音交互场景中，智能设备通常要面对设备回声、人声干扰、环境噪声、房间混响等诸多不利声学因素的影响。NUI SDK提供了一套音频前端系统来对原始音频进行增强，提高目标信号的信噪比和语音可懂度，从而提升人机/人人交互的用户体验。

- 语音唤醒

  支持定制词语的唤醒模型。当SDK检测到有人说出该词后，便抛出唤醒信号。唤醒服务支持多个唤醒词和命令词，从唤醒词录制到模型训练完成大约需要2~3周时间。

- 人声检测

  为了节约设备计算资源、减少端侧功耗，NUI SDK自建人声检测功能，只有通过人声检测的音频才会发送给云端进行语音识别。

- 在线语音识别

  对时长较短（一分钟以内）的语音进行识别。适用于较短的语音交互场景，如语音搜索、语音指令、语音短消息和语音对话等。

- 在线语音合成

  语音合成服务，通过先进的深度学习技术，将文本转换成自然流畅的语音。多种音色可供您选择，并提供调节语速、语调和音量等功能。

NUI SDK与其他原子SDK区别

对比项	语音识别SDK（含一句话识别、实时语音识别和录音文件识别）	语音合成SDK（含语音合成、长文本语音合成）	NUI SDK
打断唤醒能力	×	×	√
远场降噪	×	×	√

对比项	语音识别SDK（含一句话识别、实时语音识别和录音文件识别）	语音合成SDK（含语音合成、长文本语音合成）	NUI SDK
命令词&快捷词	×	×	√
人声检测	×	×	√
语音识别	√	√	√
语音合成	√	√	√
计费方式	• 实时语音识别和录音文件识别按语音时长计费。 • 一句话识别按调用次数计费。	按调用次数或字数计费。	按激活台数计费。

服务地址

访问类型	说明	URL
外网访问	所有服务器均可使用外网访问服务URL（SDK中默认设置了外网访问URL）。	wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1

交互状态机

交互系统分为4个状态，分别为：

- UNINIT：未初始化状态（默认状态）。
- STOP：暂停状态。SDK初始化后处于STOP状态。
- IDLE：待机状态。该状态接收语音输入可以进行语音唤醒，当发生唤醒事件后SDK仍然处于IDLE状态，可以通过interactive接口直接切换至INTERACTIVE状态。
- INTERACTIVE：识别状态。该状态可以接收音频输入，当识别结束或者识别出错则会自动切换至IDLE状态。

每个状态的交互特性如下表所示。

特性	UNINIT	STOP	IDLE	INTERACTIVE
是否接收外部音频输入	否	否	是	是
是否可唤醒	否	否	是	是
是否可进行语音识别	否	否	否	是

2.NUI SDK

本文为您介绍NUI SDK的关键接口、调用步骤及代码示例。

前提条件

- 在使用SDK之前，请先阅读接口说明，详情请参见接口说明。
- 已在智能语音管控台创建设备端解决方案项目，详情请参见创建项目。
- 已获取AccessKey ID和AccessKey Secret，详情请参见开通服务。

下载SDK

在阿里云智能语音交互管控台项目功能配置页面，下载SDK。

关键接口

- asr_engine_init

  ```
  void *asr_engine_init(void);
  ```

  功能：初始化引擎。

  返回值：返回handle供后续api使用。

- asr_engine_start

  ```
  int asr_engine_start(void *handle);
  ```

  功能：启动引擎，引擎启动后直到用户调用asr_engine_stop将一直工作。

  参数handle：asr_engine_init返回的handle。

  返回值：ERROR_CODE，0为成功，1为失败，2为参数错误。

- asr_engine_feed_data

  ```
  int asr_engine_feed_data(void handle, char data, int data_size);
  ```

  功能：向音频引擎提供录音数据。

  参数：

  - handle：asr_engine_init返回的handle。
  - data：音频数据。
  - data_size：音频数据长度，单位为字节，可以是任意长度。

  返回值：ERROR_CODE，0为成功，1为失败，2为参数错误。

- asr_engine_stop

  ```
  int asr_engine_stop(void *handle)
  ```

  功能：暂停引擎。

参数handle：asr_engine_init返回的handle。

返回值：ERROR_CODE，0为成功，1为失败，2为参数错误。

- asr_engine_finalize

```
int asr_engine_finalize(void *handle)
```

功能：释放引擎。

参数handle：asr_engine_init返回的handle。

返回值：ERROR_CODE，0为成功，1为失败，2为参数错误。

- asr_engine_get_property

```
int asr_engine_get_property(ASR_PROPERTY_TYPE property_type, char* value, int length);
```

功能：获取引擎属性。

参数：

  - value：属性。
  - length：value的长度。
  - property_type：属性类型，下表列出可选属性，参数定义见SDK头文件。

属性	说明
ASRPropertyAEC	是否支持AEC。
ASRPropertyInitOnce	是否只支持初始化一次。
ASRPropertyNeedAuth	是否需要鉴权。
ASRPropertySupportFFVP	是否支持返回前端处理后的音频。
ASRPropertySupportVAD	是否支持返回VAD的音频。
ASRPropertyNeedAuthEveryTime	是否每次都需要联网鉴权。
ASRPropertyMixPcmFormat	参考声道和录音声道的顺序，0为参考在前，1为录音在前。
ASRPropertyDynamicWuws	返回目前设置的所有命令词。
ASRPropertyErrorCode	初始化错误码。

返回值：ERROR_CODE，0为成功，1为失败，2为参数错误。

- asr_engine_set_params

```
ASR_ERROR_CODE asr_engine_set_params(ASR_PARAM_TYPE param_type, unsigned long value_size, void *value);
```

功能：用以给引擎设置参数。

参数：

- value_size：设置数据长度。
- value：设置数据。
- param_type：参数类型，下表列出可选参数类型，定义见SDK头文件。

参数类型	说明
ASR_PARAM_EVENT_CB	设置引擎事件回调，需要在asr_engine_init之前调用。
ASR_PARAM_ENABLE_AEC	设置引擎使能AEC，默认开启，暂不支持关闭。
ASR_PARAM_MIC_NUM	设置麦克风数量，暂不支持动态修改。
ASR_PARAM_REF_NUM	设置参考声道数量，暂不支持动态修改。
ASR_PARAM_ENABLE_FFVP	设置使能返回前端音频，暂不支持。
ASR_PARAM_ENABLE_VAD	设置使能返回VAD音频，默认返回，暂不支持修改。
ASR_PARAM_ENABLE_DEBUG	设置使能debug模式，设置“1”为开启，“0”为关闭，开启debug模式后会在ASR_PARAM_SET_DEBUG_PATH指定路径生成相关debug文件。
ASR_PARAM_SET_DEBUG_PATH	设置debug文件保存路径。
ASR_PARAM_SET_PRODUCT_ID	设置product_id。
ASR_PARAM_SET_SDK_CODE	设置SDK CODE，暂不用设置。

参数类型	说明
ASR_PARAM_SET_NUI_APPKEY	设置appkey，需要在asr_engine_init之前调用。
ASR_PARAM_SET_AK_ID	设置access id，需要在asr_engine_init之前调用。
ASR_PARAM_SET_AK_KEY	设置access key，需要在asr_engine_init之前调用。
ASR_PARAM_DIRECT_IP	暂不支持。
ASR_PARAM_SET_WORKSPACE	设置工作路径，需要在asr_engine_init之前调用。
ASR_PARAM_SET_MODE	设置工作模式，0为KWS模式（唤醒），1为VAD模式（不用唤醒直接进行识别），2为命令词模式。
ASR_PARAM_SET_WUWS	设置快捷词，一次调用仅设置一个，如果value是NULL则清空所有快捷词。
ASR_PARAM_ENABLE_WUW_SAVE	设置是否保存唤醒音频数据至云端，暂不支持。
ASR_PARAM_SET_API_KEY	设置第三方APIKEY，初始化必要字段。
ASR_PARAM_SET_DEVICE_ID	设置DEVICE_ID，初始化必要字段。
ASR_PARAM_SET_GAIN	设置软件增益，暂不支持。
ASR_PARAM_ENABLE_ASR	是否开启在线ASR，”1“为开启，”0“为关闭。
ASR_PARAM_ENABLE_DIALOG	是否开启内置对话，暂不支持。
ASR_PARAM_ENABLE_THIRDPARTY_CONTENT_SUPPORT	是否使能云端第三方内容服务支持，暂不支持。
ASR_PARAM_ENABLE_CMD	是否使能命令模式，暂不支持。

- 返回值：ERROR_CODE，0为成功，1为失败，2为参数错误。

- asr_engine_vad_read

  ```
  int asr_engine_vad_read(void handle, char data, int data_size);
  ```

  功能：读取vad数据。仅收到vad start事件后有效，调用不会阻塞，需要不停调用来获取所有vad数据，直到vad end发生。

  参数：

  - handle：asr_engine_init返回的handle。
  - data：放vad数据的buffer。
  - data_size：放vad数据的buffer长度，单位为字节。

  返回值：ERROR_CODE，0为成功，1为失败，2为参数错误。

- asr_engine_interactive

  ```
  int asr_engine_interactive(void *handle);
  ```

  功能：重置引擎状态并强制切换到INTERACTIVE模式。

  参数handle：asr_engine_init返回的handle。

  返回值：ERROR_CODE，0为成功，1为失败，2为参数错误。

事件

回调事件方法的原型如下：

```
void (*asr_event_callback)(ASR_EVENT_TYPE event_type, int cmd, const char *result)
```

其中ASR_EVENT_TYPE事件类型如下：

事件	说明
ASR_EVENT_AUTH_SUCCESS	鉴权成功（暂不支持，通过asr_engine_init返回值判断）
ASR_EVENT_AUTH_FAIL	鉴权失败（暂不支持，通过asr_engine_init返回值判断）
ASR_EVENT_AWAKE_SUCCESS	唤醒事件
ASR_EVENT_AWAKE_CMD	唤醒事件（和ASR_EVENT_AWAKE_SUCCESS一起发生，可以不用处理）
ASR_EVENT_VAD_START	人声开始事件

事件	说明
ASR_EVENT_VAD_END	人声结束事件
ASR_EVENT_VAD_TIMEOUT	唤醒后持续10s没有人声开始事件或者设置VAD模式下超过10s没有人声开始事件
ASR_EVENT_PARTIAL_ASR_RESULT	ASR中间结果
ASR_EVENT_FINAL_ASR_RESULT	ASR最终结果
ASR_EVENT_DIALOG	对话结果
ASR_EVENT_ERROR	错误导致引擎重置为STOP状态

其中cmd参数可以配合完成命令词的功能，用于表明唤醒的命令词是哪个，需要和ASR_PARAM_SET_WUWS配合使用。

使用ASR_PARAM_SET_WUWS设置一个命令词时，其 `value_size` 参数将作为一个value保存在一个key-value map中，这个value将作为cmd在asr_event_callback中返回给用户，从而指明是哪个唤醒词被唤醒，如果是主唤醒词，那么这个cmd的value是0，请勿设置value_size为0的命令词。

初始化错误码

错误码	说明
ASR_OK	无错误
ASR_ERROR	内部引擎错误
ASR_ERROR_INVALID_PARAM	参数错误
ASR_ERROR_ALREADY_INIT	已初始化
ASR_ERROR_NO_CALLBACK	未设置事件回调
ASR_ERROR_AUTH_STRING_EMPTY	鉴权串为空，可能是鉴权关键字段未填入。
ASR_ERROR_AUTH_FAILED	鉴权失败

初始化

SDK初始化步骤如下：

> 说明
>
> 确保设备能正常获取MAC地址。

1. 设置设备device_id和第三方APIKEY。
2. 设置必要鉴权信息，包括：
    i. 阿里云appkey
    ii. 阿里云Accesskey ID
    iii. 阿里云Accesskey Secret
3. 设置模型及资源路径，路径名可以是相对路径。
4. 设置事件回调函数。
5. 调用asr_engine_init完成SDK初始化。

示例代码

```
#include "asr_api.h"

static pthread_t recording_thread;
//sdk handler
static void *handler;

//处理语音事件，事件详情见API文档。
//尽量不要在回调函数中调用耗时间的函数，请将事件通过非阻塞式队列转移到其他线程中进行处理。
void event_callback(ASR_EVENT_TYPE event_type, int cmd, const char *result) {

}

void *recording_fn(void *param) {
    //数据大小可以根据具体软硬件进行调整
    int data_size = 512;
    char data[512] = {0};
    //录音退出时机需要根据具体业务场景来定，这里以简单的死循环模式模拟演示。
    while (1) {
        //录音部分逻辑根据具体软硬件来决定，这里假设音频全部录到data中，长度为512字节。

        //数据提供给SDK
        asr_engine_feed_data(handler, data, 512);

    //其他逻辑
```

```
    //其他逻辑
    }

    return NULL;
}

int main(int argc, char *argv[]) {
    //初始化逻辑
    //设置第三方API KEY以及DEVICE ID
    //CUSTOM_API_KEY, CUSTOM_DEVICE_ID由您根据自身产品情况来获取。
    //32是第三个参数的长度，这里根据实际情况进行设置。
    asr_engine_set_params(ASR_PARAM_SET_API_KEY, 32, CUSTOM_API_KEY);
    asr_engine_set_params(ASR_PARAM_SET_PRODUCT_KEY, 32, CUSTOM_DEVICE_ID);

    //设置SDK APPKEY，ACCESS ID，ACCESS SECRET。
    //TARGET_NUI_APPKEY，TARGET_ACCESS_ID，TARGET_ACCESS_SECRET通过阿里云获取。
    asr_engine_set_params(ASR_PARAM_SET_NUI_APPKEY, 32, TARGET_NUI_APPKEY);
    asr_engine_set_params(ASR_PARAM_SET_AK_ID, 32, TARGET_ACCESS_ID);
    asr_engine_set_params(ASR_PARAM_SET_AK_KEY, 32, TARGET_ACCESS_SECRET);

    //根据业务需求是否使能在线语音识别
    //TARGET_ENABLE_ASR为string，"0"代表false，其他代表true。
    asr_engine_set_params(ASR_PARAM_ENABLE_ASR, 32, TARGET_ENABLE_ASR);

    //设置assets所在路径
    //TARGET_WORKSPACE为assets的绝对/相对路径
    asr_engine_set_params(ASR_PARAM_SET_WORKSPACE, 32, TARGET_WORKSPACE);

    //设置事件callback
    asr_engine_set_params(ASR_PARAM_EVENT_CB, sizeof(event_callback), (void *)event_callback);

    //初始化
    handler = asr_engine_init();
    if (handler == NULL) {
        //初始化出错，根据错误码来判断错误原因。
        int error = 0;
        asr_engine_get_property(ASRPropertyErrorCode, &error, sizeof(error));
        //根据错误码来判断错误原因
        //...

        return -1;
```

```
    }

    //启动引擎
    asr_engine_start(handler);

    //启动录音线程
    pthread_create(&recording_thread, NULL, recording_fn, NULL);

    //直到退出
    pthread_join(recording_thread, NULL);

    //关闭引擎
    asr_engine_stop(handler);

    //释放引擎
    asr_engine_finalize(handler);

    return 0;
}
```