Server API

参考手册

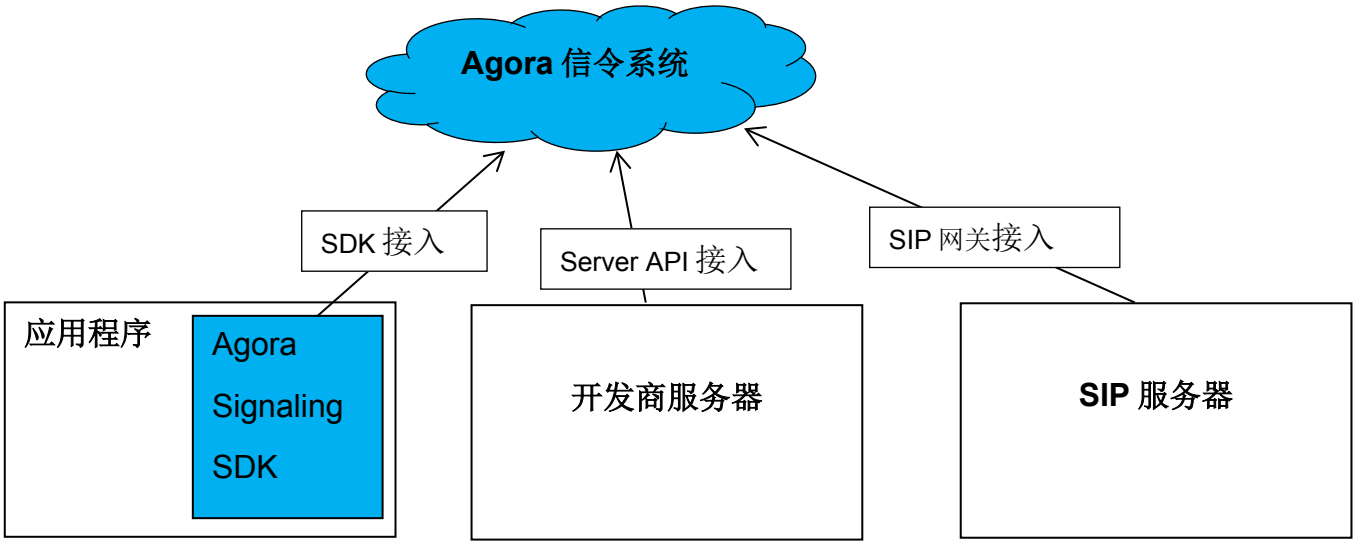
support@agora.io

目录

简介

用户可以通过以下三种方式接入 Agora 信令系统：

- **SDK 接入**：详情参见各平台 Agora Signaling SDK 参考手册。
- **SIP 网关接入**：如有需要，请联系 support@agora.io 开通此功能。
- **Server API 接入**：本文主要介绍通过 Server API 接入 Agora 信令系统。



功能

Server API 接入方式主要支持以下功能：

- **账号消息系统：**
  - Server API：免登录
  - 查阅用户状态
  - 订阅用户状态属性变化
  - 给用户发送消息
- **频道系统：**
  - 查询频道成员
- **呼叫系统：**
  - 订阅呼叫事件
  - 发起呼叫

更多详细内容，见 Server API 参考。

Server API 参考

本章介绍了用 Server API 方式接入 Agora 信令系统的 API 调用方法。

调用函数

以下为 HttpAPI 类的示例代码：

```
class HttpAPI:
    def __init__(self, appId, appCertificate, url=AGORA_SERVER_URL):
        self.appId = appId
        self.appCertificate = appCertificate
        self.callid = 1
        self.url = url

    def call(self, func, **kargs):
        self.callid += 1

        req = {
            '_appId': self.appId,
            '_callid' : self.callid,
            '_timestamp': datetime.datetime.now().isoformat(),
            '_function' : func,
        }
        req.update(kargs)

        keys = req.keys()
        keys.sort()
        signstr = ''.join(k+str(req[k]) for k in keys)
        signstr = signstr.lower()
        sign = hmac.new(self.appCertificate, signstr, hashlib.sha1).hexdigest()
        req['_sign'] = sign

        resp = requests.post(self.url, data=json.dumps(req))
        return resp.json()
```

```
    def sign(self,req):
        keys = req.keys()
        keys.sort()
        s = ''.join(k+str(req[k]) for k in keys)
        s = s.encode('utf-8')
        s = s.lower()
        print s
        sign = hmac.new(self.appCertificate, s, hashlib.sha1).hexdigest()
        return sign


########################################
#Start From Here
########################################
appId = '805d28c445a749a6b6195efbf120e0da'
appCertificate = 'b1f1ccc64cc348269c4dff653fabc8ce'


api=HttpAPI(appId, appCertificate)
#print api.call('test_ping')
print "subscribing"
print api.call('subscribe_online', url=MY_SERVER_URL)
print "querying"
print api.call('query_online', account="2222" )
print api.call('query_online', account="2226" )
print api.sign({"_callid":"qunar14648335798990529","_function":"query_online","_timestamp":"2016-06-02T10:13:00.823526","_appId":appId,"acount":"2222"})
```

调用地址

Agora Server API 服务器地址：http://api.sig.agora.io/api1

发送请求

请求方式如下所示：

```
POST, application/json :
{
"_appId” : appId
"_callid" : 请求唯一标示,
```

```
    "_timestamp" : 请求时间,
    "_sign" : 签名,
    "_function" : 函数名,

    其他参数
}
```

示例代码：

```
{
"_appId" : "***************",
"_callid" : 2,
"_timestamp" : "2016-05-10T10:01:06.233779",
"_sign" : "44e775783ddab92c425089f238bae88d911ee00b",
"_function" : "subscribe_pstn"
"url" : "http://127.0.0.1:40000/",
}
```

生成签名(_sign)

按照以下步骤生成上文所需的签名(_sign):

1. 将 key 排序。
2. 将所有的 key/value 合并。

   注：请求以 json 格式进行发送：{“name1”:”value1”, …, “nameN”:”valueN”}.这里的 nameN 和 valueN 分别对应 key 和 value。
3. 用 appCertificate，以 hmac_sha1 签名，得到 hexdegist。

以下为 python 代码：

```
keys = req.keys()
keys.sort()
signstr = ''.join(k+str(req[k]) for k in keys)
signstr = signstr.lower()
signstr = to_utf8(signstr)
appCertificate = to_utf8(appCertificate)
```

```
sign = hmac.new(appCertificate, signstr, hashlib.sha1).hexdigest()
```

发送通知

向厂商发送事件通知：

1. 设置服务器 url。

   在订阅事件时，厂商会给 Agora 一个服务器的 url。

2. 事件回调。

   事件发生时，Agora 将事件 POST 到厂商设置的 url。

```
POST，application/json :
{
"_timestamp" : 请求时间,
"_sign" : 签名,
"_event" : 事件名,
其他参数
}
```

订阅 PSTN 呼叫事件(subscribe_pstn)

subscribe_pstn(url)

厂商可以通过该接口订阅到本厂商所有的 PSTN 呼叫事件。

参数名称	描述
url	厂商服务器的 url

Agora 发送事件通知：

当厂商调用 unsubscribe_pstn 时,Agora 停止发送事件通知。

示例代码：

```
subscribe_pstn ( url="http://127.0.0.1:40000" )
```

订阅用户上下线事件(subscribe_online)

subscribe_online(url)

厂商可以通过该接口订阅用户上下线通知。

参数名称	描述

url	厂商服务器的 url

示例代码：

```
subscribe_online ( url="http://127.0.0.1:40000" )
```

Agora 发送用户上下线事件通知：

- 上线通知：

```
{"account": "xxx112", "_event": "online", "sign":
"ab4baa230c172aaed1af73605af3eff30c40e714", "is_online": true, "_timestamp":
"2016-07-28T01:51:04.345436"}
```

- 下线通知:

```
{"account": "xxx112", "_event": "online", "sign":
"ab4baa230c172aaed1af73605af3eff30c40e714", "is_online": false, "_timestamp":
"2016-07-28T01:52:04.345436"}
```

当厂商调用 unsubscribe_online 时，Agora 停止发送用户上下线事件通知。

查询用户在线状态(query_online)

query_online(account)

厂商调用本接口查询用户在线状态。

参数名称	描述
account	对方登录厂商 app 的账号

示例代码：

```
query_online(account="xxx112")
```

Agora 发送结果：

- 不在线:

```
{'account': 'xxx112', 'result': 'ok', 'is_online': false}
```

- 在线：

```
{'account': 'xxx112', 'result': 'ok', 'is_online': true}
```

服务端账号订阅事件(start_server)

厂商通过特殊的账号订阅事件，可以通过该账号发起 VOIP 呼叫和发送频道消息。

start_server(account="xxx", url="xxx")

参数名称	描述
account	用户登录厂商 app 的账号
url	厂商的服务器 url

登录之后可以通过该账号发起 VOIP 呼叫和加入频道。

有 VOIP 呼叫事件或频道事件时，Agora 会将事件 POST 到厂商指定的 URL。

示例代码：

```
start_server(account="my_server", url="http://my.server.vendorX.com/agora/events/")
```

发起 VOIP 呼叫(voip_invite)

voip_invite(account="xxx ",kargs='{"peer":"xxx", "channelName":"xxx", "extra":"{}"}')

voip_invite_bye(account="xxx ",kargs='{"peer":"xxx", "channelName":"xxx", "extra":"{}"}')

该接口用于呼叫一组用户，从而实现服务端发起多人混合呼叫。在发起 VOIP 呼叫前，需先做服务端账号订阅。

kargs 是个字符串。字符串里是一段 json，这段 json 的各个参数是做 voip_invite 时需要的参数。

参数名称	描述
account	用户自己登录厂商 app 的账号
peer	被叫用户登录厂商 app 的账号
channelName	频道名
extra	暂时设置为空。

示例代码：

```
voip_invite(account="my_server",kargs='{"peer":"xxx112", "channelName":"demoroom111", "extra":"{}"}')
```

发起呼叫后将收到以下事件列表：

- 响铃

```
{"src": "xxx112", "_timestamp": "2016-07-28T03:38:56.789488", "time_start": 1469677136.687212, "dst": "my_server", "_event": "msg", "sign": "a08565997f407660aa7617fd8c0e767213107e23", "content": "{\"peer\": \"xxx112\", \"peeruid\": 4001000108, \"channel\": \"demoroom111\", \"extra\": \"\"}", "flag": "v1:E:30", "t": "voip_invite_ack"}
```

- 接听

```
{"src": "xxx112", "_timestamp": "2016-07-28T03:38:07.141114", "time_start": 1469677087.122603, "dst": "my_server", "_event": "msg", "sign": "a5237a2404ecc446e4ff1ab5c8fd54041396cf7b", "content": "{\"peer\": \"xxx112\", \"peeruid\": 4001000108, \"channel\": \"demoroom111\", \"extra\": \"\"}", "flag": "v1:E:30", "t": "voip_invite_accept"}
```

- 挂断:

```
{"src": "xxx112", "_timestamp": "2016-07-28T03:38:14.754274", "time_start": 1469677094.737811, "dst": "my_server", "_event": "msg", "sign": "276f0d2b4c40c6d0c11dd7952abfb75d26dc42fd", "content": "{\"peer\": \"xxx112\", \"peeruid\": 4001000108, \"channel\": \"demoroom111\", \"extra\": \"\"}", "flag": "v1:E:30", "t": "voip_invite_bye"}
```

落地电话和 VOIP 混合呼叫

采用 SIP 对接的厂商，呼叫被叫带上 sip 前缀后，voip 呼叫会转成 SIP，投递到厂商的落地线路，这一路就是落地电话。

呼叫落地电话的同时，又呼叫了几路厂商的 App，便是混合呼叫。

其他用户 SDK 对用户服务器发起 VOIP 呼叫

未发起呼叫时，如果有用户发起 VOIP 呼叫给用户服务器，Agora 会通知 voip_invite 到厂商指定的 URL。随后的事件也会一一通知。

发送频道消息

该方法用于当用户需要对一个频道发起一条广播消息，通知频道内的用户。

加入频道(channel_join)

channel_join(account="xxx", kargs='"{"name":"xxx"}"')

参数名称	名称
account	用户登录厂商 app 的账号
name	频道名

示例代码:

```
channel_join(account="my_server", kargs='"{"name":"demoroom111"}"')
```

发送频道消息(sendmsg)

channel_sendmsg(account="xxx", kargs='"{"name":"xxx", "msg":"xxx"}"')

参数名称	名称
account	对方登录厂商 app 的账号
name	频道名
msg	要发到频道的消息。

示例代码:

```
channel_sendmsg(account="my_server", kargs='"{"name":"demoroom111",
"msg":"your_msg_here"}"')
```

接收频道消息:

{"src": "notify", "_timestamp": "2016-07-28T03:42:05.296785", "time_start": 1469677325.280321, "dst": "my_server", "_event": "msg", "sign": "8d559ef4524ee7ffd2bd5022d0f12d52f258cc3f", "content": "{\"msg\": \"ts:1469677325\", \"account\": \"my_server\", \"uid\": 4001000133, \"channel\": \"demoroom111\"}", "flag": "v1:E:180", "t": "channel_msg"}

注：自己发出去的消息，自己也会收到一份。