WebRTC

Version: 20240528

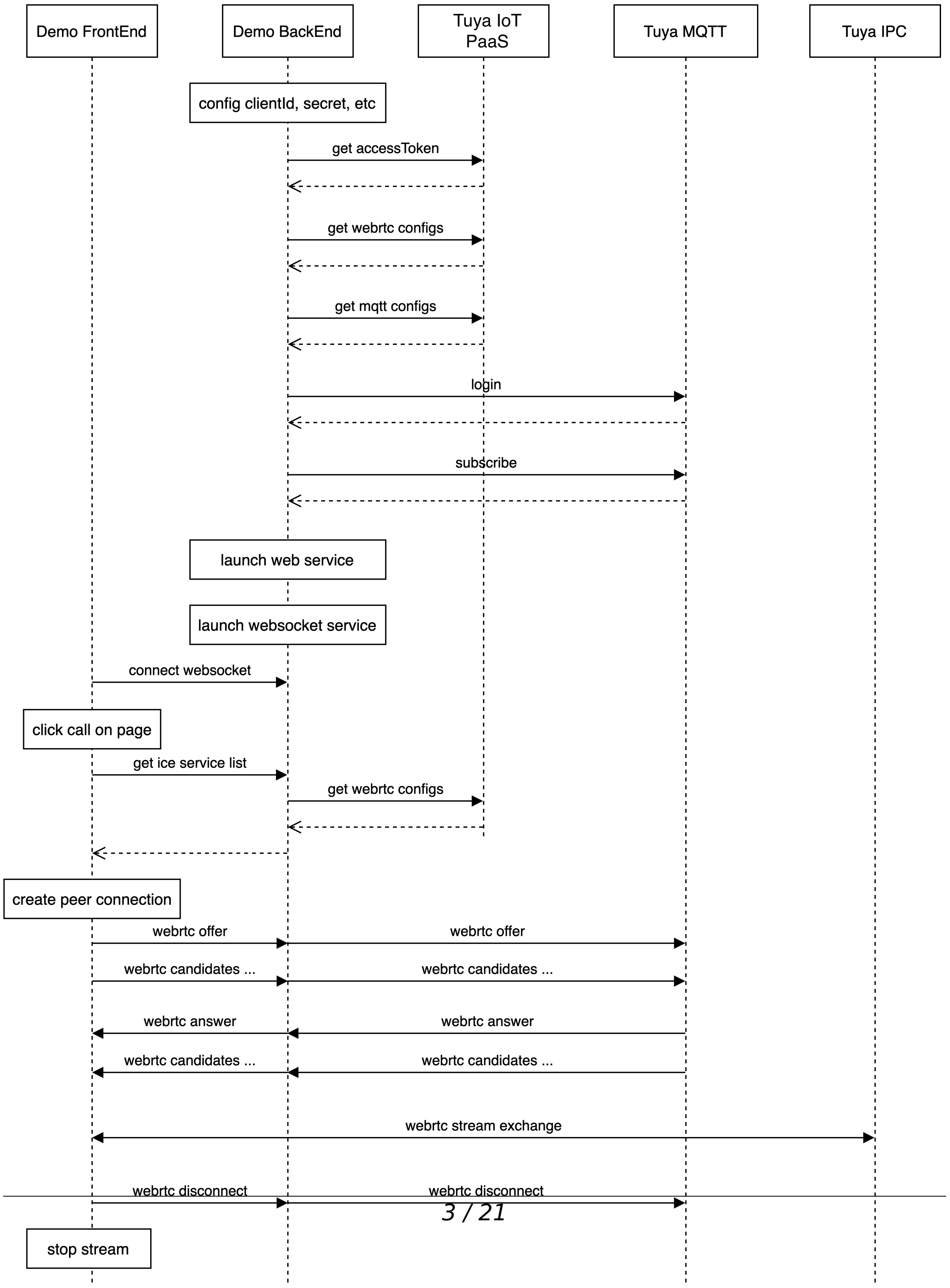
Online Version

Contents

Tuya provides access to live audio and video streams using the Web Real-Time Communication (WebRTC) protocol for IoT devices that are capable of transmitting audio and video streams. This topic describes how to integrate with live video streams, using the IP camera (IPC) as an example.

1 Interaction process

1.1 Business process



1.2 Components

- Tuya IoT Development Platform
  - Provide various HTTPS APIs for different open platforms.
- Web frontend
  - Provide a WebRTC live stream page for you to view with Google Chrome. For more information about the WebRTC protocol, see Build the backend services needed for a WebRTC app.
  - Communicate with the Web backend over WebSocket protocol.
  - Call the Javascript API to generate webRTC offers and candidates.
- Web backend
  - Host the web page.
  - Visit the Tuya IoT Development Platform and get the required configuration information over the HTTPS protocol.
  - Connect to Tuya MQTT service.
- Tuya MQTT service
  - Provide asynchronous data transmission channels.
- IP camera (IPC)

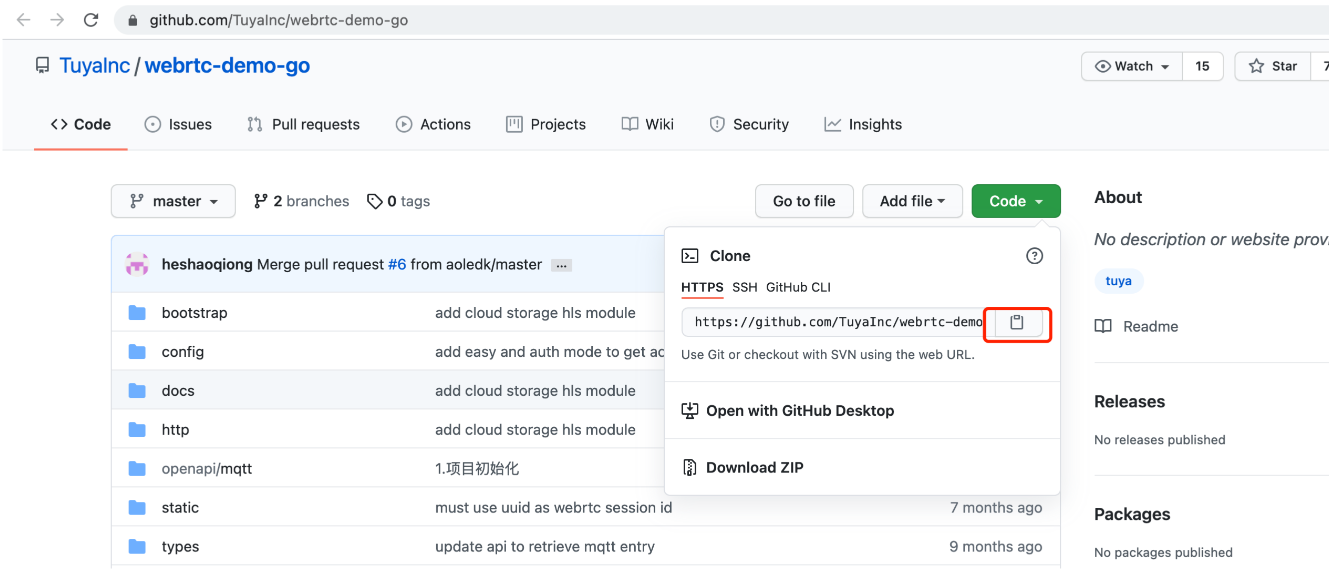
2 Connect to live audio and video streams

2.1 Prerequisites

- You have linked the IPC with the **Smart Life** app.
- You have created a cloud development project. For more information about project creation, see Create a project.
- You have linked your Smart Life app account with the project. For more information, see Link my app.

2.2 Procedure

1. Clone the webrtc-demo-go project to your local computer.



heshaoqiong Merge pull request #6 from aoledk/master

bootstrap	add cloud storage hls module	
config	add easy and auth mode to get ac	
docs	add cloud storage hls module	
http	add cloud storage hls module	
openapi/mqtt	1.项目初始化	
static	must use uuid as webrtc session id	7 months ago
types	update api to retrieve mqtt entry	9 months ago

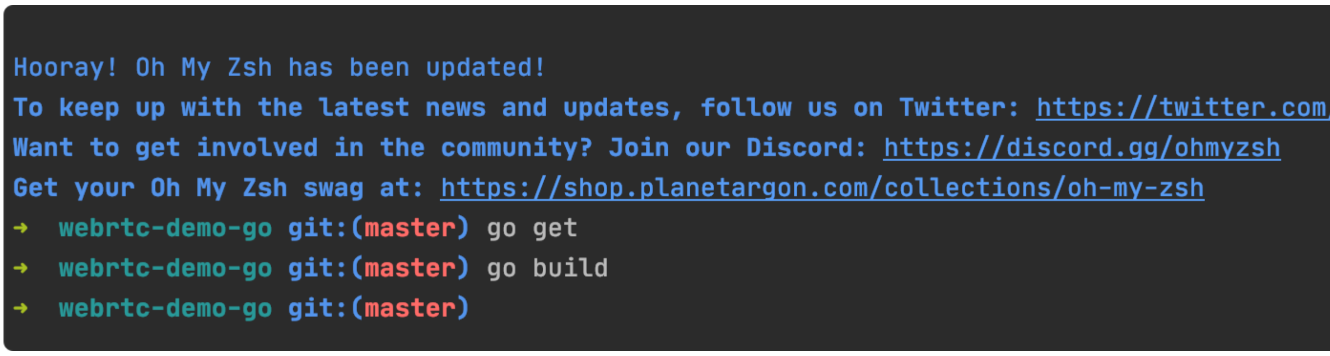
```
https://github.com/TuyaInc/webrtc-demo
```

Use Git or checkout with SVN using the web URL.

No description or website prov

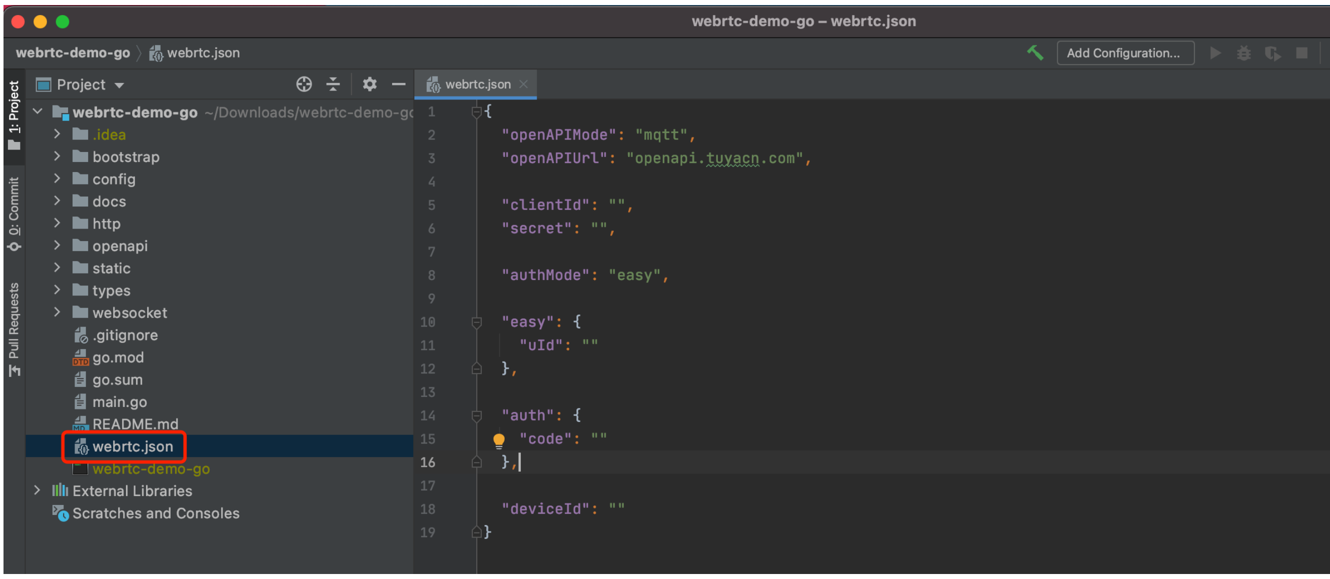
2. Run the following command in the root directory of the source code:

```
go get && go build
```



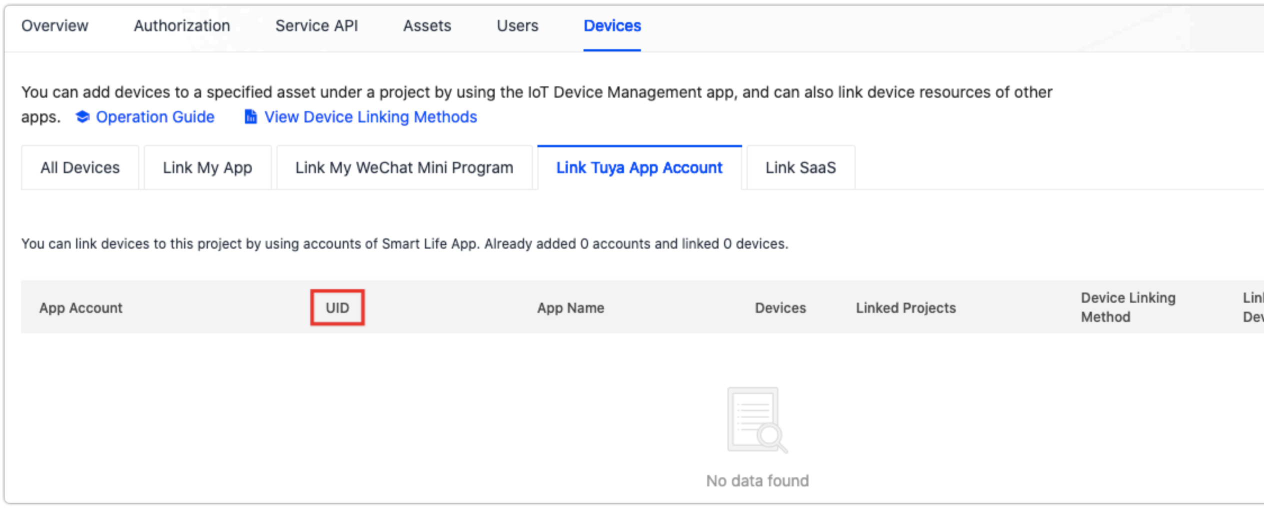
```
Hooray! Oh My Zsh has been updated!
To keep up with the latest news and updates, follow us on Twitter: https://twitter.com
Want to get involved in the community? Join our Discord: https://discord.gg/ohmyzsh
Get your Oh My Zsh swag at: https://shop.planetargon.com/collections/oh-my-zsh
➜  webrtc-demo-go git:(master) go get
➜  webrtc-demo-go git:(master) go build
➜  webrtc-demo-go git:(master)
```

3. Configure the parameters in the `webrtc.json` file.



```
{
  "openAPIMode": "mqtt",
  "openAPIUrl": "openapi.tuyacn.com",

  "clientId": "",
  "secret": "",

  "authMode": "easy",

  "easy": {
    "uId": ""
  },

  "auth": {
    "code": ""
  },

  "deviceId": ""
}
```

- `clientId`: Enter the value of **Access ID** in the **Authorization Key** section of the cloud project. For more information, see Cloud project parameters.
- `secret`: Enter the value of **Access Secret** in the **Authorization Key** section of the cloud project. For more information, see Cloud project parameters.
- `autoMode`: Select the `easy` authorization method. Enter the user ID (UID) of the linked app account in `uid`. For more information, see Device parameters.



You can add devices to a specified asset under a project by using the IoT Device Management app, and can also link device resources of other apps.

You can link devices to this project by using accounts of Smart Life App. Already added 0 accounts and linked 0 devices.

App Account	UID	App Name	Devices	Linked Projects	Device Linking Method	Lin De

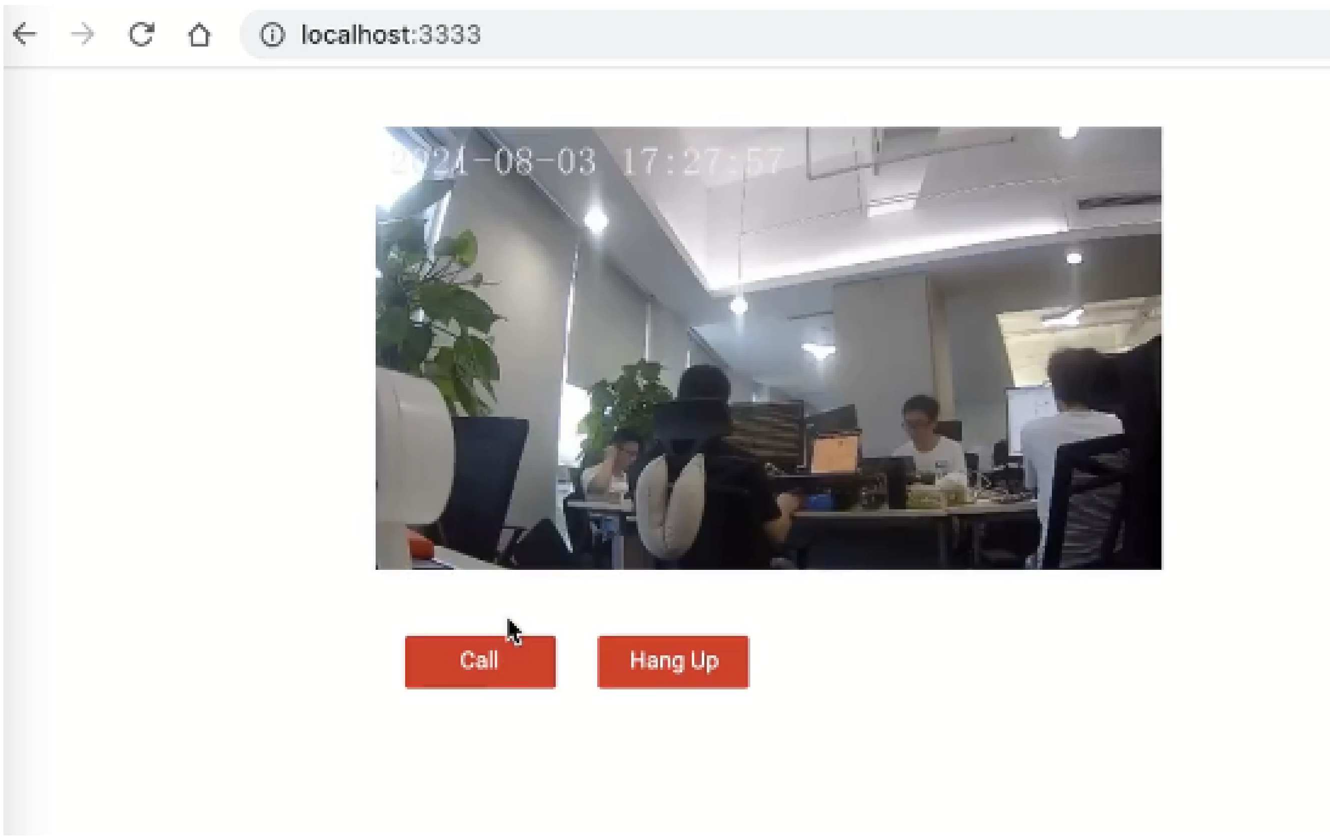
No data found

- `deviceId`: the ID of the linked device. For more information, see Device parameters.

4. Run the following command:

```
./webrtc-demo-go
```

5. Use **Google Chrome** to log in to http://localhost:3333.
6. On the web page, click **Call**.


localhost:3333
2024-08-03 17:27:57
Call
Hang Up

3 APIs

3.1 Generate MQTT configurations

API description

Generate the MQTT connection configurations for the users.

API endpoint

```
POST /v1.0/open-hub/access/config
```

Request parameter

Parameter	Data type	Location	Description	Required
link_id	String	BODY	The unique flag of the user connection with a maximum length of 8 digits.	Yes
uid	String	BODY	The user ID. It is not required in the OAuth 2.0 mode.	No
link_type	String	BODY	The connection mode. Valid value: `mqtt`.	Yes
topics	String	BODY	The MQTT topic type for WebRTC. Valid value: `ipc`.	Yes

Response parameter

Parameter	Data type	Description
result	`MQTT`	MQTT login information.
success	Boolean	The status of the response result.
t	Long	The timestamp of the response result. Unit: milliseconds.

`MQTT`

Parameter	Data type	Description
url	String	The address to be connected.
username	String	The username used for connection.
password	String	The password used for connection.
client_id	String	The `client_id` used for connection.
source_topic	String	The subscribed topic.
sink_topic	String	The posted topic.
expire_time	Integer	The expiration time.

Sample request

```
{
    "uid": "ay1564026880284v****",
    "link_id": "123456",
    "link_type": "mqtt",
    "topics": "ipc"
}
```

Sample response

```
{
    "result": {
        "client_id": "cloud_7ef68bc84629ea3f51152760cdf2****",
        "expire_time": 7200,
        "password": "0426d6917bfd8b88f037c4a598a0****",
        "sink_topic": {
            "ipc": "/av/moto/moto_id/u/{device_id}"
        },
        "source_topic": {
            "ipc": "/av/u/d09735be24f4b7eb3583b30bcaa2****"
        },
        "url": "ssl://m1-cn.wgine.com:8883",
        "username": "cloud_d09735be24f4b7eb3583b30bcaa2****"
        },
        "success": true,
        "t": 1600847208953
}
```

3.2 Get WebRTC configurations

API endpoint

```
GET /v1.0/users/{uId}/devices/{deviceId}/webrtc-configs
```

Request parameter

Parameter	Data type	Location	Description	Required
uId	String	URI	The user ID.	Yes
deviceId	String	URI	The device ID.	Yes

Response parameter

Parameter	Data type	Description
audio_attributes	AudioAttributes	The attribute of chat.
auth	String	The information about authorization.

Parameter	Data type	Description
id	String	The device ID.
moto_id	String	The ID of the connected instance.
p2p_config	`P2PConfig`	The configuration information of the connection service.
skill	String	The skill.
supports_webrtc	Boolean	Indicates whether WebRTC is supported.
vedio_clarity	Integer	The video definition.

- `AudioAttributes`

Parameter	Data type	Description
call_mode	Integer	The chat mode. Valid values:
		`1`: one-way chat.
		`2`: two-way chat.
hardware_capability	Integer	The hardware capability. Valid values:
		`1`: MIC.
		`2`: speaker.

- `P2PConfig`

Parameter	Data type	Description
ices	`Token`	The list of P2P tokens.

- `Token`

Parameter	Data type	Description
urls	String	The address of the ICE service.
username	String	The username of the ICE service.
credential	String	The password of the ICE service.
ttl	Integer	The valid duration of the ICE service. Unit: seconds.

Sample response

```
{
    "result":{
        "audio_attributes":{
            "call_mode":[
                1,
                2
            ],
            "hardware_capability":[
                1,
                2
            ]
        },
        "auth":"h85L4pljbuHFR0a/iTgViwA35xi3yTl3NyMsFQL5****",
        "id":"6cf2b6d2b09a2f8597****",
        "moto_id":"moto_cnpre002",
        "p2p_config":{
            "ices":[
                {
                    "urls":"stun:49.234.141.77:3478"
                },
                {
                    "urls":"stun:tx1stun.tuyacn.com:3478"
                },
                {
                    "urls":"nat:tx1nat.tuyacn.com:3478"
                },
                {
                    "urls":"nat:tx2nat.tuyacn.com:3478"
                },
                {
                    "credential":"kb/EA2whGCcNSM5FjXV2dxAM1MU=",
                    "ttl":36000,
                    "urls":"turn:49.234.141.77:3478",
                    "username":"1600883205:6cf2b6d2b09a2f8597****"
                },
                {
                    "credential":"kb/EA2whGCcNSM5FjXV2dxAM****",
                    "ttl":36000,
                    "urls":"turn:tx1turn.tuyacn.com:3478",
                    "username":"1600883205:6cf2b6d2b09a2f8597****"
                }
            ]
        },
        "skill":"{"webrtc":3,"audios":[{"channels":1,"dataBit":16,"c
odecType":101,"sampleRate":8000}],"videos":[{"streamType":2, "profile
Id":"","width":1920,"codecType":2,"sampleRate":90000,"height ":1080},
{"streamType":4,"width":640,"codecType":2,"height":360}]}",
        "supports_webrtc":true,
        "vedio_clarity":4
    },
    "success":true,
    "t":1600847205437
}
```

3.3 Connect to WebRTC using signaling packets

Response parameter

Parameter	Data type	Description
protocol	Integer	The protocol number of the MQTT message. WebRTC is a live-streaming service with a value of 302.
pv	String	The version of the communication protocol.
t	Integer	The Unix timestamp. Unit: seconds.
data	`Data`	The MQTT message frame.

- `Data`

Parameter	Data type	Description
header	`Header`	The MQTT message header.
msg	`Msg`	The MQTT message body. It can be `offer`, `candidate`, `answer`, and `disconnect`.

- `Header`

Parameter	Data type	Description
type	Strin	The MQTT message type. It can be `offer`, `candidate`, `answer`, and `disconnect`.
from	String	Enter the ID of the sender. For example, enter `msid` for the client, and enter `device_id` for the device.
to	Strin	Enter the ID of the receiver. For example, `device_id` or `msid`.
sub_dev_id	String	The `node_id` of a sub-device. It is only used by NVR devices.
sessi	Strin	The session ID is generated at random with a length of 32 bytes upon every connection. The signaling packets of the same connection share the same `sessionid`.
moto_id	String	You can call `/v1.0/users/{uId}/devices/{deviceId}/webrtc-configs` to get the value. If the returned result does not include the `moto_id` field, this parameter is not supported.

- `Msg`

  Currently, three MQTT message types are available: `offer`, `candidate`, and `disconnet`. The format of `msg` varies depending on MQTT message types.

  When the type is `offer`, the `Msg` format is as follows.

Parameter	Data type	Description
mode	String	The connection mode: WebRTC.
sdp	String	The WebRTC offer generated in Web.
stream_	Interge	The stream type. The default value `1` indicates a sub-stream.
auth	String	You can call `/v1.0/users/{uId}/devices/{deviceId}/webrtc-configs` to get the value of the field `auth`.

Sample response

```
{
    "protocol":302,
    "pv":"2.2",
    "t":1600820048671,
    "data":{
        "header":{
            "from":"AY1600819753305aHO5Sdj8pQMtLZ68XHMUpHKlRKJ87s",
            "to":"6c9a943f2ea6929675ymcq",
            "sessionid":"00b00036521743319b4d4c01f1705c48",
            "moto_id":"moto_5f685396jK",
            "type":"offer"
        },
        "msg":{
            "sdp":"v=0 o=- 4529163812828363188 2 IN IP4 127.0.0.1 s=
- t=0 0 a=group:BUNDLE 0 1 a=msid-semantic: WMS 1VpYoJaai0xSYjWhYxPH
qySybB3PaQ6Y3wXP m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 1
05 13 110 112 113 126 c=IN IP4 0.0.0.0 a=rtcp:9 IN IP4 0.0.0.0 a=ice
-ufrag:Q93I a=ice-pwd:P58s/ZyBRNVnuIxcrcmEmRG5 a=ice-options:trickle
 a=fingerprint:sha-256 E1:01:E0:B3:F1:97:7F:86:07:61:54:BE:42:5F:56:
E8:84:58:76:E3:E4:22:94:F1:33:2A:A3:C2:FC:67:05:3E a=setup:actpass a
=mid:0 a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level a=extm
ap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time a=ex
tmap:3 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-  e
xtensions-01 a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid a=extmap
:5 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id a=extmap:6 urn:ietf
:params:rtp-hdrext:sdes:repaired-rtp-stream-id a=sendrecv a=msid:1Vp
YoJaai0xSYjWhYxPHqySybB3PaQ6Y3wXP 1c7d25a4-9948-4165-bf4d-62fc39b8b5
28 a=rtcp-mux a=rtpmap:111 opus/48000/2 a=rtcp-fb:111 transport-cc a
=fmtp:111 minptime=10;useinbandfec=1 a=rtpmap:103 ISAC/16000 a=rtpma
p:104 ISAC/32000 a=rtpmap:9 G722/8000 a=rtpmap:0 PCMU/8000 a=rtpmap:
8 PCMA/8000 a=rtpmap:106 CN/32000 a=rtpmap:105 CN/16000 a=rtpmap:13
CN/8000 a=rtpmap:110 telephone-event/48000 a=rtpmap:112 telephone-ev
ent/32000 a=rtpmap:113 telephone-event/16000 a=rtpmap:126 telephone-
event/8000 a=ssrc:724809951 cname:7UznE7uyn6JBJ4PA a=ssrc:724809951
msid:1VpYoJaai0xSYjWhYxPHqySybB3PaQ6Y3wXP 1c7d25a4-9948-4165-bf4d-62
fc39b8b528 a=ssrc:724809951 mslabel:1VpYoJaai0xSYjWhYxPHqySybB3PaQ6Y
3wXP a=ssrc:724809951 label:1c7d25a4-9948-4165-bf4d-62fc39b8b528 m=v
ideo 9 UDP/TLS/RTP/SAVPF 96 97 98 99 100 101 122 102 120 127 119 125
 107 108 109 121 114 115 124 118 123 c=IN IP4 0.0.0.0 a=rtcp:9 IN IP
4 0.0.0.0 a=ice-ufrag:Q93I a=ice-pwd:P58s/ZyBRNVnuIxcrcmEmRG5 a=ice-
options:trickle a=fingerprint:sha-256 E1:01:E0:B3:F1:97:7F:86:07:61:
54:BE:42:5F:56:E8:84:58:76:E3:E4:22:94:F1:33:2A:A3:C2:FC:67:  05:3E a=
setup:actpass a=mid:1 a=extmap:14 urn:ietf:params:rtp-hdrext:toffset
 a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-ti
me a=extmap:13 urn:3gpp:video-orientation a=extmap:3 http://www.ietf
.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01 a=extmap:
12 http://www.webrtc.org/experiments/rtp-hdrext/playout-delay a=extm
ap:11 http://www.webrtc.org/experiments/rtp-hdrext/video-content-t  yp
e a=extmap:7 http://www.webrtc.org/experiments/rtp-hdrext/video-timi
ng a=extmap:8 http://www.webrtc.org/experiments/rtp-hdrext/color-spa
ce a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid a=extmap:5 urn:iet
f:params:rtp-hdrext:sdes:rtp-stream-id a=extmap:6 urn:ietf:params:rt
p-hdrext:sdes:repaired-rtp-stream-id a=recvonly a=rtcp-mux a=rtcp-rs
ize a=rtpmap:96 VP8/90000 a=rtcp-fb:96 goog-remb a=rtcp-fb:96 transp
ort-cc a=rtcp-fb:96 ccm fir a=rtcp-fb:96 nack a=rtcp-fb:96 nack pli
a=rtpmap:97 rtx/90000 a=fmtp:97 apt=96 a=rtpmap:98 VP9/90000 a=rtcp-
fb:98 goog-remb a=rtcp-fb:98 transport-cc a=rtcp-fb:98 ccm fir a=rtc
p-fb:98 nack a=rtcp-fb:98 nack pli a=fmtp:98 profile-id=0 a=rtpmap:9
9 rtx/90000 a=fmtp:99 apt=98 a=rtpmap:100 VP9/90000 a=rtcp-fb:100 go
og-remb a=rtcp-fb:100 transport-cc a=rtcp-fb:100 ccm fir a=rtcp-fb:1
00 nack a=rtcp-fb:100 nack pli a=fmtp:100 profile-id=2 a=rtpmap:101
rtx/90000 a=fmtp:101 apt=100 a=rtpmap:122 VP9/90000 a=rtcp-fb:122 go
og-remb a=rtcp-fb:122 transport-cc a=rtcp-fb:122 ccm fir a=rtcp-fb:1
22 nack a=rtcp-fb:122 nack pli a=fmtp:122 profile-id=1 a=rtpmap:102
H264/90000 a=rtcp-fb:102 goog-remb a=rtcp-fb:102 transport-cc a=rtcp
-fb:102 ccm fir a=rtcp-fb:102 nack a=rtcp-fb:102 nack pli a=fmtp:102
 level-asymmetry-allowed=1;packetization-mode=1;profile-level  -id=420
01f a=rtpmap:120 rtx/90000 a=fmtp:120 apt=102 a=rtpmap:127 H264/9000
0 a=rtcp-fb:127 goog-remb a=rtcp-fb:127 transport-cc a=rtcp-fb:127 c
cm fir a=rtcp-fb:127 nack a=rtcp-fb:127 nack pli a=fmtp:127 level-as
ymmetry-allowed=1;packetization-mode=0;profile-level-id=4200  1f a=rtp
map:119 rtx/90000 a=fmtp:119 apt=127 a=rtpmap:125 H264/90000 a=rtcp-
fb:125 goog-remb a=rtcp-fb:125 transport-cc a=rtcp-fb:125 ccm fir a=
rtcp-fb:125 nack a=rtcp-fb:125 nack pli a=fmtp:125 level-asymmetry-a
llowed=1;packetization-mode=1;profile-level-id=42e01f a=rtpmap:107 r
tx/90000 a=fmtp:107 apt=125 a=rtpmap:108 H264/90000 a=rtcp-fb:108 go
og-remb a=rtcp-fb:108 transport-cc a=rtcp-fb:108 ccm fir a=rtcp-fb:1
08 nack a=rtcp-fb:108 nack pli a=fmtp:108 level-asymmetry-allowed=1;
packetization-mode=0;profile-level-id=42e01f a=rtpmap:109 rtx/90000
a=fmtp:109 apt=108 a=rtpmap:121 H264/90000 a=rtcp-fb:121 goog-remb a
=rtcp-fb:121 transport-cc a=rtcp-fb:121 ccm fir a=rtcp-fb:121 nack a
=rtcp-fb:121 nack pli a=fmtp:121 level-asymmetry-allowed=1;packetiza
tion-mode=1;profile-level-id=4d0015 a=rtpmap:114 H264/90000 a=rtcp-f
b:114 goog-remb a=rtcp-fb:114 transport-cc a=rtcp-fb:114 ccm fir a=r
tcp-fb:114 nack a=rtcp-fb:114 nack pli a=fmtp:114 level-asymmetry-al
lowed=1;packetization-mode=1;profile-level-id=640015 a=rtpmap:115 rt
x/90000 a=fmtp:115 apt=114 a=rtpmap:124 red/90000 a=rtpmap:118 rtx/9
0000 a=fmtp:118 apt=124 a=rtpmap:123 ulpfec/90000 ",
            "auth":"3iHAObTiJ+P1o/OeX8My208vis9Ar6JQygHSLrBxv5U=",
            "mode":"webrtc",
            "stream_type":1
        }
    }
}
```

When the type `candidate`, the `Msg` format is as follows.

Parameter	Data type	Description
mode	String	The connection mode: WebRTC.
candidate	String	The candidate WebRTC addresses connected by both parties.

Sample response

```
{
    "protocol":302,
    "pv":"2.2",
    "t":1600820048672,
    "data":{
        "header":{
            "from":"AY1600819753305aHO5Sdj8pQMtLZ68XHMUpHKlRKJ87s",
            "to":"6c9a943f2ea6929675ymcq",
            "sessionid":"00b00036521743319b4d4c01f1705c48",
            "moto_id":"moto_5f685396jK",
            "type":"candidate"
        },
        "msg":{
            "mode":"webrtc",
            "candidate":"a=candidate:512512433 1 udp 2122260223 192.
168.0.227 50828 typ host generation 0 ufrag Q93I network-id 1"
        }
    }
}
```

When the type is `disconnect`, the `Msg` format is as follows.

Parameter	Data type	Description
mode	String	The connection mode: WebRTC.

Sample response

```
{
    "protocol":302,
    "pv":"2.2",
    "t":1600820048679,
    "data":{
        "header":{
            "from":"AY1600819753305aHO5Sdj8pQMtLZ68XHMUpHKlRKJ87s",
            "to":"6c9a943f2ea6929675ymcq",
            "sessionid":"00b00036521743319b4d4c01f1705c48",
            "moto_id":"moto_5f685396jK",
            "type":"disconnect"
        },
        "msg":{
            "mode":"webrtc"
        }
    }
}
```

4 FAQs

4.1 How to receive messages from Tuya' s MQTT service?

After you get `configs` of the open platform, you need to use the string after `/av/u/` in the JSON field of `result.source_topic.ipc` as the value of `from` in the MQTT Header. In this way, you can receive MQTT service messages as expected.