

CEC
2.5.0.0.0

用户接入——VOIP 音视频接入

文档版本 01
发布日期 2024-03-01



版权所有 © 华为技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <https://www.huawei.com>

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：

<https://www.huawei.com/cn/psirt/vul-response-process>

如企业客户须获取漏洞信息，请参见如下网址：

<https://securitybulletin.huawei.com/enterprise/cn/security-advisory>

目录

1 开发前准备	1
2 登录鉴权	2
2.1 C1 系统配置类接口鉴权方式.....	2
2.2 获取 CC-Messaging 的 Token.....	3
3 接口说明	5
4 接口开发流程	6
4.1 准备公共头域.....	6
4.2 调用接口顺序.....	6
5 代码使用示例	8
6 FAQ	13
6.1 Token 过期.....	13

1 开发前准备

请联系华为工程师获取app_key， app_secret和需要接入的渠道ID。

2 登录鉴权

[2.1 C1 系统配置类接口鉴权方式](#)

[2.2 获取CC-Messaging的Token](#)

2.1 C1 系统配置类接口鉴权方式

场景描述

Api Fabric生成token

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

说明

Domain Address请根据CEC实际的地址或域名进行替换。

例如，在华为公有云生产环境，请将Domain Address替换为service.besclouds.com，则调用URL为`https://service.besclouds.com/apigovernance/api/oauth/tokenByAkSk`

请求头

```
{  
  Content-Type: application/json  
  X-Token-Expire:600  
}
```

说明

X-Token-Expire为token超期时间，设置为600s，最大可设置为3600s。

请求参数

```
{  
  "app_key": "xxxxxxxxxxxxxxxxxxxx",  
  "app_secret": "yyyyyyyyyyyyyyyyyyyy"
```



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

该接口的请求消息体参数说明如表2-1所示。

表 2-1 消息体参数说明

参数名	数据类型	选取原则	说明
userId	String	必选	接入该渠道的用户Id
userName	String	必选	接入该渠道的用户名
channelId	String	必选	需要接入的渠道Id
locale	String	必选	语言种类

响应消息

该接口的响应消息体举例如下：

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

该接口的响应消息体参数说明如表2-2所示。

表 2-2 消息体参数说明

参数名	数据类型	说明
resultCode	String	请求结果。
token	String	token。

异常处理

常见错误返回（HTTP status 非200返回）

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

请根据exceptionInfo中返回的描述寻求原因：如出现 403返回，"exceptionInfo": "auth fail! illegal or expired token info. please apply or refresh the access token from the server!"，可能是上一步申请的AccessToken过期导致。

3 接口说明

请参见接口参考中的getClickToCallEvents、checkClickToCallSupport、dropClickToCall、doLeaveMessage接口。

4 接口开发流程

4.1 准备公共头域

4.2 调用接口顺序

4.1 准备公共头域

须知

在后续调用点击通话所有接口，都需要使用此头域。

Header :

x-app-key: 准备工作中的app_key

Authorization: 'Bearer ' + [2.1 C1 系统配置类接口鉴权方式](#)中的AccessToken

ccmessaging-token: [2.2 获取CC-Messaging的Token](#)中的token

Content-Type: application/json

4.2 调用接口顺序

单独点击通话使用接口顺序

1. 调用checkClickToCallSupport接口，检查该渠道是否支持点击通话
2. 调用createClickToCall接口，创建点击通话
3. 长轮询调用getClickToCallEvents接口，在创建完点击通话后调用该接口，获取点击通话事件
4. 调用dropClickToCall接口进行用户侧主动挂断，释放点击通话

协同点击通话使用接口顺序（先多媒体文本后点击通话）

1. 调用send接口，接入到多媒体文本会话中

2. 调用checkClickToCallSupport接口，检查该渠道是否支持点击通话
3. 调用createClickToCall接口，创建点击通话
4. 长轮询调用getClickToCallEvents接口，在创建完点击通话后调用该接口，获取点击通话事件
5. 调用dropClickToCall接口进行用户侧主动挂断，释放点击通话

协同点击通话使用接口顺序（先点击通话后多媒体文本）

1. 调用checkClickToCallSupport接口，检查该渠道是否支持点击通话
2. 调用createClickToCall接口，创建点击通话
3. 长轮询调用getClickToCallEvents接口，在创建完点击通话后调用该接口，获取点击通话事件
4. 调用send接口，接入到多媒体文本会话中
5. 调用dropClickToCall接口进行用户侧主动挂断，释放点击通话

5 代码使用示例

以协同点击通话（先多媒体文本后点击通话）接口使用代码示例。

操作步骤

步骤1 请求send接口。

```
this.$axios({
  method: 'post',
  url: APIfabric域名/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
  }
})
```

根据send接口返回的http status为200，且返回消息体resultCode为0，则成功。

此时座席侧在线交谈工作台已可以看到接入的用户。

步骤2 请求checkClickToCallSupport接口

请求之前，请确保以下两点：

- send接口得到接入成功的返回
- 检查浏览器环境支持WebRTC（请结合WebRTC官方文档查看具体如何检查）

检查示例：

```
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. 在上述检查均通过后，可开始调用checkClickToCallSupport接口。

```

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

```

2. 返回消息体：

```

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

```

如httpStatus为200，且resultCode为0，代表请求成功。

webRTCSupported，代表该租间是否支持webRTC；clickToCallSupported代表该渠道是否支持点击通话。

上述两个变量均为true时，可进行下一步，创建点击通话。

步骤3 请求createClickToCall。

须知

请确保checkClickToCallSupport返回的webRTCSupported和clickToCallSupported为true，才可调用。

请求参数中的mediaAbility为0，为发起语音通话；为1，为发起视频通话。

```

this.$axios({
  method: 'post',
  url: APIfabric域名/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'
  }
})

```

返回消息体：

```

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

```

如httpStatus 为200，且resultCode为0，代表请求成功。

步骤4 长轮询请求getClickToCallEvents。

在createClickToCall调用成功后，开始调用getClickToCallEvents接口。

说明

1. 请设置该请求的超时时间为一个较长值；调用该接口请求，处理较缓慢，经常在10秒以上；如上述请求中设置的值为60秒。
2. 该请求为长轮询请求，在请求success后，根据返回事件状态，继续调用该请求。

```

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

```

返回结果如下：

resultCode为非0时，表示请求失败，请设置重试机制：如客户端请求失败，重试3次后，仍然失败则停止请求重试。

resultCode为0，eventId为：

168101：呼叫建立；168102：呼叫排队；168106：呼叫转移；或者轮询时，可能没有事件，eventId是不携带时的情况，客户端需要保持长轮询。

当呼叫建立时，可从content中获取 webRTC必要参数：

domain为WebRTC Gateway域名；gwAddresses为WebRTC Gateway通信地址和端口；clickToCallCaller为主叫，accessCode为被叫

- 呼叫建立

```

{  resultCode:"0",
  resultDesc:"Call connected",
  "eventId": 168101,
  "content": {
    "domain":"xxx"
    "gwAddresses":["xx1","xx2"]
    "accessCode": "179080000537636"
    "clickToCallCaller":"AnonymousCard"
  }
}

```

- 呼叫排队

```

{  resultCode:"0"
  resultDesc:"Call in queue"
}

```

- ```
"eventId": 168102
}
```
- 呼叫转移

```
{
 resultCode:"0"
 resultDesc:"Call transfered"
 "eventId": 168106
}
```
  - 呼叫释放

```
{
 resultCode:"0"
 resultDesc:"Call disconnected"
 "eventId": 168110
 "content": {
 "causeId": -1
 "causeDesc": "xxxx"
 }
}
```
  - 呼叫排队超时

```
{
 resultCode:"0"
 resultDesc:"Call queue timeout"
 "eventId": 168103
 "content": {
 "causeId": -1
 "causeDesc": "xxxx"
 }
}
```
  - 呼叫失败

```
{
 resultCode:"0"
 resultDesc:"Call failed"
 "eventId": 168105
 "content": {
 "causeId": -1
 "causeDesc": "xxxx"
 }
}
```
  - 没有获取到事件

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

**步骤5** 主动挂断时，调用dropClickToCall接口。

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

返回消息体：

如httpStatus为200，且resultCode为0，代表请求成功；

```
{
 "resultCode":"0",
```

```
"resultDesc": ""
}
```

----结束

# 6 FAQ

---

## 6.1 Token过期

### 6.1 Token 过期

在长轮询请求过程中，可能会因为token过期而导致后续请求失败，此时请设计重试逻辑：

重新发起[2 登录鉴权](#)中的两个接口，更新公共请求头域中的Authorization，ccmessaging-token请求头内容，再次发起请求；

在超时情况时，当前请求点击通话的接口的HttpStatus状态为403。