Server API Reference Manual v1.0

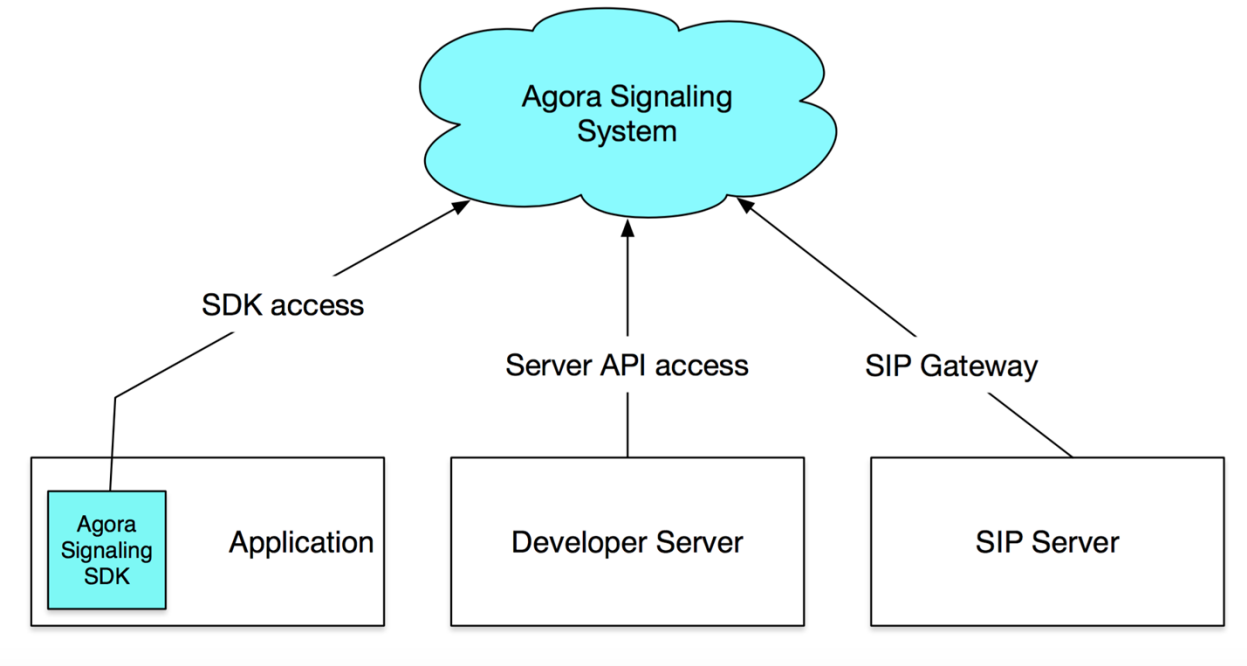
support@agora.io

Contents

Introduction

The users can access the Agora signaling system in three ways:

- **SDK Access**: refer to the specific platform reference manual for Agora Signaling SDK.
- **SIP Gateway Access**: contact support@agora.io to enable this function if necessary.
- **Server API Access**: this document mainly introduces how to access the Agora Signaling System using this method.



Function

The following functions are supported:

- Account Information System：
  - Server API: no logins required
  - Query the user status
  - Subscribe the changes of user status and attributes
  - Send messages to users
- Channel System：
  - Query the channel users
- Call System：

- Subscribe the calling events
- Initiate a call

For more information, see Server API Reference.

Server API Reference

This section introduces the API interfaces for the users who need to access the Agora signaling system using Server API.

How to Call the functions?

The following lists the sample code of HttpAPI class:

```
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"})
```

Server Address of Agora Server API

The server address of Agora Server API: http://api.sig.agora.io/api1

How to Send Request?

```
    POST, application/json :
    {
    "_appId" : appId
    "_callid" : unique identity of the call request,
    "_timestamp" : request time,
    "_sign" : signature,
    "_function" : function name,

    Other Parameters
}

Sample Code:

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

How to Generate the Signature(_sign)?

To generate the signature(_sign) required above:
1. Sort the keys.
2. Combine all the keys and values.
   The request is sent in json format {“name1”:”value1”, …, “nameN”:”valueN”}.
   Here nameN key and valueN are the mentioned keys and values.
3. Use appCertificate with hmac_sha1 to generate hexdegist.

The following is the required python code:

```
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()
```

How to Notify the Developers of the Events?

1. Set the server url.
   When subscribing the event, the developer gives Agora a server url.

2.Callback events.

When an event occurs, Agora POST the event to the developer configured url.

```
POST，application/json :
{
"_timestamp" : request time
"_sign" : signature,
"_event" : event name,
other parameters
}
```

Subscribe PSTN Call Event (subscribe_pstn)

subscribe_pstn(url)
This method allows the developer to subscribe all PSTN call events related to the developer company.

Parameter	Description
url	url of the developer server

Agora sends the event notification. Once the developer called unsubscribe_pstn, Agora stops sending the event notification.

Sample Code:

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

Subscribe User Online and Offline Events (subscribe_online)

subscribe_online(url)

This method allows the developer to subscribe the events upon users login or logout.

Parameter	Description
url	url of the developer server

Sample Code:

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

Agora sends the notification upon users login or logout:

- Online Notification：

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

- Offline Notification:

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

Once the developer called unsubscribe_online, Agora stops sending the online or offline notification.

Query User Status (query_online)

query_online(account)

This method allows the developer to query the user online status.

Parameter	Description
account	User ID defined by the client

Sample code:

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

Agora sends the notification：

- Offline:

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

- Online：

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

Subscribe Server Account Event (start_server)

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

The developer can subscribe events using special accounts to initiate VOIP calls and send channel messages.

Parameter	Description
account	User ID defined by the client
url	url of the developer server

The users can use the special account to initiate VOIP calls and join channels after logging onto the developer client.

Once VOIP call events or message events occur, Agora will POST the events to the developer specified URL.

Sample Code:

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

Initiate VOIP Call(voip_invite)

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

This method allows users to call a group of people to realize multi-party hybrid calls initiated.

Before initiating VOIP calls, ensure that you have subscribed server account event according to Subscribe Server Account Event (start_server).

Kargs is a string in json, parameters in json are required when initiating VOIP calls:

Parameter	Description
account	User ID defined by the client
peer	User ID of the other user
channelName	Channel name
extra	Set it as NULL.

Sample code:

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

Once a VOIP call is initiated, the users will receive a list of the following events:

- Ring:


```
{"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"}
```


- Accept：


```
{"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"}
```


- End:


```
{"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"}
```


Hybrid Call with PSTN and VOIP

If the developer has enabled SIP gateway access to Agora signaling system, and when the users of this developer initate VOIP calls with *sip* as suffix, then VOIP will turn into SIP, once delivered to the developer landline, then it is the so-called PSTN.

With PSTN calls, if the users called several other users using the developer application at the same time, then this is a hybrid call.

User SDK Initiating VOIP Call to User Server

When User A does not intiate a call, and if some other user initiates a VOIP call to the server of User A, Agora will send voip_invite notification to the developer specified URL. All subsequent events will be nofified one by one.

Send Channel Message

This method allows the user to send a broadcasting message to inform all the users in the channel.

Join Channel(channel_join)

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

Parameter	Description
account	User ID defined by the client
name	Channel name

Sample code:

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

Send Channel Message(sendmsg)

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

Parameter	Description
account	User ID defined by the client
name	Channel name
msg	Message body

Sample code:

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

The user will receive the following notification upon a successful execution:

{"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"}

Note
The user will receive the message sent by himself/herself.