思必驰信息科技有限公司 AISpeech Co., Ltd.	**文档名称** **DUILite SDK 开发手册**	**密级** **绝密**
		文档版本 **V1.24**

DUILite SDK 开发手册

AISPEECH

苏州思必驰信息科技有限公司

修订记录

修订日期	修订版本	部门	修订者	修改内容
2019.2.14	V0.1	sspe	袁伟杰	第一次!
2019.2.26	V0.2	sspe	袁伟杰	初稿完成
2019.3.13	V0.3	sspe	袁伟杰	增加 asrpp 和错误码
2019.5.20	V0.4	sspe	袁伟杰	更新到 sdk v0.12.0
2019.5.27	V0.5	sspe	袁伟杰	声纹 V3 方案
2019.7.5	V0.6	sspe	袁伟杰	更新上传配置
2019.07.26	V0.7	sspe	袁伟杰	更新到 sdk v0.14.1
2019.08.16	V1.15.0	sspe	袁伟杰	对齐 sdk 1.15.0 版本
2019.9.16	V1.16.0	sspe	袁伟杰	对齐 sdk 1.16.0 版本
2019.10.16	V1.17.0	sspe	袁伟杰	对齐 sdk 1.17.0 版本
2019.11.16	V1.18.0	sspe	袁伟杰	对齐 sdk 1.18.0 版本
2019.12.10	V1.19.0	sspe	袁伟杰	对齐 sdk 1.19.0 版本
2020.01.21	V1.20.0	sspe	袁伟杰	对齐 sdk 1.20.0 版本
2020.03.02	V1.21	sspe	袁伟杰	对齐 sdk 1.21.x 版本
2020.04.03	V1.22	sspe	袁伟杰	对齐 sdk 1.22.x 版本
2020.05.11	V1.23	sspe	袁伟杰	对齐 sdk 1.23.x 版本
2020.06.08	V1.24	sspe	袁伟杰	对齐 sdk 1.24.x 版本

目 录

1 概述

1.1 简介

DUILite SDK 作为思必驰离线语音技术 SDK，集成了一整套成熟的单点语音技术，旨在使第三方应用可以快捷便利地集成和使用思必驰的离线语音技术。

本文档定义 DUILite SDK 的核心能力、使用说明、体系架构和 API 接口，不包含性能指标等。

1.2 适用性

1.2.1 平台支持

- 类 UNIX 系统，如 Linux、Ubuntu、OpenWrt 和 Qnx 等
- Windows 系统
- Android 系统
- IOS 系统

1.2.2 硬件需求

目前市面上大部分硬件配置，如 Android Armv7 的四核 CPU、1.2G 的主频、1G 的 RAM，基本能够满足现有 SDK 模块需求。

不同硬件的具体性能资源指标，如内存，CPU 和实时率等，请联系相关人员进行实测。

1.2.3 输出物

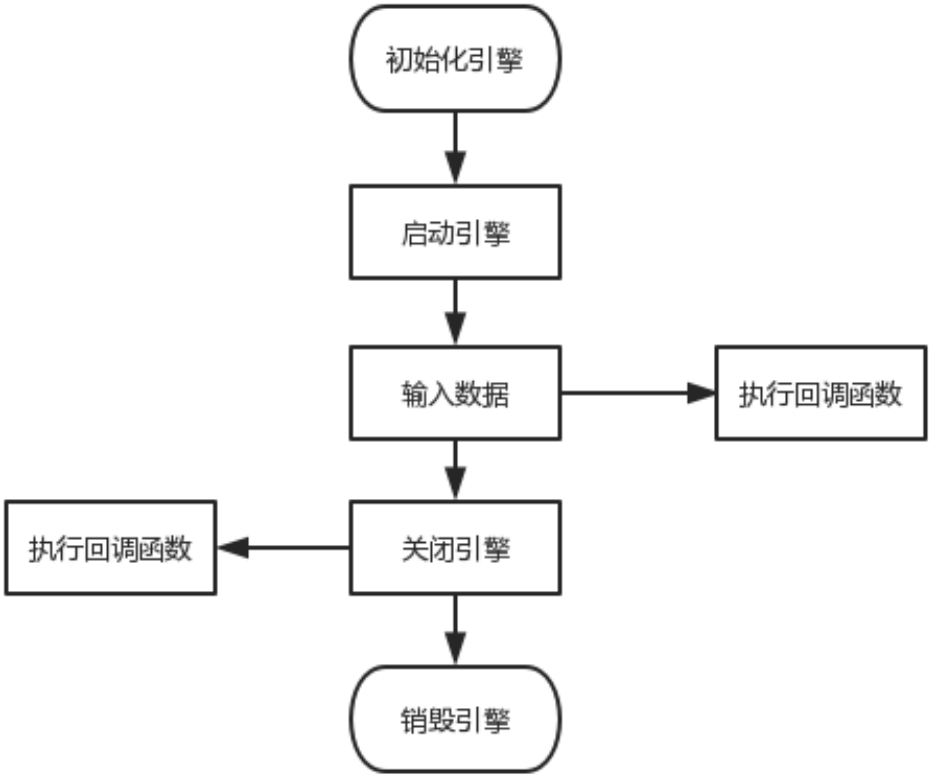
平台系统	输出物	释义
类 UNIX 系统	duilite.h	头文件
	libduilite*.so	动态库文件
Windows 系统	duilite.h	头文件
	libduilite*.dll+libduilite*.lib	动态库文件+导入库文件
Android 系统	libduilite*.so	jni 动态库文件
IOS 系统	duilite.h	头文件
	lib*.a	静态库文件

1.3 注意事项

- 不包含业务逻辑与在线云端功能
- 默认只支持单线程使用，不保证多线程接口调用安全
- 支持多实例使用

1.4 典型调用原则

各个模块接口典型调用顺序如下：



2 软件定义

2.1 回调函数

2.1.1 通用回调函数

*typedef int (*duilite_callback)(void *userdata, int type, char *msg, int len);*

接口能力

负责输出结果，通用回调函数吗，若无特殊说明所有模块均使用此回调函数。

输入参数

userdata - 用户数据，注册回调时传入，回调时透传；

type - 回调数据类型，

DUILITE_MSG_TYPE_JSON 表示 JSON 字符串；

DUILITE_MSG_TYPE_BINARY 表示二进制数据；

DUILITE_MSG_TYPE_TLV 表示 TLV 数据包；

msg - 回调数据，在各个模块中详细说明；

len - 回调数据的长度；

返回值

一般为 0，语音合成模块例外，详见语音合成模块。

2.1.2 模型数据回调函数

```
typedef int (*duilite_model_callback)(void *userdata, int type, char *id,
                                      char **data, int *data_size, int *data_nmemb);
```

接口能力

负责输出模型数据，用于支持模型数据外部存储功能，具体使用方式参见各个模块，目前仅支持声纹模块。

输入参数

userdata - 用户数据，注册回调时传入，回调时透传；

type – 模型数据操作类型，详见各模块中说明；

id – 该条模型数据的标识符；

data – 模型数据，数组；

data_size – 每条模型数据的大小，数组；

data_nmemb – 模型数据的数量；

返回值

成功 – 0；

失败 – 非 0；

2.2 通用功能

2.2.1 duilite_library_load

*int duilite_library_load(char *cfg);*

接口能力

加载库函数，完成初始化与授权，接口阻塞，调用一次即可。

输入参数

cfg – 初始化参数，省略部分为授权相关，授权配置详见附录 3.2。

```
{
    ......
    "prof": {
        "enable": 1,
        "output": "",
    "level": 1
    },

    "deviceId": "abcd",
    "userId": "user",
  "productVersion": "0.1.2",
    "upload": {
        "enable": 1,
        "logId": 141,
        "tmpdir":"upload_dir",
        "tmpdirMaxSize": 104857600,
        "netType": "wifi",
        "mno": "office",
        "clientIP": "20.13.21.24"
    }
}
```

字段	释义	类型

prof	enable	日志开关，1 为开，0 为关，可选，默认为关	int
	output	日志输出路径，可选，默认为标准输出	string
	level	日志等级，默认为 1	int
deviceId		设备唯一标识，用于数据上传	string
userId		用户标识，可选，用于数据上传	string
productVersion		dui 发布产品版本，使用数据上传时必选	string
upload	enable	数据上传开关，1	int

		为开，0 为关，可选 默认为关	
	logId	云端日志 id，必选，统一使用 141	int
	tmpdir	存储目录，需要有可写权限，必选	string
	tmpdirMaxSize	存储目录最大空间，单位为 byte，可选，默认为 100m	int
	netType	客户端网络类型，可选	string
	mno	网络运营商类型，可选	string
	clientIP	客户端 ip，可选	string

日志等级取值如下：

Level	Value	Type

VERBOSE	1	Int
DEBUG	2	Int
INFO	3	Int
WARNING	4	Int
ERROR	5	Int

返回值

成功 – 0；

失败 – 非 0；

2.2.2 duilite_library_release

void duilite_library_release();

接口能力

释放库资源。

输入参数

无

返回值

无

2.2.3 duilite_library_opt

*int duilite_library_opt(int opt, char *data, int size);*

接口能力

实现一些辅助功能，如获取版本号和检查授权结果等。

输入参数

opt – 功能类型：

opt	功能说明
DUILITE_OPT_VERSION_STR_GET	获取版本信息
DUILITE_OPT_AUTH_CHECK	duilite_library_load 时授权失败后，获取服务端授权错误码
DUILITE_OPT_AUTH_TRY	执行授权操作
DUILITE_OPT_UPLOAD_MODE_GET	重新从服务端更新数据上传的权限

data – 调用者外部申请的内存与该块内存大小，作用如下：

opt	data
DUILITE_OPT_VERSION_STR_GET	用于存放输出的版本信息
DUILITE_OPT_AUTH_CHECK	NULL
DUILITE_OPT_AUTH_TRY	输入授权参数，同 duilite_library_load 授权相关参数
DUILITE_OPT_UPLOAD_MODE_GET	NULL

size – data 大小；

返回值

opt	返回值
DUILITE_OPT_VERSION_STR_GET	成功 - 输入参数 data 中版本信息的实际大小 失败 - 非 0
DUILITE_OPT_AUTH_CHECK	成功 – 服务端授权错误码 失败 - 非 0，若为-1，则为授权库返回，一般为本地校验失败
DUILITE_OPT_AUTH_TRY	成功 - 0 失败 - 非 0
DUILITE_OPT_UPLOAD_MODE_GET	成功 - 0 失败 - 非 0

2.3 语法编译

语法编译（gram），基于 ebnf 的语法格式（详见附录 3.3），对输入的文本内容进行解析，生成可用于语音识别的语法网络资源。

2.3.1 duilite_gram_new

```
struct duilite_gram *duilite_gram_new(char *cfg);
```

接口能力

初始化 gram 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path"
}
```

其中，resBinPath 为语法编译资源文件，必选。

返回值

成功 – 引擎指针；

失败 – NULL；

2.3.2 duilite_gram_start

*int duilite_gram_start(struct duilite_gram *gram, char *param);*

接口能力

启动 gram 引擎，开始语法编译。

输入参数

gram – 引擎指针；

param – 启动参数，JSON 字符串，格式如下：

```
{
    "outputPath": "res_file_path",
```

```
    "ebnf": "ebnf_content"
}
或
{
    "outputPath": "res_file_path",
    "ebnfFile": "ebnf_content.xbnf"
}
```

其中 outputPath 为生成的语法网络资源文件存放路径，必选；ebnf 为基于 ebnf 的语法格式的文本内容；ebnfFile 为基于 ebnf 的语法格式的文件。ebnf 与 ebnfFile 二选一即可。

返回值

成功 – 0；

失败 – 非 0；

2.3.3 duilite_gram_delete

*int duilite_gram_delete(struct duilite_gram *gram);*

接口能力

销毁 gram 引擎。

输入参数

gram – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.4 语音识别

语音识别（asr），利用声学模型，语言模型和其他相关的资源，将输入的语音转为文字。

2.4.1 duilite_asr_new

*struct duilite_asr *duilite_asr_new(char *cfg, duilite_callback callback, void *userdata);*

接口能力

初始化 asr 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path",
    "netBinPath": "net_file_path"
}
```

其中，resBinPath 为识别资源文件，包括 grammar 和 ngram 两种，区别见附录 3.4，必选；netBinPath 为语法网络资源文件，即 gram 引擎生成的资源文件，当 resBinPath 使用 ngram 资源时，可选；当 resBinPath 使用 grammar 增量网络资源

时，请勿配置。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程。识别结果通过该函数返回，识别结果示例见附录 3.4；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.4.2 duilite_asr_start

*int duilite_asr_start(struct duilite_asr *asr, char *param);*

接口能力

启动 asr 引擎。

输入参数

asr – 引擎指针；

param – 启动参数，JSON 字符串，格式如下：

```
{
    "env": "...;...;"
}
```

其中 env 可选，具体配置示例见附录 3.4。

返回值

成功 – 0；

失败 – 非 0；

2.4.3 duilite_asr_feed

*int duilite_asr_feed(struct duilite_asr *asr, char *data, int len);*

接口能力

输入音频数据，每次识别操作（每启动引擎视为一次）输入音频数据时长最长为60s，超过限制的数据不会处理，并返回错误。

输入参数

asr – 引擎指针；

data – 音频数据，格式为单通道，16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.4.4 duilite_asr_stop

*int duilite_asr_stop(struct duilite_asr *asr);*

接口能力

停止 asr 引擎，并通过串行调用通用回调函数输出识别结果，一般在语音结束时调用。

输入参数

asr – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.4.5 duilite_asr_cancel

*int duilite_asr_cancel(struct duilite_asr *asr);*

接口能力

取消当前识别操作，并重置 asr 引擎，不会有识别结果输出。

输入参数

asr – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.4.6 duilite_asr_delete

*int duilite_asr_delete(struct duilite_asr *asr);*

接口能力

销毁 asr 引擎。

输入参数

asr – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.5 语音端点检测

语音端点检测（Voice Activity Detection，VAD），检测语音的开始和结束的时间点，过滤静音，提取有效人声，配合语音识别使用。

2.5.1 duilite_vad_new

*struct duilite_vad *duilite_vad_new(char *cfg,*

*duilite_callback callback, void *userdata);*

接口能力

初始化 vad 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path",
    "pauseTime": 300,
    "fullMode": 0
}
```

其中，resBinPath 为资源文件，必选；pauseTime 为语音结束延长时间，单位为毫秒（ms），可选，默认为 0；fullMode 为全双工模式开关，1 为开，0 为关，可选，默认为关，当开启时可以连续输出语音状态检测跳变状态。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；

DUILITE_MSG_TYPE_BINARY 返回切割后音频；DUILITE_MSG_TYPE_JSON 返回检测结果，格式如下：

```
{"status":1}
{"status":2}
```

其中 1 表示音频开始，2 表示音频结束。

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.5.2 duilite_vad_start

*int duilite_vad_start(struct duilite_vad *vad, char *param);*

接口能力

启动 vad 引擎。

输入参数

vad – 引擎指针；

param – 启动参数，JSON 字符串，可选，不需要可设为 NULL，格式如下：

```
{
    "pauseTime": 300
}
```

通过 pauseTime 更新语音结束延长时间。

返回值

成功 – 0；

失败 – 非 0；

2.5.3 duilite_vad_feed

*int duilite_vad_feed(struct duilite_vad *vad, char *data, int len);*

接口能力

输入音频数据，同时串行调用通用回调函数返回音频数据与检测状态变化。

输入参数

vad – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.5.4 duilite_vad_stop

*int duilite_vad_stop(struct duilite_vad *vad);*

接口能力

停止 vad 引擎，并将缓冲区的音频数据处理完毕；一般在收到语音结束状态时调用；注意，不能在通用回调函数中调用该接口。

输入参数

vad – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.5.5 duilite_vad_cancel

*int duilite_vad_cancel(struct duilite_vad *vad);*

接口能力

取消 vad 检测。

输入参数

vad – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.5.6 duilite_vad_delete

*int duilite_vad_delete(struct duilite_vad *vad);*

接口能力

销毁 vad 引擎。

输入参数

vad – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.6 语音唤醒

语音唤醒（wakeup），又称关键词检测（keyword spooting），一般输入经过信号处理后的单通道音频数据。

2.6.1 duilite_wakeup_new

```
struct duilite_wakeup *duilite_wakeup_new(char *cfg,
                                          duilite_callback callback, void
                                          *userdata);
```

接口能力

初始化 wakeup 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path"
}
```

其中，resBinPath 为资源文件，必选。

callback – 通用回调函数，唤醒时返回唤醒结果 json 字符串，格式见附录 3.5。

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.6.2 duilite_wakeup_register

```
int duilite_wakeup_register(struct duilite_wakeup *wakeup, int callback_type,
                            duilite_callback callback, void
                            *userdata);
```

接口能力

设置通用回调函数。需要注意的是，此处设置一些辅助性的函数，非初始化接口设置的唤醒通用回调函数。

输入参数

wakeup – 引擎指针；

callback_type – 通用回调函数类型，如下表：

通用回调函数类型	备注
DUILITE_CALLBACK_WAKEUP_VPRINTCUT	声纹回调，输出的数据送往声纹引擎，输出的数据有音频数

	据和一些 json 字符串，开发者负责将输出数据透传至声纹引擎。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 0；

失败 – 非 0；

2.6.3 duilite_wakeup_start

*int duilite_wakeup_start(struct duilite_wakeup *wakeup, char *param);*

接口能力

启动 wakeup 引擎。

输入参数

wakeup – 引擎指针；

param – 启动参数，JSON 字符串，格式如下：

```
{
    "env": "words=ni hao xiao le;thresh=0.2;",
    "throwWaitWakeup": 0
}
```

其中 env 配置示例见附录 3.5，必选；throwWaitWakeup 表示是否开启待唤醒状

态，1 表示是，0 或者未设置该字段表示否，可选，默认为 0。

返回值

成功 – 0；

失败 – 非 0；

2.6.4 duilite_wakeup_feed

*int duilite_wakeup_feed(struct duilite_wakeup *wakeup, char *data, int len);*

接口能力

输入音频数据，并计算；若唤醒，通过调用通用回调函数返回结果。

输入参数

wakeup – 引擎指针；

data – 音频数据；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.6.5 duilite_wakeup_stop

*int duilite_wakeup_stop(struct duilite_wakeup *wakeup);*

接口能力

停止 wakeup 引擎，并将缓冲区的音频数据处理完毕；一般在收到语音唤醒信号后不需要引擎继续工作时调用；注意，不能在通用回调函数中调用该接口。

输入参数

wakeup – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.6.6 duilite_wakeup_cancel

*int duilite_wakeup_cancel(struct duilite_wakeup *wakeup);*

接口能力

取消当前唤醒工作，并重置引擎。

输入参数

wakeup – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.6.7 duilite_wakeup_set

*int duilite_wakeup_set(struct duilite_wakeup *wakeup, char *param);*

接口能力

配置 wakeup 引擎。

输入参数

wakeup – 引擎指针；

param – 配置参数，JSON 字符串，可选，不需要可设为 NULL，格式如下：

```
{
    "env": "words=ni hao xiao le;thresh=0.4;"
}
```

其中 env 配置示例见附录 3.5，可选，用于动态更新唤醒词的阈值(目前只支持更新 thresh)。

返回值

成功 – 0；

失败 – 非 0；

2.6.8 duilite_wakeup_delete

*int duilite_wakeup_delete(struct duilite_wakeup *wakeup);*

接口能力

销毁 wakeup 引擎。

输入参数

wakeup – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.7 回声消除

回声消除（echo），用于消除设备自身的播放声音，通常输入两个通道的音频数据，输出一个通道。

2.7.1 duilite_echo_new

```
struct duilite_echo *duilite_echo_new(char *cfg,
                                      duilite_callback callback, void *userdata);
```

接口能力

初始化 echo 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path",
    "channels": 2,
    "micNum": 1,
    "sampleFormat": 16
}
```

其中，resBinPath 为资源文件，必选；channels 为输入音频总通道数，必选；micNum 为输入音频的录音通道数，必选；sampleFormat 为单通道采样字节数，必选。

callback – 通用回调函数，返回处理后的音频，内部串行调用，与外部调用线程属于同一线程；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.7.2 duilite_echo_start

*int duilite_echo_start(struct duilite_echo *echo, char *param);*

接口能力

启动 echo 引擎。

输入参数

echo – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.7.3 duilite_echo_feed

*int duilite_echo_feed(struct duilite_echo *echo, char *data, int len);*

接口能力

输入音频数据，进行回声消除处理，并串行调用通用回调函数返回处理后音频数据。

输入参数

echo – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.7.4 duilite_echo_stop

*int duilite_echo_stop(struct duilite_echo *echo);*

接口能力

保留接口。

输入参数

echo – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.7.5 duilite_echo_cancel

*int duilite_echo_cancel(struct duilite_echo *echo);*

接口能力

保留接口。

输入参数

echo – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.7.6 duilite_echo_delete

*int duilite_echo_delete(struct duilite_echo *echo);*

接口能力

销毁 echo 引擎。

输入参数

echo – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.8 线性麦克风阵列

线性麦克风阵列（fespl），支持线性四麦和线性六麦，包含回声消除（echo），声源定位（doa），波束成形（beamforming）和语音唤醒（wakeup）等功能模块。

2.8.1 duilite_fespl_new

*struct duilite_fespl *duilite_fespl_new(char *cfg);*

接口能力

初始化 fespl 引擎，并在后台开启 wakeup，echo 和 beamforming 等计算线程。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "aecBinPath": "res_file_path",
    "wakeupBinPath": "res_file_path",
    "beamformingBinPath": "res_file_path",
    "env": "xxx",
    "rollBack": 1200,
    "maxVolume": 0
}
```

其中，aecBinPath、wakeupBinPath 和 beamformingBinPath 为资源文件，aecBinPath 或 wakeupBinPath 若不配置可设为 OFF，必选；env 同 2.6 节语音唤醒中配置，可选；rollBack 表示 env 中 major 为 1 的唤醒词的音频回滚时长，单位毫秒，可选，默认为 1200；maxVolume 表示大音量检测功能开关，1 表示开（aecBinPath 不为 OFF 才生效），0 表示关，默认为关，可选。

返回值

成功 – 引擎指针；

失败 – NULL；

2.8.2 duilite_fespl_register

*int duilite_fespl_register(struct duilite_fespl *fespl,*

```
int callback_type, duilite_callback callback, void *userdata);
```

接口能力

设置通用回调函数。

输入参数

fespl – 引擎指针；

callback_type – 通用回调函数类型，共有三种，处于不同的线程中，如下表：

通用回调函数类型	备注
DUILITE_CALLBACK_FESPL_WAKEUP	唤醒回调，唤醒时触发，返回 json 字符串，字段释义同 2.6 节语音唤醒结果： `{ "wakeupWord": "xxx", "major": 1, "status": 1, "confidence": 0.465926 }`
DUILITE_CALLBACK_FESPL_DOA	在唤醒回调之后，返回唤醒角度信息，为 json 字符串，格式如下： `{"doa": 95}`
DUILITE_CALLBACK_FESPL_BEAMFORMING	音频回调，实时输出经过信号

	处理后的音频，通常用于识别；当唤醒时，会在唤醒结果与角度回调之后，返回一个 json 字符串：
	`{"wakeup_type": 1}`
	或
	`{"wakeup_type": 2}`
	其中 1 表示，此唤醒词在 env 中配置 major=1，需要回滚音频，在收到该 json 串后，回滚音频开始返回；2 表示不需要回滚。

DUILITE_CALLBACK_FESPL_VPRINTCUT	声纹回调，开发者负责将输出数据透传至声纹引擎，不拆包，不组包。

callback – 通用回调函数，需要注意的是，此模块的回调函数为各处理线程分别调用，与外部主线程不在同一线程；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 0；

失败 – 非 0；

2.8.3 duilite_fespl_start

```
int duilite_fespl_start(struct duilite_fespl *fespl, char *param);
```

接口能力

启动 fespl 引擎。

输入参数

fespl – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.8.4 duilite_fespl_feed

```
int duilite_fespl_feed(struct duilite_fespl *fespl, char *data, int len);
```

接口能力

输入音频数据。

输入参数

fespl – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.8.5 duilite_fespl_stop

*int duilite_fespl_stop(struct duilite_fespl *fespl);*

接口能力

重置引擎，并将残留数据处理完成。

输入参数

fespl – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.8.6 duilite_fespl_set

*int duilite_fespl_set(struct duilite_fespl *fespl, char *param);*

接口能力

设置一些变量，比如大音量状态和 env 更新等。

输入参数

fespl – 引擎指针；

param – 设置参数，全量配置格式如下，请开发者按需选择：

```
{
    "env": "words=ni hao xiao le;thresh=0.2;major=1;",
    "smode": 1,
    "maxVolumeState": 1,
    "wakeupSwitch": 0,
    "doa": 90,
    "lpSwitch": 0,
}
```

其中 env 同 2.6 节语音唤醒中配置，常用于动态更新唤醒词与阈值；smode 用于设置处理模式，1 为普通模式，2 为卖场模式，默认为普通模式；maxVolumeState 用于设置大音量状态，启用大音量检测功能时，在每次 feed 之前调用，0 表示非大音量 1 表示大音量； wakeupSwitch 用于动态开关唤醒，0 为关，1 为开，默认为开；doa 用于外部设置角度，角度在每次唤醒后更新；lpSwitch 用于设置低功耗模式，0 为关，1 为开，默认为关。

返回值

成功 – 0；

失败 – 非 0；

2.8.7 duilite_fespl_get

*int duilite_fespl_get(struct duilite_fespl *fespl, char *param);*

接口能力

获取引擎信息，比如输入，输出通道信息等。

输入参数

fespl – 引擎指针；

param – 获取内容选项，如下：

参数	释义
vprintWavChan	声纹回调输出音频总通道数
bfWavChan	声纹回调输出增强后音频通道数
aecWavChan	声纹回调输出回声消除后音频通道数
vpCbLen	声纹回调输出 tlv 数据包长度，该长度每次输出固定，输入声纹时也保持该长度

返回值

成功 – 0；

失败 – 非 0；

2.8.8 duilite_fespl_delete

*int duilite_fespl_delete(struct duilite_fespl *fespl);*

接口能力

销毁 fespl 引擎。

输入参数

fespl – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.9 环形麦克风阵列

环性麦克风阵列（fespa），包含回声消除（echo），声源定位（doa），波束成形（beamforming）和语音唤醒（wakeup）等功能模块。

2.9.1 duilite_fespa_new

*struct duilite_fespa *duilite_fespa_new(char *cfg);*

接口能力

初始化 fespa 引擎，并在后台开启 wakeup，echo 和 beamforming 等计算线程。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "aecBinPath": "res_file_path",
    "wakeupBinPath": "res_file_path",
    "beamformingBinPath": "res_file_path",
    "env": "xxx",
    "rollBack": 1200,
    "maxVolume": 0
}
```

其中，aecBinPath、wakeupBinPath 和 beamformingBinPath 为资源文件，aecBinPath 或 wakeupBinPath 若不配置可设为 OFF，必选；env 同 2.6 节语音唤醒中配置，可选；rollBack 表示 env 中 major 为 1 的唤醒词的音频回滚时长，单位毫秒，可选，默认为 1200；maxVolume 表示大音量检测功能开关，1 表示开（aecBinPath 不为 OFF 才生效），0 表示关，默认为关，可选。

返回值

成功 – 引擎指针；

失败 – NULL；

2.9.2 duilite_fespa_register

*int duilite_fespa_register(struct duilite_fespa *fespa,*

int callback_type, duilite_callback callback, void

**userdata);*

接口能力

设置通用回调函数。

输入参数

fespa – 引擎指针；

callback_type – 通用回调函数类型，共有三种，处于不同的线程中，如下表：

通用回调函数类型	备注

DUILITE_CALLBACK_FESPA_WAKEUP	唤醒回调，唤醒时触发，返回 json 字符串，字段释义同 2.6 节语音唤醒结果：
	`{`
	`"wakeupWord": "xxx",`
	`"major": 1,`
	`"status": 1,`
	`"confidence": 0.465926`
	`}`

DUILITE_CALLBACK_FESPA_DOA	在唤醒回调之后，返回唤醒角

	度信息，为 json 字符串，格式如下： `{"doa": 95}`

DUILITE_CALLBACK_FESPA_BEAMFORMING	音频回调，实时输出经过信号处理后的音频，通常用于识别；当唤醒时，会在唤醒结果与角度回调之后，返回一个 json 字符串：
	`{"wakeup_type": 1}`
	或
	`{"wakeup_type": 2}`
	其中 1 表示，此唤醒词在 env 中配置 major=1，需要回滚音频，在收到该 json 串后，回滚音频开始返回；2 表示不需要回滚。

DUILITE_CALLBACK_FESPA_VPRINTCUT	声纹回调，开发者负责将输出数据透传至声纹引擎，不拆包，不组包。

callback – 通用回调函数，需要注意的是，此模块的回调函数为各处理线程分别调用，与外部主线程不在同一线程；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 0；

失败 – 非 0；

2.9.3 duilite_fespa_start

*int duilite_fespa_start(struct duilite_fespa *fespa, char *param);*

接口能力

启动 fespa 引擎。

输入参数

fespa – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.9.4 duilite_fespa_feed

*int duilite_fespa_feed(struct duilite_fespa *fespa, char *data, int len);*

接口能力

输入音频数据。

输入参数

fespa – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.9.5 duilite_fespa_stop

*int duilite_fespa_stop(struct duilite_fespa *fespa);*

接口能力

保留接口。

输入参数

fespa – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.9.6 duilite_fespa_set

*int duilite_fespa_set(struct duilite_fespa *fespa, char *param);*

接口能力

设置一些参数，比如大音量状态和 env 更新等。

输入参数

fespa – 引擎指针；

param –设置参数，全量配置格式如下，请开发者按需选择：

```
{
    "env": "words=ni hao xiao le;thresh=0.2;major=1;",
    "maxVolumeState": 1,
     "doa": 90,
     "wakeupSwitch": 1,
     "aecGain": 3.6,
     "lpSwitch": 0,
}
```

其中 env 同 2.6 节语音唤醒中配置，常用于动态更新唤醒词与阈值；maxVolumeState 用于设置大音量状态，启用大音量检测功能时，在每次 feed 之前调用，0 表示非大音量，1 表示大音量；wakeupSwitch 用于动态开关唤醒，0 为关，1 为开，默认为开；doa 用于外部设置角度，角度在每次唤醒后更新；aecGain 用于设置增益，供 aec 使用；lpSwitch 用于设置低功耗模式，0 为关，1 为开，默认为关。

返回值

成功 – 0；

失败 – 非 0；

2.9.7 duilite_fespa_get

*int duilite_fespa_get(struct duilite_fespa *fespa, char *param);*

接口能力

获取引擎信息，比如输入，输出通道信息等。

输入参数

fespa – 引擎指针；

param – 获取内容选项，如下：

参数	释义
vprintWavChan	声纹回调输出音频总通道数
bfWavChan	声纹回调输出增强后音频通道数
aecWavChan	声纹回调输出回声消除后音频通道数
vpCbLen	声纹回调输出 tlv 数据包长度，该长度每次输出固定，输入声纹时

	也保持该长度

返回值

成功 – 0；

失败 – 非 0；

2.9.8 duilite_fespa_delete

*int duilite_fespa_delete(struct duilite_fespa *fespa);*

接口能力

销毁 fespa 引擎。

输入参数

fespa – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.10 语音合成

语音合成（SpeechSynthesizer），又称为文语转换（Text to Speech，tts），将文本信息转化为声音信息，生成 16k 采样率，有符号 16bit 编码的单通道流式音频

数据。

2.10.1 duilite_cntts_new

*struct duilite_cntts * duilite_cntts_new(char *cfg,*

*duilite_callback callback, void *userdata);*

接口能力

初始化 cntts 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "frontBinPath": "res_file_path",
    "backBinPath": "res_file_path",
    "dictPath": "res_file_path",
    "userDict": "res_file_path",
    "optimization": 1
}
```

其中，frontBinPath 为前端资源文件，包含文本归一化，分词和韵律等配置，必选；backBinPath 为后端资源文件，主要为发音人配置，必选；dictPath 为词典文件，必选；userDict 保留参数，可选；optimization 为优化选项，用于降低 CPU 使用率，1 位打开优化，0 位关闭优化，默认关闭优化，可选。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；返回

合成的音频，格式为单通道，16k 采样率，有符号 16bit 编码；合成最后一次回调，数据长度为 0，表示合成结束；若需要中断合成，将回调返回值设置为非 0 即可。

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.10.2 duilite_cntts_start

*int duilite_cntts_start(struct duilite_cntts *cntts, char *param);*

接口能力

启动 cntts 引擎。

输入参数

cntts – 引擎指针；

param – 启动参数，JSON 字符串，可选，不需要可设为 NULL，格式如下：

```
{
    "speed": 1.2,
    "volume": 60,
    "lmargin": 5,
    "rmargin": 10,
    "useSSML": 1
}
```

其中 speed 表示语速，范围为 0.5-2.0，越小越快，默认为 1.0，可选；volume

表示音量，范围为 0-100，默认为 50，可选；lmargin 表示音频头部静音时间，范围为 5-20，默认为 5，可选；rmargin 表示音频尾部静音时间，范围为 5-20，默认为 10，可选；useSSML 表示使用 SSML 进行合成，具体使用方法请参考 SSML 使用文档，默认关闭，可选。

返回值

成功 – 0；

失败 – 非 0；

2.10.3 duilite_cntts_feed

```
int duilite_cntts_feed(struct duilite_cntts *cntts, char *data);
```

接口能力

该接口阻塞，输入合成文本，开始合成，实时通过串行调用通用回调函数返回合成的音频。

输入参数

cntts – 引擎指针；

data – 需要合成的文本，格式为以’\0’结尾的字符串，若开启 ssml，则输入 ssml 标记语言。

返回值

成功 – 0；

失败 – 非 0；

2.10.4 duilite_cntts_set

*int duilite_cntts_set(struct duilite_cntts *cntts, char *param);*

接口能力

设置一些参数，比如发音人资源，实现切换发音人功能；该接口在启动引擎后调用，不可在合成过程中调用。

输入参数

cntts – 引擎指针；

param – 设置参数，格式如下：

```
{
    "backBinPath": "res_file_path"
}
```

其中 backBinPath 为发音人资源文件。

返回值

成功 – 0；

失败 – 非 0；

2.10.5 duilite_cntts_delete

*int duilite_cntts_delete(struct duilite_cntts *cntts);*

接口能力

销毁 cntts 引擎。

输入参数

cntts – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.11 单通道语音降噪

单通道语音降噪（nr），主要针对平稳噪声，多用于语音唤醒之前，对语音唤醒有明显提升效果。

2.11.1 duilite_nr_new

*struct duilite_nr *duilite_nr_new(char *cfg,*

*duilite_callback callback, void *userdata);*

接口能力

初始化 nr 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": " res_file_path"
}
```

其中，resBinPath 为 nr 资源文件，必选。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；处理后音频通过该函数返回；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.11.2 duilite_nr_start

*int duilite_nr_start(struct duilite_nr *nr, char *param);*

接口能力

启动 nr 引擎。

输入参数

nr – 引擎指针；

param – 保留参数，设置为 NULL 即可；

返回值

成功 – 0；

失败 – 非 0；

2.11.3 duilite_nr_feed

*int duilite_nr_feed(struct duilite_nr *nr, char *data, int len);*

接口能力

输入音频数据。

输入参数

nr – 引擎指针；

data – 音频数据，格式为单通道，16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.11.4 duilite_nr_stop

*int duilite_nr_stop(struct duilite_nr *nr);*

接口能力

停止 nr 引擎。

输入参数

nr – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.11.5 duilite_nr_delete

*int duilite_nr_delete(struct duilite_nr *nr);*

接口能力

销毁 nr 引擎。

输入参数

nr – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.12 车载双麦

车载双麦（fespCar），包含回声消除（echo），声源定位（doa），波束成形（beamforming）和语音唤醒（wakeup）等功能模块。

2.12.1 duilite_fespCar_new

*struct duilite_fespCar *duilite_fespCar_new(char *cfg);*

接口能力

初始化 fespCar 引擎，并在后台开启 wakeup，echo 和 beamforming 等计算线程。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "aecBinPath": "res_file_path",
    "wakeupBinPath": "res_file_path",
    "beamformingBinPath": "res_file_path",
    "env": "xxx",
    "rollBack": 1200,
    "maxVolume": 0
}
```

其中，aecBinPath、wakeupBinPath 和 beamformingBinPath 为资源文件，aecBinPath 或 wakeupBinPath 若不配置可设为 OFF，必选；env 同 2.6 节语音唤醒

中配置，可选；rollBack 表示 env 中 major 为 1 的唤醒词的音频回滚时长，单位毫秒，可选，默认为 1200；maxVolume 表示大音量检测功能开关， 1 表示开（aecBinPath 不为 OFF 才生效），0 表示关，默认为关，可选。

返回值

成功 – 引擎指针；

失败 – NULL；

2.12.2 duilite_fespCar_register

```
int duilite_fespCar_register(struct duilite_fespCar *fespCar, int callback_type,
                             duilite_callback callback, void
                             *userdata);
```

接口能力

设置通用回调函数。

输入参数

fespCar – 引擎指针；

callback_type – 通用回调函数类型，共有三种，处于不同的线程中，如下表：

通用回调函数类型	备注

DUILITE_CALLBACK_FESPCAR_WAKEUP	唤醒回调，唤醒时触发，返回 json 字符串，字段释义同 2.6 节语音唤醒结果： `{ "wakeupWord": "xxx", "major": 1, "status": 1, "confidence": 0.465926 }`
DUILITE_CALLBACK_FESPCAR_DOA	在唤醒回调之后，返回唤醒角度信息，为 json 字符串，格式如下： {"doa": 1} 或 {"doa": 2} 其中 1 为主驾方位，2 位副驾方位。
DUILITE_CALLBACK_FESPCAR_BEAMFORMING	音频回调，实时输出经过信号处理后的音频，通常用于识别；当唤醒时，会在唤醒结果与角度回调之后，返回一个 json 字符串：

	{"wakeup_type": 1} 或 {"wakeup_type": 2} 其中 1 表示，此唤醒词在 env 中配置 major=1，需要回滚音频，在收到该 json 串后，回滚音频开始返回；2 表示不需要回滚。
DUILITE_CALLBACK_FESPCAR_VPRINTCUT	声纹回调，开发者负责将输出数据透传至声纹引擎，不拆包，不组包。

callback – 通用回调函数，需要注意的是，此模块的回调函数为各处理线程分别调用，与外部主线程不在同一线程；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 0；

失败 – 非 0；

2.12.3 duilite_fespCar_start

*int duilite_fespCar_start(struct duilite_fespCar *fespCar, char *param);*

接口能力

启动 fespCar 引擎。

输入参数

fespCar – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.12.4 duilite_fespCar_feed

*int duilite_fespCar_feed(struct duilite_fespCar *fespCar, char *data, int len);*

接口能力

输入音频数据。

输入参数

fespCar – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.12.5 duilite_fespCar_stop

*int duilite_fespCar_stop(struct duilite_fespCar *fespCar);*

接口能力

保留接口。

输入参数

fespCar – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.12.6 duilite_fespCar_set

*int duilite_fespCar_set(struct duilite_fespCar *fespCar, char *param);*

接口能力

设置一些参数，比如大音量状态和 env 更新等。

输入参数

fespCar – 引擎指针；

param – 设置参数，格式如下：

```
{
    "env": "words=ni hao xiao le;thresh=0.2;major=1;",
    "maxVolumeState": 1,
     "driveMode": 1,
    "wakeupSwitch": 0
}
```

其中 env 同 2.6 节语音唤醒中配置，常用于动态更新唤醒词与阈值；maxVolumeState 用于设置大音量状态，启用大音量检测功能时，在每次 feed 之前调用，0 表示非大音量，1 表示大音量；driveMode 用于设置驾驶模式，0 为定位模式，按照声源定位；1 为主驾模式；2 为副驾模式；3 为全车模式；wakeupSwitch 用于动态开关唤醒，0 为关，1 为开，默认为开。

返回值

成功 – 0；

失败 – 非 0；

2.12.7 duilite_fespCar_get

*int duilite_fespCar_get(struct duilite_fespCar *fespCar, char *param);*

接口能力

获取引擎信息，比如输入，输出通道信息等。

输入参数

fespCar – 引擎指针；

param – 获取内容选项，如下：

参数	释义
vprintWavChan	声纹回调输出音频总通道数
bfWavChan	声纹回调输出增强后音频通道数
aecWavChan	声纹回调输出回声消除后音频通道数
driveMode	获取驾驶模式，结果同 duilite_fespCar_getDriveMode

返回值

成功 – 0；

失败 – 非 0；

2.12.8 duilite_fespCar_delete

*int duilite_fespCar_delete(struct duilite_fespCar *fespCar);*

接口能力

销毁 fespCar 引擎。

输入参数

fespCar – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.13 声纹识别

声纹识别（vprint），对输入语音的发音人的声音特征进行识别，需要配合信号处理（fespx）或者唤醒（wakeup）使用。

2.13.1 duilite_vprint_new

```
struct duilite_vprint *duilite_vprint_new(char *cfg, duilite_callback callback,
                                          void *userdata);
```

接口能力

初始化 vprint 引擎，接口能力为 duilite_vprint_new 的子集，通过通用回调函数

输出结果，模型升级发生在该阶段。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "vprintBinPath": "res_file_path",
    "asrppBinPath": "res_file_path",
    "modelFile": "model_file_path"
}
```

其中，vprintBinPath 和 asrppBinPath 为资源文件，必选；modelFile 为模型文件，通过注册操作生成，执行测试操作时会读取该文件。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；注册，测试和删除等所有操作的结果都从该回调返回，返回结果见附录 3.6；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.13.2 duilite_vprint_new2

*struct duilite_vprint *duilite_vprint_new2(char *cfg, duilite_callback callback,*

*duilite_model_callback model_callback, void *userdata);*

接口能力

初始化 vprint 引擎，接口能力为 duilite_vprint_new 的超集，通过通用回调函数输出结果，通过模型数据回调函数输出模型数据，模型升级发生在该阶段。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "vprintBinPath": "res_file_path",
    "asrppBinPath": "res_file_path"
}
```

其中，vprintBinPath 和 asrppBinPath 为资源文件，必选。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；注册，测试和删除等所有操作的结果都从该回调返回，返回结果见附录 3.6；

model_callback – 模型数据回调函数，内部串行调用，与外部调用线程属于同一线程；

其中参数 id 由注册人+注册词组成，每个人每个词的注册模型形成单条模型数据：

opt	功能说明
DUILITE_MODEL_MSG_TYPE_VPSELECT	初始化声纹引擎时，加载已注册的声纹模型 会

	查询三次：第一次查询数据个数，第二次查询数据大小，第三次查询所有数据
DUILITE_MODEL_MSG_TYPE_VPUPDATE	重新注册成功后，更新某个 id 的声纹模型数据
DUILITE_MODEL_MSG_TYPE_VPINSERT	注册成功后，插入某个 id 的声纹模型数据
DUILITE_MODEL_MSG_TYPE_VPDELETE	删除某个 id 的声纹模型数据

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.13.3 duilite_vprint_start

*int duilite_vprint_start(struct duilite_vprint *vprint, char *param);*

接口能力

启动 vprint 引擎，并通过通用回调函数输出结果。

输入参数

vprint – 引擎指针；

param – 启动参数，见附录 3.6；

返回值

成功 – 0；

失败 – 非 0；

2.13.4 duilite_vprint_feed

*int duilite_vprint_feed(struct duilite_vprint *vprint, char *data, int len, int type);*

接口能力

输入音频数据。

输入参数

vprint – 引擎指针；

data – 输入数据，为音频数据和唤醒 json 信息，需要保证时序；

len – 输入数据长度；

type – 输入数据类型，

DUILITE_MSG_TYPE_JSON 表示 JSON 字符串；

DUILITE_MSG_TYPE_BINARY 表示二进制数据；

DUILITE_MSG_TYPE_TLV 表示 TLV 数据包；

返回值

成功 – 0；

失败 – 非 0；

2.13.5 duilite_vprint_stop

*int duilite_vprint_stop(struct duilite_vprint *vprint);*

接口能力

停止 vprint 引擎，保留接口。

输入参数

vprint – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.13.6 duilite_vprint_delete

*int duilite_vprint_delete(struct duilite_vprint *vprint);*

接口能力

销毁 vprint 引擎。

输入参数

vprint – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.14 Speex 音频压缩

音频压缩，目前支持 speex 压缩（speexenc），将送往云端的音频压缩，可降低终端网络的上行压力。

2.14.1 duilite_speexenc_new

*struct duilite_speexenc *duilite_speexenc_new(char *cfg,*

*duilite_callback callback, void *userdata);*

接口能力

初始化 speexenc 引擎。

输入参数

cfg – 初始化配置，保留项，传入 NULL 即可；

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；处理后的音频通过该函数返回；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.14.2 duilite_speexenc_start

*int duilite_speexenc_start(struct duilite_speexenc *speexenc, char *param);*

接口能力

启动 speexenc 引擎。

输入参数

speexenc – 引擎指针；

param – 启动参数，JSON 字符串，可选，默认值与格式如下：

```
{
    "quality": 8,
```

```
        "complexity": 2,
        "vbr": 0,
        "rate": 16000,
        "channels": 1,
        "bits": 16
    }
```

其中 quality 表示压缩的质量，质量越高，压缩后的比特率越大，音质也越好，取值范围为 0 到 10 之间的整数；complexity 表示复杂度，复杂度越高，压缩率越高，CPU 使用率越高，音质越好，取值范围为 1 到 10 之间的整数；vbr 为可变比特率，一般情况下设为 0 即可；rate 表示采样率、channels 表示通道数，bits 表示单通道采样字节数。

返回值

成功 – 0；

失败 – 非 0；

2.14.3 duilite_speexenc_feed

*int duilite_speexenc_feed(struct duilite_speexenc *speexenc, char *data, int len);*

接口能力

输入音频数据。

输入参数

speexenc – 引擎指针；

data –音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.14.4 duilite_speexenc_stop

*int duilite_speexenc_stop(struct duilite_speexenc *speexenc);*

接口能力

停止 speexenc 引擎。

输入参数

speexenc – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.14.5 duilite_speexenc_delete

*int duilite_speexenc_delete(struct duilite_speexenc *speexenc);*

接口能力

销毁 speexenc 引擎。

输入参数

asr – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.15 语义理解

语义理解（semantic），对输入的文本进行解析，准确理解并获取文本真实意图。

2.15.1 duilite_semantic_new

```
struct duilite_semantic *duilite_semantic_new(char *cfg,
                                              duilite_callback callback, void *userdata);
```

接口能力

初始化 semantic 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "luaPath": "res_file_path1,res_file_path2",
    "resPath": "res_file_path",
    "vocabPath": "res_file_path"
}
```

其中，luaPath 为 lua 文件目录，存在多个路径时以’，‘分隔，必选；resBinPath 为资源文件，必选；vocabPath 为词典搜索目录，必选。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；结果通过该函数返回，结果如下：

```
{
    "semantics": {
        "request": {
            "slotcount": 4,
            "domain": "地图",
            "action": "地图",
            "param": {
                "intent": "周边搜索",
                "请求类型": "周边",
                "__act__": "inform",
                "poi类型": "充电桩"
            }
        }
    },
    "version": "2018.9.30.15:31:42",
    "systime": 35.823974609375,
    "input": "给我找旁边的充电桩",
    "res": "navi_local.0.0.9"
}
```

结果成员释义如下：

成员	释义
semantics	语义结果
version	语义内核版本

systime	实际处理时间
input	输入内容
res	资源

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.15.2 duilite_semantic_start

*int duilite_semantic_start(struct duilite_semantic *semantic, char *param);*

接口能力

启动 semantic 引擎，处理输入文本。

输入参数

asr – 引擎指针；

param – 启动参数，JSON 字符串，格式如下：

```
{
    "refText":"给我找旁边的充电桩"
}
```

其中 refText 表示输入文本，必选。

返回值

成功 – 0；

失败 – 非 0；

2.15.3 duilite_semantic_delete

*int duilite_semantic_delete(struct duilite_semantic *semantic);*

接口能力

销毁 semantic 引擎。

输入参数

semantic – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.16 多路语音唤醒

多路语音唤醒（nwakeup），为语音唤醒（wakeup）的多通道封装。

2.16.1 duilite_nwakeup_new

*struct duilite_nwakeup *duilite_nwakeup_new(char *cfg,*

duilite_callback callback, void

```
*userdata);
```

接口能力

初始化 nwakeup 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path",
    "channels": 4,
    "parallel_mode": 1,
    "env": "words=ni hao tian mao;thresh=0.20;"
}
```

其中，resBinPath 为资源文件，必选；channels 为原始音频通道数，必选，范围为 1-5；parallel_mode 表示是否开启多线程处理模式，1 为开启，0 为关闭，默认关闭，可选；env 配置示例见附录 3.5，必选。

callback – 通用回调函数，若为单线程模式，内部串行调用，与外部调用线程属于同一线程；若为多线程模式，则内部线程调用，与外部主线程不在同一线程；唤醒时返回唤醒结果 json 字符串如下：

```
{
    "index": 2,
    "result": {
       ......
    }
}
```

其中，index 为通道索引，第一个通道索引 index 为 1；result 中内容同 wakeup 返回结果，格式见附录 3.5。

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.16.2 duilite_nwakeup_start

```
int duilite_nwakeup_start(struct duilite_nwakeup *nwakeup, char *param);
```

接口能力

启动 nwakeup 引擎。

输入参数

nwakeup – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.16.3 duilite_nwakeup_feed

*int duilite_nwakeup_feed(struct duilite_nwakeup *nwakeup, int index, char *data, int len);*

接口能力

输入音频数据，并计算；若唤醒，通过通用回调函数返回结果。

输入参数

nwakeup – 引擎指针；

index – 通道索引，表明输入音频的通道索引，第一个通道 index 为 0；

data – 音频数据；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.16.4 duilite_nwakeup_stop

*int duilite_nwakeup_stop(struct duilite_nwakeup *nwakeup);*

接口能力

停止 nwakeup 引擎，并重置引擎。

输入参数

nwakeup – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.16.5 duilite_nwakeup_cancel

*int duilite_nwakeup_cancel(struct duilite_nwakeup *nwakeup);*

接口能力

取消当前唤醒工作，并重置引擎。

输入参数

nwakeup – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.16.6 duilite_nwakeup_set

*int duilite_nwakeup_set(struct duilite_nwakeup *nwakeup, char *param);*

接口能力

配置 nwakeup 引擎。

输入参数

nwakeup – 引擎指针；

param – 配置参数，JSON 字符串，可选，不需要可设为 NULL，格式如下：

```
{
    "env": "words=ni hao xiao le;thresh=0.4;"
}
```

其中 env 配置示例见附录 3.5，可选，用于动态更新唤醒词的阈值(目前只支持更新 thresh)。

返回值

成功 – 0；

失败 – 非 0；

2.16.7 duilite_nwakeup_delete

```
int duilite_nwakeup_delete(struct duilite_nwakeup *nwakeup);
```

接口能力

销毁 nwakeup 引擎。

输入参数

nwakeup – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.17 分布式麦克风阵列

dmasp（distributed microphone array process，dmasp），用于麦克风分布式放置的场景。

2.17.1 duilite_dmasp_new

```
struct duilite_dmasp *duilite_dmasp_new(char *cfg,
                                        duilite_callback callback, void *userdata);
```

接口能力

初始化 dmasp 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "wakeupBinPath": "res_file_path",
    "resBinPath": "res_file_path",
     "paallel_mode": 0
}
```

其中，resBinPath 为信号资源文件，必选；wakeupBinPath 为唤醒资源文件，必

选；parallel_mode 表示是否开启多线程处理模式，1 为开启，0 为关闭，默认关闭，可选。

callback – 通用回调函数，需要注意的是，此模块的回调函数为各处理线程分别调用，与外部主线程不在同一线程；返回唤醒信息和处理后的音频：

回调返回值类型	备注

DUILITE_MSG_TYPE_JSON	唤醒时触发，返回 json 字符串，字段释义同 2.6 节语音唤醒结果：
	`{`
	`"wakeupWord": "xxx",`
	`"major": 1,`
	`"status": 1,`
	`"confidence": 0.465926`
	`}`

DUILITE_MSG_TYPE_BINARY	信号处理后音频

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.17.2 duilite_dmasp_start

*int duilite_dmasp_start(struct duilite_dmasp *dmasp, char *param);*

接口能力

启动 dmasp 引擎。

输入参数

dmasp – 引擎指针；

param – 启动参数：

```
{
    "env": "words=ni hao xiao pi;thresh=0.29;"
}
```

env 配置示例见附录 3.5，必选。

返回值

成功 – 0；

失败 – 非 0；

2.17.3 duilite_dmasp_feed

*int duilite_dmasp_feed(struct duilite_dmasp *dmasp, char *data, int len);*

接口能力

输入音频数据，进行处理，并回调返回处理后音频数据。

输入参数

dmasp – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.17.4 duilite_dmasp_stop

*int duilite_dmasp_stop(struct duilite_dmasp *dmasp);*

接口能力

停止 dmasp 引擎，并重置引擎。

输入参数

dmasp – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.17.5 duilite_dmasp_cancel

*int duilite_dmasp_cancel(struct duilite_dmasp *dmasp);*

接口能力

取消 dmasp 引擎操作，并重置引擎。

输入参数

dmasp – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.17.6 duilite_dmasp_set

*int duilite_dmasp_set(struct duilite_dmasp *dmasp, char *param);*

接口能力

设置一些参数。

输入参数

dmasp – 引擎指针；

param – 设置参数：

```
{
    "wakeupSwitch":1
}
```

wakeupSwitch 用于动态开关唤醒，0 为关，1 为开，默认为开。

返回值

成功 – 0；

失败 – 非 0；

2.17.7 duilite_dmasp_delete

*int duilite_dmasp_delete(struct duilite_dmasp *dmasp);*

接口能力

销毁 dmasp 引擎。

输入参数

dmasp – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.18 识别++

识别++（asrpp），对输入语音的发音人的性别，情感和年龄进行识别。

2.18.1 duilite_asrpp_new

```
struct duilite_asrpp *duilite_asrpp _new(char *cfg,
                                         duilite_callback callback, void *userdata);
```

接口能力

初始化 asrpp 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": " res_file_path"
}
```

其中，resBinPath 为识别++资源文件，必选。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；结果通过该函数返回，见附录 3.7；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.18.2 duilite_asrpp_start

*int duilite_asrpp_start(struct duilite_asrpp *asrpp, char *param);*

接口能力

启动 asrpp 引擎。

输入参数

asrpp – 引擎指针；

param – 保留参数，设置为 NULL 即可；

返回值

成功 – 0；

失败 – 非 0；

2.18.3 duilite_asrpp_feed

*int duilite_asrpp_feed(struct duilite_asrpp *asrpp, char *data, int len);*

接口能力

输入音频数据。

输入参数

asrpp – 引擎指针；

data – 音频数据，格式为单通道，16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.18.4 duilite_asrpp_stop

*int duilite_asrpp_stop(struct duilite_asrpp *asrpp);*

接口能力

停止 asrpp 引擎，并通过通用回调函数输出结果。

输入参数

asrpp – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.18.5 duilite_asrpp_cancel

*int duilite_asrpp_cancel(struct duilite_asrpp *asrpp);*

接口能力

取消 asrpp 引擎操作，并重置，不会返回结果。

输入参数

asrpp – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.18.6 duilite_asrpp_delete

*int duilite_asrpp_delete(struct duilite_asrpp *asrpp);*

接口能力

销毁 asrpp 引擎。

输入参数

asrpp – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.19 自动增益控制

自动增益控制（automatic gain control，agc），用于使增益自动随信号强度而调整的方法。

2.19.1 duilite_agc_new

```
struct duilite_agc *duilite_agc_new(char *cfg,
                                    duilite_callback callback, void *userdata);
```

接口能力

初始化 agc 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path"
}
```

其中，resBinPath 为资源文件，必选。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；返回处理后的音频；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.19.2 duilite_agc_start

*int duilite_agc_start(struct duilite_agc *agc, char *param);*

接口能力

启动 agc 引擎。

输入参数

agc – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.19.3 duilite_agc_feed

*int duilite_agc_feed(struct duilite_agc *agc, char *data, int len);*

接口能力

输入音频数据，进行处理，并回调返回处理后音频数据。

输入参数

agc – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.19.4 duilite_agc_stop

*int duilite_agc_stop(struct duilite_agc *agc);*

接口能力

停止 agc 引擎，并将缓冲区数据处理完成。

输入参数

agc – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.19.5 duilite_agc_cancel

```
int duilite_agc_cancel(struct duilite_agc *agc);
```

接口能力

取消 agc 引擎操作，并重置引擎。

输入参数

agc – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.19.6 duilite_agc_delete

```
int duilite_agc_delete(struct duilite_agc *agc);
```

接口能力

销毁 agc 引擎。

输入参数

agc – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.20 会议记录（C 类）

会议记录（meeting recorder，mr），C 类会议宝产品的信号处理模块。

2.20.1 duilite_mr_new

*struct duilite_mr *duilite_mr_new(char *cfg);*

接口能力

初始化 mr 引擎，并在后台开启 echo 和 beamforming 等计算线程。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "beamformingBinPath": "res_file_path"
}
```

其中， beamformingBinPath 为资源文件，必选。

返回值

成功 – 引擎指针；

失败 – NULL；

2.20.2 duilite_mr_register

*int duilite_mr_register(struct duilite_mr *mr, int callback_type, duilite_callback callback, void *userdata);*

接口能力

设置通用回调函数。

输入参数

mr – 引擎指针；

callback_type – 通用回调函数类型，共有三种，处于不同的线程中，如下表：

通用回调函数类型	备注
DUILITE_CALLBACK_MR_BEAMFORMING	音频回调，实时输出经过信号处理后的音频。
DUILITE_CALLBACK_MR_POST	音频回调，实时输出经过后处理的音频。
DUILITE_CALLBACK_MR_DOA	实时返回输入源角度信息，为

	json 字符串，格式如下： {"doa": 70}

callback – 通用回调函数，需要注意的是，此模块的回调函数为各处理线程分别调用，与外部主线程不在同一线程；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 0；

失败 – 非 0；

2.20.3 duilite_mr_start

*int duilite_mr_start(struct duilite_mr *mr, char *param);*

接口能力

启动 mr 引擎。

输入参数

mr – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.20.4 duilite_mr_feed

*int duilite_mr_feed(struct duilite_mr *mr, char *data, int len);*

接口能力

输入音频数据。

输入参数

mr – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.20.5 duilite_mr_stop

*int duilite_mr_stop(struct duilite_mr *mr);*

接口能力

保留接口。

输入参数

mr – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.20.6 duilite_mr_set

*int duilite_mr_set(struct duilite_mr *mr, char *param);*

接口能力

设置一些参数。

输入参数

mr – 引擎指针；

param – 设置参数，格式如下：

```
{
    "wpeSwitch": 0
}
```

其中 wpeSwitch 表示去混响的开关，0 为关，1 为开。

返回值

成功 – 0；

失败 – 非 0；

2.20.7 duilite_mr_delete

*int duilite_mr_delete(struct duilite_mr *mr);*

接口能力

销毁 mr 引擎。

输入参数

mr – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.21 家居双麦

家居双麦（fespd），包含回声消除（echo），声源定位（doa），波束成形（beamforming）和语音唤醒（wakeup）等功能模块。

2.21.1 duilite_fespd_new

*struct duilite_fespd *duilite_fespd_new(char *cfg);*

接口能力

初始化 fespd 引擎，并在后台开启 wakeup，echo 和 beamforming 等计算线程。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "aecBinPath": "res_file_path",
    "wakeupBinPath": "res_file_path",
    "beamformingBinPath": "res_file_path",
    "env": "xxx",
    "rollBack": 1200,
    "maxVolume": 0
}
```

其中，aecBinPath、wakeupBinPath 和 beamformingBinPath 为资源文件，aecBinPath 或 wakeupBinPath 若不配置可设为 OFF，必选；env 同 2.6 节语音唤醒中配置，可选；rollBack 表示 env 中 major 为 1 的唤醒词的音频回滚时长，单位毫秒，可选，默认为 1200；maxVolume 表示大音量检测功能开关，1 表示开（aecBinPath 不为 OFF 才生效），0 表示关，默认为关，可选。

返回值

成功 – 引擎指针；

失败 – NULL；

2.21.2 duilite_fespd_register

*int duilite_fespd_register(struct duilite_fespd *fespd,*

int callback_type, duilite_callback callback, void

**userdata);*

接口能力

设置通用回调函数。

输入参数

fespd – 引擎指针；

callback_type – 通用回调函数类型，共有三种，处于不同的线程中，如下表：

通用回调函数类型	备注

DUILITE_CALLBACK_FESPD_WAKEUP	唤醒回调，唤醒时触发，返回 json 字符串，字段释义同 2.6 节语音唤醒结果：
	`{`
	`"wakeupWord": "xxx",`
	`"major": 1,`
	`"status": 1,`
	`"confidence": 0.465926`
	`}`

DUILITE_CALLBACK_FESPD_DOA	在唤醒回调之后，返回唤醒角度信息，为 json 字符串，格式如下： `{"doa": 95}`
DUILITE_CALLBACK_FESPD_BEAMFORMING	音频回调，实时输出经过信号处理后的音频，通常用于识别；当

	唤醒时，会在唤醒结果与角度回调之后，返回一个 json 字符串：
	`{"wakeup_type": 1}`
	或
	`{"wakeup_type": 2}`
	其中 1 表示，此唤醒词在 env 中配置 major=1，需要回滚音频，在收到该 json 串后，回滚音频开始返回；2 表示不需要回滚。

DUILITE_CALLBACK_FESPD_VPRINTCUT	声纹回调，开发者负责将输出数据透传至声纹引擎，不拆包，不组包。

callback – 通用回调函数，需要注意的是，此模块的回调函数为各处理线程分别调用，与外部主线程不在同一线程；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 0；

失败 – 非 0；

2.21.3 duilite_fespd_start

*int duilite_fespd_start(struct duilite_fespd *fespd, char *param);*

接口能力

启动 fespd 引擎。

输入参数

fespd – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.21.4 duilite_fespd_feed

*int duilite_fespd_feed(struct duilite_fespd *fespd, char *data, int len);*

接口能力

输入音频数据。

输入参数

fespd – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.21.5 duilite_fespd_stop

```
int duilite_fespd_stop(struct duilite_fespd *fespd);
```

接口能力

重置引擎，并将残留数据处理完成。

输入参数

fespd – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.21.6 duilite_fespd_set

```
int duilite_fespd_set(struct duilite_fespd *fespd, char *param);
```

接口能力

设置一些变量，比如大音量状态和 env 更新等。

输入参数

fespd – 引擎指针；

param –设置参数，全量配置格式如下，请开发者按需选择：

```
{
    "env": "words=ni hao xiao le;thresh=0.2;major=1;",
    "smode": 1,
    "maxVolumeState": 1,
    "nlmsModeReset": 1,
    "wakeupSwitch": 0,
    "doa": 90,
    "lpSwitch": 0,
}
```

其中 env 同 2.6 节语音唤醒中配置，常用于动态更新唤醒词和阈值；smode 用于设置处理模式，1 为普通模式，2 为卖场模式，默认为普通模式；maxVolumeState 用于设置大音量状态，启用大音量检测功能时，在每次 feed 之前调用，0 表示非大音量 1 表示大音量；nlmsModeReset 常在处理短音频时设置，防止不够收敛，1 为开，0 为关，默认为关； wakeupSwitch 用于动态开关唤醒，0 为关，1 为开，默认为开；doa 用于外部设置角度，角度在每次唤醒后更新；lpSwitch 用于设置低功耗模式，0 为关，1 为开，默认为关。

返回值

成功 – 0；

失败 – 非 0；

2.21.7 duilite_fespd_get

*int duilite_fespd_get(struct duilite_fespd *fespd, char *param);*

接口能力

获取引擎信息，比如输入，输出通道信息等。

输入参数

fespd – 引擎指针；

param – 获取内容选项，如下：

参数	释义
vprintWavChan	声纹回调输出音频总通道数
bfWavChan	声纹回调输出增强后音频通道数
aecWavChan	声纹回调输出回声消除后音频通道数
vpCbLen	声纹回调输出 tlv 数据包长度，该长度每次输出固定，输入声纹时

	也保持该长度

返回值

成功 – 0；

失败 – 非 0；

2.21.8 duilite_fespd_delete

*int duilite_fespd_delete(struct duilite_fespd *fespd);*

接口能力

销毁 fespd 引擎。

输入参数

fespd – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.22 多设备选择

多设备选择（mds，multiple device selection），用于在存在多个语音采集设备的场景中，选择最优输入设备。

2.22.1 duilite_mds_new

*struct duilite_mds *duilite_mds_new(char *cfg,*

*duilite_callback callback, void *userdata);*

接口能力

初始化 mds 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path",
    "channels": 2
}
```

其中，resBinPath 为资源文件，必选；channels 为输入音频通道数，必选。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；返回计算结果，JSON 字符串，格式如下：

```
{"snr": 0.450158}
```

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 引擎指针；

失败 – NULL；

2.22.2 duilite_mds_start

*int duilite_mds_start(struct duilite_mds *mds, char *param);*

接口能力

启动 mds 引擎。

输入参数

mds – 引擎指针；

param – 启动参数：

```
{
    "speechLen": 1000,
    "doa": 90
}
```

其中，speechLen 为输入的音频中，有效语音的长度，单位为毫秒（ms），可选；doa 为输入音频中，说话人角度（多通道输入时使用），可选，根据资源配置而定。

返回值

成功 – 0；

失败 – 非 0；

2.22.3 duilite_mds_feed

*int duilite_mds_feed(struct duilite_mds *mds, char *data, int len);*

接口能力

输入音频数据。

输入参数

mds – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.22.4 duilite_mds_stop

*int duilite_mds_stop(struct duilite_mds *mds);*

接口能力

停止 mds 引擎，并重置。

输入参数

mds – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.22.5 duilite_mds_cancel

*int duilite_mds_cancel(struct duilite_mds *mds);*

接口能力

重置 mds 引擎。

输入参数

mds – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.22.6 duilite_mds_delete

*int duilite_mds_delete(struct duilite_mds *mds);*

接口能力

销毁 mds 引擎。

输入参数

mds – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.23 大间距双麦

大间距双麦（fasp），包含盲分离、降噪和通道选择等功能模块。

2.23.1 duilite_fasp_new

*struct duilite_fasp *duilite_fasp_new(char *cfg);*

接口能力

初始化 fasp 引擎。

输入参数

cfg – 初始化配置，JSON 字符串，格式如下：

```
{
    "resBinPath": "res_file_path"
}
```

其中，resBinPath 为资源文件，必选。

返回值

成功 – 引擎指针；

失败 – NULL；

2.23.2 duilite_fasp_register

*int duilite_fasp_register(struct duilite_fasp *fasp,*

int callback_type, duilite_callback callback, void

userdata);

接口能力

设置通用回调函数。

输入参数

fasp – 引擎指针；

callback_type – 通用回调函数类型，共有两种，如下表：

通用回调函数类型	备注
DUILITE_CALLBACK_FASP_DATA_CHS1	音频回调，进行通道选择时，每次回调输出，该次被选择的处理后的通道数据；未进行通道选择时，则输出处理后第一个通道的数据。
DUILITE_CALLBACK_FASP_DATA_CHS2	音频回调，进行通道选择时，无输出；未进行通道选择时，则输出处理后第二个通道的数据。

callback – 通用回调函数，内部串行调用，与外部调用线程属于同一线程；返回处理后的音频；

userdata – 用户指针，透传，不需要可设为 NULL；

返回值

成功 – 0；

失败 – 非 0；

2.23.3 duilite_fasp_start

*int duilite_fasp_start(struct duilite_fasp *fasp, char *param);*

接口能力

启动 fasp 引擎。

输入参数

fasp – 引擎指针；

param – 保留参数；

返回值

成功 – 0；

失败 – 非 0；

2.23.4 duilite_fasp_feed

*int duilite_fasp_feed(struct duilite_fasp *fasp, char *data, int len);*

接口能力

输入音频数据。

输入参数

fasp – 引擎指针；

data – 音频数据，格式为 16k 采样率，有符号 16bit 编码；

len – 音频数据长度；

返回值

成功 – 0；

失败 – 非 0；

2.23.5 duilite_fasp_stop

```
int duilite_fasp_stop(struct duilite_fasp *fasp);
```

接口能力

重置引擎，并将残留数据处理完成。

输入参数

fasp – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

2.23.6 duilite_fasp_delete

*int duilite_fasp_delete(struct duilite_fasp *fasp);*

接口能力

销毁 fasp 引擎。

输入参数

fasp – 引擎指针；

返回值

成功 – 0；

失败 – 非 0；

3 附录

3.1 模块映射表

模块名	模块代号
GRAM	1
ASR	2
VAD	4
WAKEUP	8
ECHO	16
FESPL	32
FESPA	64
CNTTS	128
NR	256
FDM	512
GENDER	1024
VPRINT	2048
QBYE	4096
SPEEXENC	8192
SEMANTIC	16384
NWAKEUP	32768
DMASP	65536

MR	131072
ASRPP	262144
AGC	524288
AUDIOHANDLER	1048576
FESPCAR	2097152
FESPD	4194304
MDS	8388608
FEDN	16777216
AEC_AGC	33554432
FASP	67108864

3.2 授权

DUILite SDK 必须通过授权方可使用，目前支持如下三种授权方式，配置参数在初始化时输入，具体的申请方式与收费模式，请咨询相关人员。

3.2.1 预烧录

预烧录方式，涉及到产线支持，一般只应用于特殊的嵌入式硬件产品。需要在 DUI 平台产品授权页面下载已生成好的 profile 文件，烧录到设备中去，一台设备对应一个 profile 文件，不需要联网，sdk 配置如下：

```
{
    "productId": "xxx",
```

```
    "savedProfile": "xxx",
    "deviceProfile": "xxx"
}
```

参数	释义
productId	DUI 平台产品 ID
deviceProfile	deviceProfile 文件内容
savedProfile	deviceProfile 文件路径

deviceProfile 与 savedProfile 二选一即可，若都填写，则使用 deviceProfile。

3.2.2 预登记

预登记方式不涉及到产线支持，推荐嵌入式产品使用此种方式，需要开发者在控制台上传所有设备的 DeviceName 列表的文本文件，即白名单，并要求设备在一次运行时进行了联网注册，只有在白名单内的设备才会注册成功, sdk 配置如下：

```
{
    "productId": "xxx",
    "savedProfile": "xxx",
    "productKey": "xxx",
    "productSecret": "xxx",
    "devInfo": {
        "deviceName": "xxx",
        "platform": "xxx",
        "productId": "xxx",
        "deviceId": "xxx",
        "clientName": "xxx",
        "instructionSet": "xxx",
        "chipModel": "xxx"
    }
}
```

参数		释义
productId		DUI 平台产品 ID
savedProfile		保存 deviceProfile 文件的路径
productKey		与 productSecret 一起作为产品密钥
productSecret		与 productKey 一起作为产品密钥
devInfo		设备详细信息，部分可选
	deviceName	设备标识，一台设备对应一个设备标识，必选
	platform	设备平台类型，如 Linux，必选
	productId	DUI 平台产品 ID，可选
	deviceId	设备 ID，可选
	clientName	客户端名字，可选
	instructionSet	指令集，可选
	chipModel	芯片型号，可选

devInfo 中的 deviceName，用于标识唯一的一台设备，可以使用设备 MAC ID 等信息。若两台设备使用相同的 deviceName，则前一台设备会下线，只允许最近使用该 deviceName 的设备访问服务。

3.2.3 动态注册

动态注册方式与预登记配置的参数是相同的，不同之处在于动态注册使用的

deviceName 不需要进行预登记，在设备运行时进行动态的注册即可。同一个产品动态注册和预登记方式的 productKey 和 productSecret 是不相同的。

3.2.4 aiengine 授权

Aiengine 授权主要为兼容老用户，sdk 配置如下：

```
{
    "appKey": "xxx",
    "secretKey": "xxxx",
    "deviceId": "xxx",
    "provision": "xxx",
    "serialNumber": "xxx"
}
```

参数	释义
appKey	产品 ID
secretKey	产品安全码
deviceId	设备 ID，由小写字母和数字组成，不含特殊字符
provision	授权文件路劲
serialNumber	下载的授权序列号路径

3.3 动态语法编译 XBNF

3.3.1 语法规则

XBNF 文件采用 ebnf 语法作为基础语法，EBNF (extended Backus-Naur Form，扩展巴科斯-瑙尔范式)是表达作为描述计算机编程语言和形式语言的正规方式的上下文无关文法的元语法 (metalanguage)符号表示法。

3.3.2 常用符号说明

用途	符号表示	描述
注释	#	该符号之后的文本为注释文本
定义	=	为变量定义内容
终止	;	表达式的结束
分割	|	列举，从中选择一个
可选	[....]	用户描述当前语句重复 0 或 1 次
分组	(....)	描述当前内容在一个组合内，多与”|”搭配使用
0 到多的重复	{....}	被包含语句重复 0 到多次
1 到多的重复	<....>	被包含语句重复 1 到多次
指定范围的重复	/min=a,max=b /	描述钱一个语句重复[a,b]次

3.3.3 基本语法说明

ebnf 语法有两部分组成：定义变量及主执行语句，例如：

```
$hello=你好;# 定义变量 变量名由字母或者中文组成
(\<s\> $hello \<\s\>) # 主执行语句
```

一般在前后加上“<s>”、“</s>”，用于表明句子的开始及结束。变量在使用的

时候，使用“$变量名“引用，以空格结束。

3.3.4 语义设定

a) K，V 项的设置

在扩展属性语法“/.../”中设置“K”、“V”，用于识别反馈语义，如：

```
$digit=(1|2|3|4|5|6|7|8|9|0);
$phone_num=($digit/min=1,max=11,k="number"/);
$expr1=(打电话)/k="action" , v="call"/ $phone_num;
(\<s\> ($expr1) \<\/s\> )
```

如果扩展属性语法“/.../”中存在“K”值，即当前结果是要做语义输出的，如果没有指定 V”，则采用对应的识别结果作为对应的键值；如果指定 V”，则使用 V”做为输出。

b) 层级嵌套的设定

语法属性支持层级的设计，输出结果为 JSON，如：

```
$digit=(1|2|3|4|5|6|7|8|9|0);
$phone_num=($digit/min=1,max=11,k="number"/);
$expr1=(( 打 电 话 )/k="action" , v="call"/ $phone_num) /k="test"/;
(\<s\> ($expr1) \<\/s\> )
```

如果识别结果为“打电话给 123456”，对应的语义输出为：{“test”：{“action”：”call”,”number”：”123456”}}。

c) 扩展的属性

同时支持自定义扩展属性，用于 JSON 输出，如：

```
$NUM=( $DIGIT /min=1,max=11,k="number",check="hook" /)
```

其 对 应 的 输 出 为 {”number”:”123456”,”check ”:”hook”}，check=“hook”即为自定义属性。

3.3.5 语法文件样例

```
$phone_num=(1386000000|15152439000|119|110); $person=(小明|金钟|刘德华|张学友)/k="person"/;
$phone_action=(打[电话]给 | 电话给 | 呼叫 | 联系 | 致电 | 拨通 | 拨打)/action="call"/;
$cmd=($phone_action ($person | $phone_num)) /domain="phone"/;
$appF=(QQ|微信|相机|手电筒|图库|地图)/k="Fun"/;
$app_action=(打开| 开启| 启动)/action="OPEN"/;
$OPEN=($app_action $appF) /domain="open"/;
(\<s\> (( $cmd ) | ( $OPEN ) ) \<\/s\> )
```

3.4 语音识别引擎启动参数及结果示例

识别资源分为两种，grammar 与 ngram，区别如下：

- grammar 就是基于 EBNF (extended Backus-Naur Form)的语法格式生成的语言模型，限定了说法的内容；
- ngram 是基于这样一种假设，第 n 个词的出现只与前面 n-1 个词相关，而与其它任何词都不相关，是用语料训练出来的，支持的说法不固定。

grammar 识别默认结果示例如下：

```
{
    "version": "1.9.20.2018.12.21.11:12:39",
    "res": "res_file_path",
    "eof": 1,
    "rec": "你 好 小 乐 打 开 空 调",
    "conf": 0.674354,
```

```
    "wavetime": 4823,
    "vadtime": 4823,
    "prutime": 0,
    "systime": 135,
    "rectime": 0,
    "sestime": 137,
    "delaytime": 0,
    "delayframe": 0,
    "net_type": 0,
    "post": {
        "sem": {
            "operation": "open",
            "object": "空调",
            "domain": "device_control"
        }
    }
}
```

ngram 识别默认结果如下：

```
{
    "version": "1.9.20.2018.12.21.11:12:39",
    "res": "res_file_path",
    "eof": 1,
    "rec": "你好 小 乐 打开 空调",
    "conf": 0.674354,
    "wavetime": 4823,
    "vadtime": 4823,
    "prutime": 0,
    "systime": 147,
    "rectime": 0,
    "sestime": 148,
    "posttime": 2,
    "delaytime": 2,
    "delayframe": 0,
    "net_type": 1,
    "post": {}
}
```

融合(grammar+ngram)识别默认结果如下：

```
{
    "grammar": {
        "version": "1.9.20.2018.12.21.11:12:39",
```

```
        "res": "res_file_path",
        "eof": 1,
        "rec": "你 好 小 乐 打 开 空 调",
        "conf": 0.674354,
        "wavetime": 4823,
        "vadtime": 4823,
        "prutime": 0,
        "systime": 152,
        "rectime": 0,
        "sestime": 156,
        "posttime": 3,
        "delaytime": 3,
        "delayframe": 0,
        "net_type": 0,
        "post": {
            "sem": {
                "operation": "open",
                "object": "空调",
                "domain": "device_control"
            }
        }
    },
    "ngram": {
        "version": "1.9.20.2018.12.21.11:12:39",
        "res": "res_file_path",
        "eof": 1,
        "rec": "你好 小 乐 打开 空调",
        "conf": 0,
        "wavetime": 4823,
        "vadtime": 4823,
        "prutime": 0,
        "systime": 152,
        "rectime": 0,
        "sestime": 156,
        "posttime": 3,
        "delaytime": 3,
        "delayframe": 0,
        "net_type": 1,
        "post": {
            "sem": {
                "operation": "open",
                "object": "空调",
                "domain": "device_control"
```

```
                }
            }
        }
    }
```

各字段释义如下表：

字段	释义
version	解码器版本
res	解码资源版本
eof	识别状态标志：音频结束为 1，未结束为 0
rec	识别结果
conf	置信度
wavetime	实际处理的音频时间（vad 切出的音频时间）
vadtime	同 wavetime
prutime	解码过程中的剪枝耗时
systime	实际解码时间（CPU 时间）
rectime	多线程下的解码时间，当前为 0
sestime	实际解码时间
delaytime	音频送完到输出结果的延时
delayframe	音频送完到输出结果这段时间里处理的

		音频帧数
net_type		net 类型，0 为 grammar，1 为 ngram
post		后处理输出
	sem	语义信息

3.4.1 实时反馈

使用实时反馈功能，增加 env 配置如下：

```
"use_frame_split=1;"
```

其中，1 表示打开实时反馈；0 或者未设置该参数，表示关闭实时反馈。

实时反馈结果示例如下：

```
{"grammar":{"rec":"切 歌","var":0},"ngram":{"rec":"你","var":0}}
{"grammar":{"rec":"你 好 小 乐 切 歌","var":0},"ngram":{"rec":"你好 小 乐","var":0}}
{"grammar":{"rec":"你 好 小 乐 打 开 空 调","var":0},"ngram":{"rec":"你好 小 乐 打开 空调","var":0}}
```

其中"var":0 表示实时反馈结果，根据所使用的资源，输出 grammar，ngram 结果。

3.4.2 语义信息

输出语义信息，增加 env 配置如下：

```
"use_xbnf_rec=1;"
```

其中 1 表示输出语义信息，未设置该参数时默认输出语义信息；0 表示不输出语义信息。

语义信息只在识别最终结果中，示例如下：

```
{
    ......
    "eof": 1,
    ......
    "post": {
        "sem": {
            "operation": "open",
            "object": "空调",
            "domain": "device_control"
        }
    }
}
```

其中 sem 表示语义信息，其内容由根据 xbnf 文件生成的语法网络资源文件或 ngram 资源文件所决定。

3.4.3 结果拼音输出

输出拼音结果，增加 env 配置如下：

```
"use_pinyin=1;"
```

其中，1 表示打开拼音输出；0 或者未设置该参数，表示关闭拼音输出。

```
{
    ......
    "pinyin": "ni hao xiao le da kai kong diao",
    ......
}
```

3.4.4 结果分割符

识别结果分割符，默认为空格，若需要修改，增加 env 配置如下，以","为示例：

```
"rec_wrd_sep=\",\";"
```

识别结果如下：

```
{
    ......
    "rec": "你,好,小,乐,打,开,空,调",
```

```
        ......
    }
```

3.4.5 多候选结果

多候选结果仅在单独 ngram 识别时可用，增加 env 配置如下：

```
    "nbest=n;"
```

其中 n 表示候选结果数量。n 为 3 时，识别结果示例如下：

```
    {
        ......
        "nbest": ["你好 小 乐 打开 空调", "你好 小 勒 打开 空调", "你好 晓 乐 打开 空调"],
        ......
    }
```

3.4.6 融合识别 ngram 置信度

融合识别时，若需要输出 ngram 识别结果的置信度，增加 env 配置如下：

```
    "hold_conf=1;"
```

其中 1 表示输出 ngram 识别结果的 confidence；0 或未设置该选项表示不输出 ngram 识别结果的 confidence。

识别结果示例如下：

```
    {
        "grammar": {
            ......
        },
        "ngram": {
            ......
            "conf": 0.674354,
            ......
        }
    }
```

3.4.7 oneshot

用于提高 ngram 的 oneshot 识别准确率，增加 env 配置如下：

```
"one_shot_json=\"wakeup_word.json\";"
```

其中 wakeup_word.json 文件格式及示例如下：

```
{
    "one_shot_words": [
        "你好小勒",
        "你好小德"
    ]
}
```

识别结果示例如下：

```
{
    ......
    "rec": "你好小勒 打开 空调",
    ......
}
```

3.4.8 动态加载

在某些场景下，存在语法网络资源占用内存较大的情况，比如电话黄页等。可以通过将语法网络资源分开编译，动态加载的方法优化内存占用，下面举例说明：

duilite_asr_new 传入的 netBinPath 语法网络资源所对应的 xbnf 文件将复杂说法内容抽离，使用@符号引用在外部定义的复杂说法，格式及示例如下：

```
$DEVICE_CMD_OPEN = ((打开)/operation="open"/@DEVICE);
$DEVICE_CMD_CLOSE = ((关闭)/operation="close"/@DEVICE);
$DEVICE_CMD_ALL = ($DEVICE_CMD_OPEN|$DEVICE_CMD_CLOSE) /domain="device_control"/;
$SEMANTIC_OUTPUT = @WAKEUP_WORD ($DEVICE_CMD_ALL);
( \<s\> ($SEMANTIC_OUTPUT) \<\/s\> )
```

增加 env 配置如下：

```
"expand_fn=\"slot.json\";"
```

其中 slot.json 文件格式及示例如下：

```
{
    "slot": [{
        "name": "DEVICE",
        "path": "device.slot.bin"
    }, {
        "name": "WAKEUP_WORD",
        "path": "wakeup_word.slot.bin"
    }]
}
```

需要注意的是，slot.json 中的 name 字段的值需要和 xbnf 文件中定义与引用的变量名保持一致。

device.slot.bin 所对应的 xbnf 文件中对 DEVICE 中定义：

```
$DEVICE = (电视|空调|电灯|净化器)/k="object"/;
( \<s\> ($DEVICE) \<\/s\> )
```

wakeup_word.slot.bin 所对应的 xbnf 文件中对 WAKEUP_WORD 中定义：

```
$WAKEUP_WORD = [你好小乐];
( \<s\> ($WAKEUP_WORD) \<\/s\> )
```

3.4.9 热词更新

单独使用 grammar 的增量网络资源，可实现热词更新，增加 env 配置如下：

```
"dynamic_list=\"word1,word2,word3\";blacklist=\"word1,word2,word3\
";"
```

dynamic_list 为热词白名单；blacklist 为热词黑名单；若命中黑名单，则识别结果为空。

3.4.10 filler

filler 用于减少误识别，增加 env 配置如下：

```
"use_filler=1;filler_penalty_score=2.0;"
```

use_filler 为该功能开关，1 为开，0 为关，默认关闭；filler_penalty_score 为惩罚分数，可选，默认为 2.0。

3.5 语音唤醒引擎启动参数及结果示例

3.5.1 启动参数

启动参数示例：

```
{"env": "words=ni hao xiao le;thresh=0.2;dcheck=1;vad=1;"}
```

字段	释义	示例
words	唤醒词，拼音形式，词之间逗号分隔，字之间空格分隔，必选。	words=ni hao xiao le,ni hao xiao chi;
thresh	唤醒词的阈值，具体取值请咨询技术对接人员，必选。	thresh=0.2,0.4;
dcheck	唤醒后唤醒词解码功能，1 表示打开，0 表示关闭，	dcheck=1,0;

	默认关闭，可选	
major	设置主唤醒词，1 为是，0 位否，默认为否，可选	major=1,0;
vad	唤醒中 vad 的开关，1 表示打开，0 表示关闭，默认为 1，可选。	vad=1;

当有多个唤醒词时，各参数需要一一对应，如：

```
{"env": "words=ni hao xiao le,ni hao xiao chi;thresh=0.2,0.4;" }
```

3.5.2 结果说明

status 为 1：

```
{
    "version": "res_file_path",
    "lib_version": "1.5.6",
    "vad_version": "version",
    "status": 1,
    "wakeupWord": "ni hao xiao le",
    "major": 0,
    "boundary": 0,
    "confidence": 0.390171,
    "frame": 58,
    "wakeup_frame": 51,
    "abs_feed_frame": 57,
    "words": {
        "ni hao xiao le": 0.2000,
        "ni hao xiao chi": 1.5000
    }
}
```

status 为 2：

```
{
    "version": "res_file_path",
    "lib_version": "1.6.8",
    "vad_version": "version",
    "status": 2,
    "wakeupWord": "ni hao xiao chi",
    "major": 1,
    "confidence": 0.506597,
    "pre_start_offset": 45760,
    "start_offset": 45760,
    "end_offset": 11200,
    "speech_len": 34560,
    "abs_feed_bytes": 309440,
    "delay": 157.766113,
    "speed": "normal",
    "words": {
        "ni hao xiao chi": 0.4000
    }
}
```

各字段释义如下表：

成员	释义
version	资源文件版本
lib_version	算法版本
vad_version	若在唤醒中使用了 vad，则表示资源版本， 可选
status	唤醒状态，详见图表 3.5.3
wakeupWord	唤醒词
major	是否为主唤醒词
confidence	唤醒置信度

status:1	boundary	边界信息开关
	frame	唤醒帧号
	wakeup_frame	通过 vad 后的唤醒帧号
	abs_feed_frame	唤醒已处理的绝对帧序号
status:2	pre_start_offset	一般同 start_offset
	start_offset	唤醒词起始点相对于已输入音频的偏移量
	end_offset	唤醒点相对于已输入音频的偏移量
	speech_len	实际唤醒音频长度
	abs_feed_bytes	唤醒已处理的字节数
	abs_recv_bytes	唤醒已接收字节数
	delay	略
	speed	语速，取值有 normal、fast、slow
words		通过 env 配置的唤醒词列表

3.5.3 唤醒状态码

语音唤醒引擎，唤醒回调中的唤醒状态如下：

状态值	释义
-1	唤醒出现异常
0	等待唤醒
1	已唤醒，一般收到该状态，表明唤醒成

	功，若是多麦信号处理配合声纹使用则无该状态。
2	唤醒边界，输出边界信息
3	唤醒重启
4	预唤醒，达到低阈值，准备进入唤醒状态，常用来保存音频
5	唤醒音频信息，配合声纹使用，该信息后立马返回唤醒音频

3.6 声纹识别引擎启动参数及结果示例

声纹识别引擎启动参数示例：

```
{
    "env": "bfchannel=5;aecchannel=4;outchannnel=1;name=aispeech;
audionum=3;word=xiao wei xiao wei;op=register;snr=20;thresh=1.0f"
}
```

成员	释义
bfchannel	配合单通道唤醒时无须配置；配合信号处理使用时必选，增强后音频通道数
aecchannel	配合单通道唤醒时无须配置；配合信号处理使用时必选，回声消

	除后音频通道数
outchannnel	配合单通道唤醒时无须配置；配合信号处理使用时必选，表示使用几个通道作为声纹输入，通常为 1 或回声消除后音频通道数，视项目而定
name	注册人姓名
audionum	音频数量
word	注册词，支持配置相似发音词： 如 word=主声纹注册词,相似发音,相似发音词...; 声纹内核会自动将这几个词视为同一个词。
op	操作类型
snr	可选，信噪比
thresh	可选，在测试模式下，外部微调阈值；阈值会动态更新到默认阈值+外部微调阈值
sensitivity_level	可选，在测试模式下，调整识别率灵敏度，默认为 0；若设置，该值应小于 toal_sensitivity_level

声纹识别引擎结果示例：

```
{
    "libversion": "1.2.1.4",
    "binversion": "res_file_path",
    "cid": "2730f0e",
    "option": "Register",
```

```
    "state": 0,
    "registers": [{}],
  "snr": 51,
    "result": {}
}
```

成员		释义
libversion		库版本
binversion		资源版本
cid		声纹算法版本信息
option		操作类型，表明此次结果的类型
state		操作状态
registers		注册人信息
snr		信噪比信息
result	score	得分
	scene	场景
	thresh	阈值
	sensitivity_level	识别灵敏度等级
	word	注册词
	register	注册人
	time	略
	speech	略

	RTF	略
total_sensitivity_level		仅创建引擎时返回，识别率灵敏度等级总数，不同资源的不同等级表现性能会根据项目定制

声纹引擎操作状态如下：

状态值	释义
0	成功
1	未知失败
2	用户尚未注册
3	用户未注册过该文本
4	用户已经注册过该文本
5	注册用户数量已满
6	不支持该性别
7	不支持该词语
8	用户的注册音频数量不够
9	模型不兼容，需要重新注册，一般在创建引擎时
10	升级模型结束
11	语速过快
12	语速过慢
13	信噪比过低
14	音频截幅

15	存储模型失败,写文件异常,清空写入失败的模型文件,引擎可以正常运行,引擎实例内存中还有注册人信息
16	解析模型失败,读文件异常,清空读取失败的模型文件,引擎可以正常运行,注册人列表为空
17	识别率灵敏度等级非法，一般在测试模式
18	音频长度不足，本次无法识别

下面对各个操作场景举例说明:

a) 创建引擎

```
    struct duilite_vprint *vprint =duilite_vprint_new(vprint_cfg,
_callback, NULL);
    if (!vprint) {
        ......
        goto error;
    }
```

返回结果如下:

```
    {
        "libversion": "1.2.1.9",
        "binversion": "res_file_path",
        "option": "NewInstance",
        "cid":"2730f0e",
        "state": 0,
        "registers": [{
            "name": "aispeech",
            "wordlist": [{
                "gender": "male",
                "word": "ni hao xiao le"
            }]
        }]
    }
```

其中 state 为 0 表示创建成功；state 若为 9，则表示模型不匹配，需要重新训

练；state 若为 10，则表示模型升级成功。

b) 注册

开始注册：

```
    char *vprint_register_param = "{{\"env\":\"bfchannel=5;aecchannel=4;outchannnel=1;name=aispeech;audionum=3;word=ni hao xiao le;op=register;snr=20;\"}}";
    if (0 != duilite_vprint_start(vprint, vprint_register_param)) {
        ......
        goto error;
    }
```

返回结果如下：

```
    {
        "libversion": "1.2.1.9",
        "binversion": "res_file_path",
        "option": "RegisterStart",
        "cid":"2730f0e",
        "state": 0
    }
```

开始注册成功后，输入音频和 json 信息，进行注册，返回结果如下：

```
    {
        "libversion": "1.2.1.9",
        "binversion": "res_file_path",
        "option": "Register",
        "cid":"2730f0e",
        "state": 0,
         "snr": 51,
        "registers": [{
            "name": "aispeech",
            "wordlist": [{
                "gender": "male",
                "word": "ni hao xiao le"
            }]
        }]
    }
```

若 state 为 8 表示用户的注册音频数量未达标，即未达到开始注册时传入的 audionum 的值，需要继续输入音频，进行注册；若 state 为 11、12、13、14 则表示音频质量问题，需要重新输入注册音频。

status 为 0 表示注册成功，registers 中为已经注册的人的信息。

c) 测试

开始测试：

```
char *vprint_test_param = "{\"env\": \"bfchannel=5;aecchannel=4;
outchannnel=1;op=test;snr=20;\" }";
  if (0 != duilite_vprint_start(vprint, vprint_test_param)) {
      ......
      goto end;
  }
```

返回结果如下：

```
{
    "libversion": "1.2.1.9",
    "binversion": "xxx",
    "option": "TestStart",
    "cid":"2730f0e",
    "state": 0
}
```

其中 state 为 0 表示测试准备完成，可以开始进行测试。

输入音频和 json 信息，开始测试，返回结果如下：

```
{
    "libversion": "1.2.1.9",
    "binversion": "xxx",
    "option": "Test",
    "cid":"2730f0e",
    "snr": 51,
```

```
    "result": {
        "score": 0,
        "thresh": -5,
        "word": "ni hao xiao le",
        "register": "aispeech",
        "time": 17.468994,
        "speech": 3.953187,
        "RTF": 0.004419
    },
    "state": 0
}
```

若 state 为 11、12、13、14 则表示音频质量问题，需要重新输入注册音频。

status 为 0 表示测试成功，其中 register 为识别出来的注册人姓名，若为 others 则代表不是系统中的注册人。

d) 查询

开始查询：

```
char *vprint_query_param = "{\"env\": \"op=query;\" }";

if (0 != duilite_vprint_start(vprint, vprint_query_param)) {
    ......
    goto error;
}
```

返回结果如下：

```
{
    "libversion": "1.2.1.9",
    "binversion": "xxx",
    "option": "Query",
    "cid":"2730f0e",
    "state": 0,
    "registers": [{
        "name": "aispeech",
        "wordlist": [{
            "gender": "male",
```

```
            "word": "ni hao xiao le"
        }]
    }]
}
```

其中 state 表示查询成功，registers 中为系统中已经注册的注册人信息。

e) 重新训练

重新训练与注册的逻辑类似，一般在模型不匹配时进行重新训练。

开始重新训练：

```
    char *vprint_update_param = "{\"env\":\" bfchannel=5;
aecchannel=4;outchannnel=1;name=aispeech;audionum=3;word=ni hao xiao
le;op=update;snr=20;\"}";
    if (0 != duilite_vprint_start(vprint, vprint_update_param)) {
        ......
        goto error;
    }
```

返回结果如下：

```
{
    "libversion": "1.2.1.9",
    "binversion": "res_file_path",
    "option": " UpdateStart",
    "cid":"2730f0e",
    "state": 0
}
```

开始重新训练成功后，输入音频和 json 信息，进行重新训练，返回结果如下：

```
{
    "libversion": "1.2.1.9",
    "binversion": "res_file_path",
    "option": "Update",
     "cid":"2730f0e",
    "state": 0,
     "snr": 51,
    "registers": [{
        "name": "aispeech",
        "wordlist": [{
```

```
            "gender": "male",
            "word": "ni hao xiao le"
        }]
    }]
}
```

若 state 为 8 表示用户的重新训练音频数量未达标，即未达到开始重新训练时传入的 audionum 的值，需要继续输入音频，重新训练；若 state 为 11、12、13、14 则表示音频质量问题，需要重新输入重新训练音频。

status 为 0 表示重新训练成功，registers 中为已经注册的人的信息。

f) 删除

开始删除：

```
char *vprint_unregister_param = "{\"env\": \"name=aispeech;word=ni hao xiao le;op=unregister;\" }";

if (0 != duilite_vprint_start(vprint, vprint_unregister_param)) {
    ......
    goto end;
}
```

返回结果如下：

```
{
    "libversion": "1.2.1.9",
    "binversion": "xxx",
    "option": "UnRegister",
    "cid":"2730f0e",
    "state": 0,
    "registers": []
}
```

其中 state 为 0 表示删除成功；register 为系统中还剩下的人。

g) 追加注册

开始追加注册：

```
    char *vprint_append_param = "{{\"env\":\"bfchannel=5;
aecchannel=4;outchannnel=1;name=aispeech;
word=ni hao xiao le;op=append;snr=2;\"}}";
    if (0 != duilite_vprint_start(vprint, vprint_append_param)) {
        ......
        goto error;
    }
```

返回结果如下：

```
    {
        "libversion": "1.2.1.9",
        "binversion": "res_file_path",
        "option": "AppendStart",
        "cid":"2730f0e",
        "state": 0
    }
```

开始追加注册成功后，输入音频和 json 信息，进行追加注册，返回结果如下：

```
    {
        "libversion": "1.2.1.9",
        "binversion": "res_file_path",
        "option": "Append",
         "cid":"2730f0e",
        "snr": 51,
         "state": 0,
        "registers": [{
            "name": "aispeech",
            "wordlist": [{
                "gender": "male",
                "word": "ni hao xiao le"
            }]
        }]
    }
```

若 state 为 11、12、13、14 则表示音频质量问题，需要重新追加注册。

status 为 0 表示追加注册成功，registers 中为已经注册的人的信息

3.7 识别++引擎启动参数及结果示例

3.7.1 启动参数

略。

3.7.2 结果说明

```
{
    "libversion": "1.9.36",
    "binversion": "res_file_path",
    "delay": "496.442871",
    "speech": "154342",
    "coreversion": "2.0",
    "emotion": {
        "emotion": "nature",
        "libversion": "1.0.0",
        "delay": "466.605469",
        "binversion": "res_file_path"
    },
    "gender": {
        "gender": "male",
        "libversion": "1.0.0",
        "delay": "10.736084",
        "binversion": "res_file_path"
    },
    "age": {
        "age": "youth",
        "libversion": "1.0.0",
        "delay": "23.771729",
        "binversion": "res_file_path"
    }
}
```

各字段释义如下表：

成员	释义

libversion	算法版本
binversion	资源文件版本
delay	处理延时
speech	处理语音时长
coreversion	算法版本
emotion	情感，取值有 happy、angry、nature
gender	性别，取值有 female、male
age	年龄，取值有 kids、youth、elder

4 常见错误码

错误码	数值	释义
SpeechNoError	0	成功，无错误
SpeechAuthError	-9999	授权错误
SpeechReadFileError	-9998	读文件失败
SpeechWriteFileError	-9997	写文件失败
SpeechRemoveFileError	-9996	删除文件失败
SpeechStatFileError	-9995	获取文件信息失败
SpeechNotDir	-9994	目录不存在
SpeechOpenDirError	-9993	打开目录失败
SpeechMakeDirError	-9992	创建目录失败
SpeechDirPathInvaild	-9991	目录路径非法
SpeechWsaInitError	-9990	Wsa 初始化失败
SpeechInputNull	-9899	输入为空
SpeechNotJson	-9898	非 Json 格式
SpeechNoField	-9897	配置字段缺失
SpeechNoFile	-9896	文件缺失
SpeechNoMem	-9895	内存申请失败
SpeechNoInstance	-9894	引擎指针为空
SpeechCoreError	-9893	内核错误

SpeechCoreNotSupport	-9892	内核不支持该功能
SpeechExceedLimit	-9891	超过相应限制
SpeechSpeexChannelError	-9799	Speex 通道错误
SpeechSpeexRateError	-9798	Speex 比特率错误
SpeechSpeexOggInitError	-9797	Ogg 初始化错误
SpeechSpeexStateError	-9796	Speex 引擎错误
SpeechNWakeupIndexError	-9699	多路唤醒通道索引错误
SpeechUploadInitError	-9599	数据上传模块初始化失败