CEC
2.5.0.0.0

User Access--Voice and Video Access

Issue	01
Date	2024-03-01

HUAWEI TECHNOLOGIES CO., LTD.

Huawei Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China
Website:	https://www.huawei.com
Email:	support@huawei.com

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Preparations Before Development

Contact Huawei engineers to obtain **app_key**, **app_secret**, and ID of the channel to be accessed.

2 Sign-In Authentication

2.1 Generating a Token Using the API Fabric

Scenario

The API Fabric generates a token.

URL: https://*Domain Address*/apigovernance/api/oauth/tokenByAkSk

NOTE

Replace *Domain Address* with the actual address or domain name of the AICC.

For example, in the Huawei public cloud production environment, replace *Domain Address* with **service.besclouds.com**. The invoking URL is **https://service.besclouds.com/apigovernance/api/oauth/tokenByAkSk**.

Request Header

{

Content-Type: application/json

X-Token-Expire:600

}

NOTE

X-Token-Expire indicates the token expiration time, which is set to **600**.

Request Parameters

{

"app_key": "**xxxxxxxxxxxxxxxx**",

"app_secret": "**yyyyyyyyyyyyyyyyyy**"

}

NOTE

app_key indicates the app ID, and **app_secret** indicates the key. The two values are fixed.

Response Parameters

{

"AccessToken": "**zzzzzzzzzzzzzzzzzz**",

"ApplyType": "Bearer",

"CreateTime": "1545650171",

"Expires": "600",

"Scope": "46fb027c8b4f1541e459cadea096495a",

"AppKey": "xxxxxxxxxxxxxxxxxx",

"UserID": "Anonymous"

}

NOTE

AccessToken indicates the token of the API.

2.2 Obtaining the Token of CC-Messaging

Interface Function

The ApplyToKen interface is invoked to obtain the token of CC-Messaging.

Request Method

This interface supports only the POST method, and does not support the PUT, GET, or DELETE method.

Request URL

https://*API Fabric domain name*/apigovernance/api/oauth/tokenByAkSk

Request Message

- Message header

  x-app-key: **app_key** in **1 Preparations Before Development**

  Authorization: **Bearer** + **AccessToken** obtained in **2.1 Generating a Token Using the API Fabric**

  Content-Type:application/json; charset=UTF-8

- Message body

  The following provides an example of the request message body of this interface:

```
{   "userId":"xxx",
    "userName":"xxx",
    "channelId":"xxx",
    "locale":"zh",
}
```

  Table 2-1 describes the parameters in the request message body of this interface.

Table 2-1 Parameters in the message body

Parameter	Type	Mandatory	Description
userId	String	Yes	ID of the user who accesses the channel.
userName	String	Yes	User name for accessing the channel.
channelId	String	Yes	ID of the channel to be accessed.
locale	String	Yes	Language type.

Response Message

The following provides an example of the response message body of this interface:

```
{   "resultCode":"0",
   "token":"xxx"
}
```

Table 2-2 describes the parameters in the response message body of this interface.

Table 2-2 Parameters in the message body

Parameter	Type	Description
resultCode	String	Request result.
token	String	A token.

Troubleshooting

The following is the common returned error information (returned when **http status** is not **200**):

```
{     "errorCode":"0",
    "exceptionInfo":"xxx"
}
```

Find the cause based on the description in **exceptionInfo**. For example, if error code 403 and the following **exceptionInfo** are returned, the token applied in the previous section has expired:

"auth fail! please apply or refresh the access token from the server!"

3 Interface Description

For details, see the description of the getClickToCallEvents, checkClickToCallSupport, dropClickToCall and doLeaveMessage interfaces in ***Interface Reference***.

4 Interface Development Process

4.1 Preparing the Common Header Fields

NOTICE

This header field is required when all click-to-dial interfaces are invoked.

Header:

x-app-key: **app_key** in **1 Preparations Before Development**

Authorization: **Bearer** + **AccessToken** obtained in **2.1 Generating a Token Using the API Fabric**

ccmessaging-token: token obtained in **2.2 Obtaining the Token of CC-Messaging**

Content-Type: application/json

4.2 Interface Invoking Sequence

Interface Invoking Sequence for Making a Click-to-Dial Voice Call

1. Invoke the checkClickToCallSupport interface to check whether the channel supports the click-to-dial function.
2. Invoke the createClickToCall interface to create a click-to-dial call.
3. Invoke the getClickToCallEvents interface in the case of long polling after a click-to-dial call is created to obtain the click-to-dial call event.
4. Invoke the dropClickToCall interface to hang up the call on the subscriber side and release the click-to-dial call.

Interface Invoking Sequence for Making a Collaborative Click-to-Dial Call (Multimedia Text and then Click-to-Dial Call)

1. Invoke the send interface to access a multimedia text session.
2. Invoke the checkClickToCallSupport interface to check whether the channel supports the click-to-dial function.
3. Invoke the createClickToCall interface to create a click-to-dial call.
4. Invoke the getClickToCallEvents interface in the case of long polling after a click-to-dial call is created to obtain the click-to-dial call event.
5. Invoke the dropClickToCall interface to hang up the call on the subscriber side and release the click-to-dial call.

Interface Invoking Sequence for Making a Collaborative Click-to-Dial Call (Click-to-Dial Call and then Multimedia Text)

1. Invoke the checkClickToCallSupport interface to check whether the channel supports the click-to-dial function.
2. Invoke the createClickToCall interface to create a click-to-dial call.
3. Invoke the getClickToCallEvents interface in the case of long polling after a click-to-dial call is created to obtain the click-to-dial call event.
4. Invoke the send interface to access a multimedia text session.
5. Invoke the dropClickToCall interface to hang up the call on the subscriber side and release the click-to-dial call.

5 Code Example

The following uses the interface for making a collaborative click-to-dial call (multimedia text and then click-to-dial call) as an example.

Procedure

Step 1 Request the send interface.

```
this.$axios({
    method: 'post',
    url: API Fabric domain name/apiaccess/ccmessaging/send,
    headers: {
       'Content-Type': 'application/json',
       'x-app-key': c.appKey,
       'Authorization': fabric.token,
       'ccmessaging-token': ccmessaging.token
    },
    data: {
       'channel': 'WEB',
       'content': 'start',
      'controlType': 'CONNECT',
       'from': userId,
       'mediaType': 'TEXT',
       'sourceType': 'CUSTOMER',
      'to': channelId
    }
})
```

If the **http status** value returned by the send interface is **200** and the **resultCode** value in the returned message body is **0**, the request is successful.

In this case, the agent can view the connected subscriber on the online chat workbench.

Step 2 Request the checkClickToCallSupport interface.

Before sending the request, ensure that:

- The send interface has received a response indicating that the access is successful.
- The browser supports WebRTC. (For details about how to check whether the browser supports WebRTC, see the WebRTC official documentation.)

Check example:

```
if (!navigator.mediaDevices || !navigator.mediaDevices.getUserMedia) {
      return Promise.reject(new Error('WebRTC is not supported'))
}
let cam = false
let mic = false
let spkr = false
return navigator.mediaDevices.enumerateDevices().then((deviceInfos) => {
   deviceInfos.forEach(function (d) {
       switch (d.kind) {
            case 'videoinput':
             cam = true
             break
           case 'audioinput':
              mic = true
              break
            case 'audiooutput':
              spkr = true
              break
         }
  })
  // Chrome supports 'audiooutput', Firefox and Safari do not support.
  if (navigator.webkitGetUserMedia === undefined) {
    spkr = true
  }
  if (!spkr) {
    return Promise.reject(new Error('Missing a speaker! Please connect one and reload'))
  }
  if (!mic) {
    return Promise.reject(new Error('Missing a microphone! Please connect one and reload'))
  }
  return Promise.resolve(cam)
})
```

1. After the preceding checks are successful, invoke the checkClickToCallSupport interface.

   ```
   this.$axios({
       method: 'get',
      url: API Fabric domain name/apiaccess/ccmessaging/v1/checkClickToCallSupport?channel=WEB,
       headers: {
          'Content-Type': 'application/json',
          'x-app-key': appKey,
          'Authorization': fabric.token,
          'ccmessaging-token': ccmessaging.token
       }
   })
   ```

2. The returned message body is as follows:

   ```
   {    "resultCode":"0",
       "resultDesc": "",
       "webRTCSupported":true,
       "clickToCallSupported":true
   }
   ```

   If the value of **httpStatus** is **200** and the value of **resultCode** is **0**, the request is successful.

   webRTCSupported indicates whether the tenant space supports WebRTC. **clickToCallSupported** indicates whether the channel supports the click-to-dial function.

   If the values of the preceding two variables are **true**, you can go to the next step to create a click-to-dial call.

Step 3 Request the createClickToCall interface.

NOTICE

Ensure that the values of **webRTCSupported** and **clickToCallSupported** returned by the checkClickToCallSupport interface are **true**.

In request parameters, the values of **mediaAbility** are described as follows: **0** indicates a voice call, and **1** indicates a video call.

```
this.$axios({
   method: 'post',
   url: API Fabric domain name/apiaccess/ccmessaging/v1/createClickToCall,
   headers: {
      'Content-Type': 'application/json',
      'x-app-key': appKey,
      'Authorization': fabric.token,
      'ccmessaging-token': ccmessaging.token
   },
   data: {
      'channel': 'WEB',
      'mediaAbility': '0'
   }
})
```

The returned message body is as follows:

```
{    "resultCode":"0",
   "resultDesc": ""
}
```

If the value of **httpStatus** is **200** and the value of **resultCode** is **0**, the request is successful.

Step 4 Request the getClickToCallEvents interface in the case of long polling.

After the createClickToCall interface is successfully invoked, the getClickToCallEvents interface is invoked.

NOTE

1. Set the timeout interval of the request to a longer value. Requests are processed slowly, usually for more than 10 seconds. For example, the value in the preceding request is set to 60 seconds.
2. The request is a long polling request. After the request is successful, the request is invoked based on the returned event status.

```
this.$axios({
   method: 'get',
   url: API Fabric domain name/apiaccess/ccmessaging/v1/getClickToCallEvents?channel=WEB,
   timeout: 60000,
   headers: {
      'Content-Type': 'application/json',
      'x-app-key': appKey,
      'Authorization': fabric.token,
      'ccmessaging-token': ccmessaging.token
   }
})
```

The command output is as follows:

If the value of **resultCode** is not **0**, the request fails. In this case, set a retry mechanism. For example, if the client fails to send the request for three consecutive times, the client stops sending the request again.

When the value of **resultCode** is **0**, the values of **eventId** are described as follows:

168101: call setup. **168102**: call queuing. **168106**: call forwarding. During polling, there may be no event, and **eventId** is not returned. In this case, the client needs to maintain long polling.

When a call is set up, the following mandatory WebRTC parameters can be obtained in **content**:

domain indicates the WebRTC gateway domain name. **gwAddresses** indicates the communication address and port of the WebRTC gateway. **clickToCallCaller** is the calling party, and **accessCode** is the called party.

- Call setup
```
{    resultCode:"0",
    resultDesc:"Call connected",
    "eventId": 168101,
   "content":{
        "domain":"xxx"
        "gwAddresses":["xx1","xx2"]
        "accessCode": "179080000537636"
        "clickToCallCaller":"AnonymousCard"
    }
}
```
- Call queuing
```
{    resultCode:"0"
    resultDesc:"Call in queue"
    "eventId": 168102
}
```
- Call transfer
```
{
    resultCode:"0"
    resultDesc:"Call transfered"
    "eventId": 168106
}
```
- Call release
```
{
    resultCode:"0"
    resultDesc:"Call disconnected"
    "eventId": 168110
   "content":{
        "causeId":  -1
        "causeDesc": "xxxx"
    }
}
```
- Call queuing timeout
```
{
    resultCode:"0"
    resultDesc:"Call queue timeout"
    "eventId": 168103
   "content":{
        "causeId":  -1
        "causeDesc": "xxxx"
    }
}
```
- Call failure
```
{
    resultCode:"0"
    resultDesc:"Call failed"
    "eventId": 168105
   "content":{
        "causeId":  -1
        "causeDesc": "xxxx"
    }
}
```

- No events obtained

```
{
    resultCode:"0"
    resultDesc:"ClickToCall polled without any events"
}
```

Step 5 Request the dropClickToCall interface to hang up the call.

```
this.$axios({
    method: 'post',
    url: API Fabric domain name/apiaccess/ccmessaging/v1/dropClickToCall,
    headers: {
        'Content-Type': 'application/json',
        'x-app-key': appKey,
        'Authorization': fabric.token,
        'ccmessaging-token': ccmessaging.token
    },
    data: {
        'channel': 'WEB'
    }
})
```

The returned message body is as follows:

If the value of **httpStatus** is **200** and the value of **resultCode** is **0**, the request is successful.

```
{
    "resultCode":"0",
    "resultDesc": ""
}
```

----End

6 FAQ

6.1 Token Expired

During long polling, subsequent requests may fail due to token expiration. In this case, design the retry logic as follows:

Re-initiate the two interfaces in **2 Sign-In Authentication**, update the values of **Authorization** and **ccmessaging-token** in the common header field, and initiate the request again.

When a click-to-dial request times out, the value of **httpStatus** of the interface for requesting the click-to-dial service is **403**.