API of Darwin Conversational AI Platform

API of Darwin Conversational AI Platform.

access service

- ACCOUNT : contact with api_issue@xiaoda.ai to apply server URL and tocken
- PROTOCOL: HTTPS
- METHOD : POST

protocol

request

request分为`query`和`event`两类。

- `query` : 用户正常的对话请求；
- `event` : 用于客户端事件，用于通知chatbot用户登录、退出、超时未响应等事件的发生；

query

```
{
    "query"   : { "query": "你好啊",
                   "confidence":1.0
                 },
    "userContext"  : { "source" : "xiaoai", "accessTocken" : "xxxxxx"},
    "session" : "userId",
    "agent"   : "indentifyCode"
}
```

- `query.query` : 用户的对话内容，ASR的结果，为utf-8格式的字符串；
- `query.confidence` : 如果用户的对话内容是经过ASR处理后得到，该项为ASR的信心概率。浮点数，范围`0 ~ 1.0`。如果无法获得则默认填写`1.0`；
- `userContext.source` : 用于标识用户来源（例如小爱音箱填写“xiaoai”）。需要提前发邮件到api_issue@xiaoda.ai进行协商申请；
- `userContext.accessTocken` : 非必须字段。如果用户使用了oauth2鉴权，那么这里填写用户对应的access tocken值。具体参见oauth2标准协议。
- `session` : session id，用于区分不同session。此处暂时填写用户的ID（不超过32位的字符串，只可包含字母、数字和下滑线并以字母开头）；
- `agent` : 接收对话的skill agent的名字，协商取值；例如幸运数字技能填写“indentifyCode”；

event

```
{
    "event"   : { "name"     : "open-skill-indentifyCode",
                "content"  : {}
            },
    "userContext"  : { "source" : "xiaoai", "accessTocken" : "xxxxxx"},
    "session" : "userId",
    "agent"   : "indentifyCode"
}
```

- `event.name`：事件名；
  - `open-skill-indentifyCode`：技能被打开，规则为`"open-skill-" + 技能名`；
  - `quit-skill-indentifyCode`：技能退出，规则为`"quit-skill-" + 技能名`；
  - `no-response-indentifyCode`：用户在技能内一定时间内没有响应，规则为`"no-response-" + 技能名`；
  - `play-finish-indentifyCode`：媒体资源播放结束，规则为`"play-finish-" + 技能名`；该事件需要在`content`中携带媒体资源的url：`content : {url : "https://www.xiaodamp.com/audio/5.mp3"}`
  - `record-finish-indentifyCode`：客户端录音结束，规则为`"record-finish-" + 技能名`；该事件需要在`content`中携带录音资源的media id：`content : {mediaId : "xxxxxxxxxxx"}`
  - `record-fail-indentifyCode`：客户端录音结束，规则为`"record-fail-" + 技能名`；
  - 其它事件名及参数，可以自定义；
- `event.content`：用于事件携带参数。具体格式由不同的事件决定。
- `userContext.source`：用于标识用户来源（例如小爱音箱填写“xiaoai”）。需要提前发邮件到api_issue@xiaoda.ai进行协商申请；
- `userContext.accessTocken`：非必须字段。如果用户使用了oauth2鉴权，那么这里填写用户对应的access tocken值。具体参见oauth2标准协议。
- `session`：session id，用于区分不同session。此处暂时填写用户的ID（不超过32位的字符串，只可包含字母、数字和下滑线并以字母开头）；
- `agent`：接收对话的skill agent的名字，协商取值；例如幸运数字填写“indentifyCode”；

response

Robot返回的消息格式如下：

```
{
    "intents": [
        {
            "name": "say-hi",
            "confidence": 1
        }
    ],
    "reply": [
        "你好啊，欢迎使用幸运数字",
        "你的幸运数字是 12345",
    ],
    "data" : [
    ]
}
```

- `intent`：数组；用户的对话被识别的意图以及对应的信心概率指数，一般可以不用处理；
- `reply`：数组；返回给用户的对话，可以一次回复多句。端侧根据需要可以将所有回复合并成后单句后播报给用户；
- `data`：指令数组；目前支持如下指令：
  - `{"type" : "quit-skill"}`：指示技能退出，关闭麦克风；
  - `{"type" : "tts", "text" : "听完音频后请回答"}`：指示播放对应文字的tts；
  - `{"type" : "play-audio", "url" : "http://www.xiaodamp.cn/audio/5.mp3"}`：指示播放指定的音频文件；
  - `{"type" : "start-record"}`：指示开始录音；
  - `{"type" : "play-record", "mediaId" : "xxxxxxxxxx"}`：指示播放录音，`mediaId`标识录音文件资源id；
- **注意**：如果`response`里面包含了`reply`字段，则`reply`优先于`data`。也就是客户端侧优先播放`response.reply`，然后再按照`response.data`里面的指令顺序播放。

```
// 例：听音乐猜歌手的response可以构造为如下格式：

{
    "intents": [
        {
            "name": "guess-song-player",
            "confidence": 1
        }
    ],
    "reply": [
        "猜猜下面歌曲的歌手，请听音乐："
    ],
    "data" : [
        {"type" : "audio", "url" : "http://www.xiaodamp.cn/audio/5.mp3"},
        {"type" : "tts", "text" : "音乐播放完成，请说出对应的歌手吧"}
    ]
}
```

sdk

以下是nodejs SDK手册。经 `node 8.11.1`版本测试通过，其它更低版本的不支持`class`,`const`,`let`,`async`,`await`等特性的node版本暂不可用。

install

```
npm install darwin-sdk
```

quick start

```
```

```
const Chatbot = require('darwin-sdk').Chatbot;
const Query = require('darwin-sdk').Query;
const Response = require('darwin-sdk').Response;


/*********************************************************
 * prepare config.json for chatbot_url, agent and source
 * {
 *     "chatbot_url" : "https://xiaodamp.com/api/chatbot/",
 *     "agent"  : "indentifyCode",
 *     "source" : "xiaoai"
 * }
*********************************************************/
const config = require('./config')

const chatbot = new Chatbot(config.chatbot_url, config.agent,
config.source)
const rsp = await chatbot.dispose(new Query('test-darwin-user-1', '你好'))
console.log(rsp.getReply())
```

api

Chatbot

Chatbot API reference

```
const Chatbot = require('darwin-sdk').Chatbot
```

Constructor

Initialize new chatbot.

`const chatbot = new Chatbot(url, agent, source)`

Param	Type	Description
url	`String`	chatbot service url
agent	`String`	chatbot agent name
source	`String`	optional: client source

上面参数中：url是darwin平台的服务地址。目前需要发邮件到api_issue@xiaoda.ai进行申请获得； agent是对应的技能引擎的名称，已有技能需要发邮件到api_issue@xiaoda.ai进行获得； source指示了技能访问者的客户端来源，目前已支持如下客户端。其它新的接入端需要申请协商新的source值。

音箱类型	source
小米智能音箱	`xiaoai`

音箱类型	source
百度智能音箱	`dueros`
叮咚智能音箱	`dingdong`
天猫智能音箱	`aligenie`
华为智能音箱	`huawei`
360智能音箱	`360`

dispose

dispose a request.

`await chatbot.dispose(userId, request)`

Param	Type	Description
userId	String	less than (or equal to) 32 bit
request	Request's subclass	eg. Query、Event、OpenSkillEvent ...

```
await chatbot.dispose(new Query('test-darwin-user-1', '你好'))
```

注意：该方法为异步。返回值是Response类型

request

Request分为Query和Event，Event又具体有OpenSkillEvent,QuitSkillEvent,NoResponseEvent,PlayFinishEvent,RecordFinishEvent； 所有的Request共享了以下接口：

function	description
`setAccessTocken(tocken)`	设置access tocken, 具体对应`userContext.accessTocken`字段
`setSource(source)`	指示客户端source值, 具体对应`userContext.source`字段
`setDisplay(enable)`	指示客户端是否支持屏显，具体对应`userContext.supportDisplay`字段

Query

Constructor

Initialize new query.

`const query = new Query(userId, text)`

Param	Type	Description
userId	`String`	user id
text	`String`	query text

```
const Query = require('darwin-sdk').Query

const query = new Query('user1', 'hello')
await chatbot.dispost(query)
```

Event

Constructor

Initialize new Event.

`const event = new event(userId, eventType)`

Param	Type	Description
userId	`String`	user id
eventType	`String`	type of event

除了专有event(OpenSkillEvent,QuitSkillEvent,NoResponseEvent,PlayFinishEvent,RecordFinishEvent)的event type已指定外，其它event type需自定义。不同type的event可以指定携带不同的参数，见`addContent`；

addContent

add parameter for event

Param	Type	Description
key	`String`	parameter key
value	`String`	parameter value

```
const Event = require('darwin-sdk').Event

const event = new Event('user1', 'playFinish')
event.addContent('audio', 'audio1.mp3')

await chatbot.dispost(event)
```

OpenSkillEvent

Constructor

Initialize new OpenSkillEvent. Indicate enter skill.

`const event = new OpenSkillEvent(userId)`

Param	Type	Description
userId	`String`	user id

```
const OpenSkillEvent = require('darwin-sdk').OpenSkillEvent

const event = new OpenSkillEvent('user1')
await chatbot.dispost(event)
```

QuitSkillEvent

Constructor

Initialize new QuitSkillEvent. Indicate quit skill.

`const event = new QuitSkillEvent(userId)`

Param	Type	Description
userId	`String`	user id

```
const QuitSkillEvent = require('darwin-sdk').QuitSkillEvent

const event = new QuitSkillEvent('user1')
await chatbot.dispost(event)
```

NoResponseEvent

Constructor

Initialize new NoResponseEvent. Indicate no response of user when openning mic.

`const event = new NoResponseEvent(userId)`

Param	Type	Description
userId	`String`	user id

```
const NoResponseEvent = require('darwin-sdk').NoResponseEvent

const event = new NoResponseEvent('user1')
await chatbot.dispost(event)
```

PlayFinishEvent

Constructor

Initialize new PlayFinishEvent. Indicate audio playing finished.

`const event = new PlayFinishEvent(userId, url)`

Param	Type	Description
userId	`String`	user id
url	`String`	audio url

```
const PlayFinishEvent = require('darwin-sdk').PlayFinishEvent

const event = new PlayFinishEvent('user1',
'https://www.xiaodamp.com/audio/5.mp3')
await chatbot.dispost(event)
```

RecordFinishEvent

Constructor

Initialize new RecordFinishEvent. Indicate record finished.

`const event = new RecordFinishEvent(userId, mediaId)`

Param	Type	Description
userId	`String`	user id
mediaId	`String`	record media id

```
const RecordFinishEvent = require('darwin-sdk').RecordFinishEvent

const event = new RecordFinishEvent('user1', 'record-file-id')
await chatbot.dispost(event)
```

RecordFailEvent

Constructor

Initialize new RecordFailEvent. Indicate record failed.

`const event = new RecordFailEvent(userId)`

Param	Type	Description

Param	Type	Description
userId	`String`	user id

```
const RecordFailEvent = require('darwin-sdk').RecordFailEvent

const event = new RecordFailEvent('user1')
await chatbot.dispost(event)
```

response

Response API reference

Response封装了从darwin chatbot返回的消息体，对常用字段进行了封装。

attribute	return type	Description
body	`object`	消息体原始内容
getReplies()	`Array`	response.reply
getReply	`String`	将`response.reply`数组内的所有语句合并成一句返回
getInstructs	`Array`	response.data
getIntents	`Array`	将`response.intents`数组内的所有的意图名映射成的数组
hasTts	`boolean`	`response.data`中是否包含tts指令
hasInstructOfQuit	`boolean`	`response.data`中是否有指令指示客户端技能退出

```
const rsp = await chatbot.dispose(new Query('test-darwin-user-1', '你好'))
console.log(rsp.getReply())
if (rsp.hasInstructOfQuit()) {
    // ...
}
```

getInstructs

`getInstructs()`返回响应消息中的携带的指令数组。

目前约定了如下格式的指令，其它的格式可协商定义。

- `{"type" : "quit-skill"}`：指示技能退出，关闭麦克风；
- `{"type" : "tts", "text" : "听完音频后请回答"}`：指示播放对应文字的tts；
- `{"type" : "play-audio", "url" : "http://www.xiaodamp.cn/audio/5.mp3"}`：指示播放指定的音频文件；
- `{"type" : "start-record"}`：指示开始录音；

- `{"type" : "play-record", "mediaId" : "xxxxxxxxxx"}`：指示播放录音，`mediaId`标识录音文件资源id；

其它

如果运行时想打开sdk的debug打印，可以在启动时加上 `DEBUG=darwin:*`，例如`DEBUG=darwin:* node index.js`。 有问题请提issue到api_issue@xiaoda.ai