CEC
2.5.0.0.0

# User Access--Web Lightweight Client Integration (JS)

| | |
|---|---|
| **Issue** | 01 |
| **Date** | 2024-03-01 |

HUAWEI TECHNOLOGIES CO., LTD.

## Trademarks and Permissions

HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

## Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.


# 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 Lightweight Web Chat Control Integration (Token Mode)



## 1.1 Overview

The lightweight web chat control is a customer-side online customer service product that can be quickly integrated into websites. **Currently, only web integration on PCs is supported.** You can use our product to quickly build an intelligent online customer service system in the CEC SaaS (Software as a Service) system to chat with customers online on web pages. **Figure 1-1** and **Figure 1-2** show the pages of our product.

The lightweight web chat control provides the function of chatting with customers online on web pages. The features are as follows:

- It is lightweight, easy to be integrated into your web pages, and does not occupy the main pages of portals and workbenches.
- It is easy to operate. Customers can easily find the customer service window and chat with online agents on the page.

**NOTE**

Before using this document, you need to learn about the following two customer authentication modes:

1. Token mode: Used in scenarios where access customers need to be authenticated.
2. Authorization mode: Guest mode that is used in scenarios where access customers are not authenticated.

Select the development guide based on your requirements. If you want to use the authorization mode, go to **2 Lightweight Web Chat Control Integration (Authorization Mode)**.

**Figure 1-1** Online customer service window on the customer side

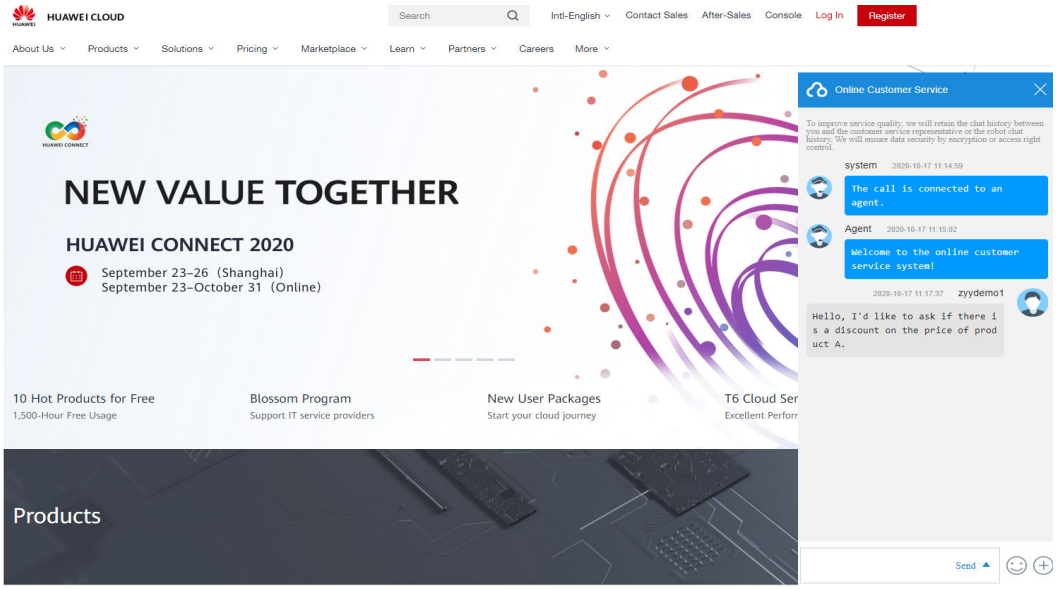
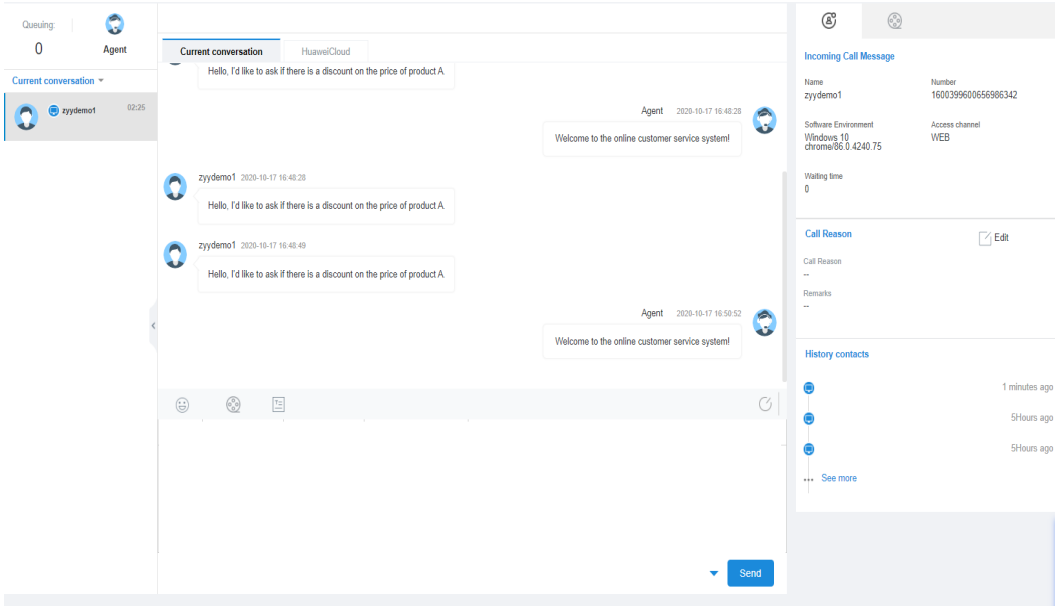


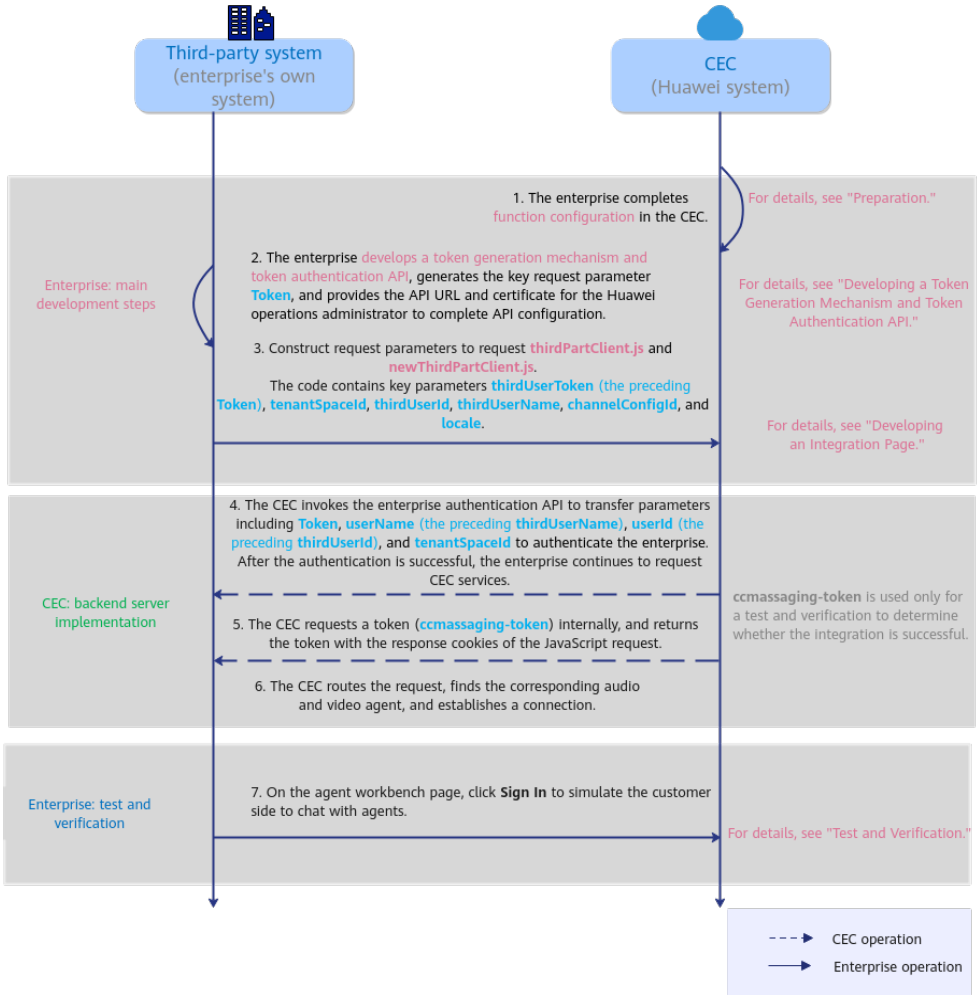

**Figure 1-2** Online customer service page on the agent side



## 1.2 Integration Principle

**Figure 1-3** shows the integration principle.

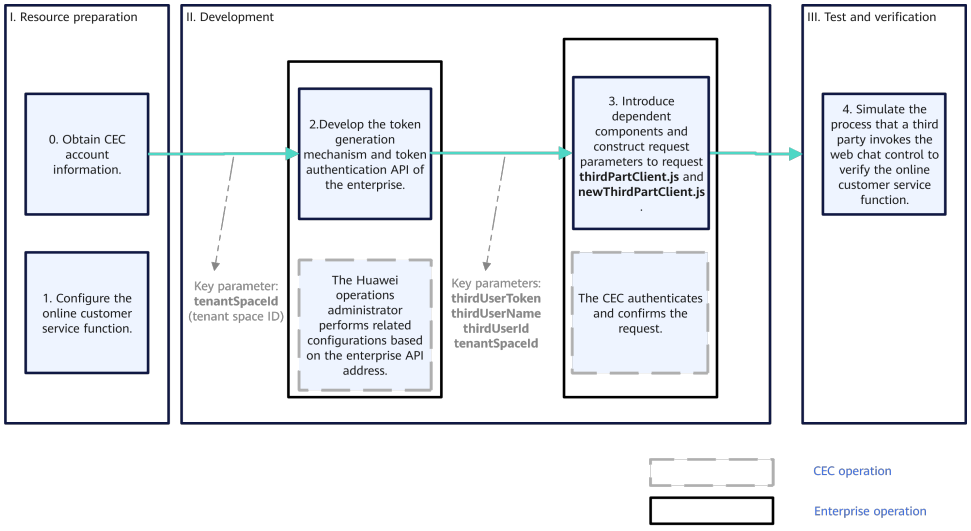
**Figure 1-3** Integration principle of the web chat control in token mode




## 1.3 Integration Procedure

The lightweight web chat control can be quickly and efficiently integrated into your pages. You can perform integration development by referring to **Figure 1-4**.

**Figure 1-4** Integration development procedure




**Step 1** Before the development, prepare resources, obtain sign-in information, and configure the online customer service function. For details, see **1.4 Preparation**.

**Step 2** Develop the token generation mechanism and token authentication API of the enterprise, and provide them for the CEC. After the authentication is successful, the CEC returns the token to the third-party system. The third-party system checks whether the request is sent by the current system. For details, see **1.5.1 Developing a Token Generation Mechanism and Token Authentication API**.

**NOTICE**

Huawei uses HTTPS to ensure the security of information transmission channels. Third parties must ensure that the developed authentication functions have security protection capabilities, including password complexity verification, anti-brute force cracking, and anti-DoS attack.

**Step 3** Introduce related dependent frameworks, construct request parameters to request **thirdPartyClient.js** and **newThirdPartyClient.js**, and return the token, tenant ID, enterprise user ID and username, and other information to the CEC for authentication. After the authentication is successful, the lightweight access of the CEC web page chat capability can be implemented. For details, see **1.5.2 Developing an Integration Page**.

**Step 4** Test and verify the functions of the web chat control. Initiate an online chat in the customer-side window to check the function of chatting with agents. For details, see **1.6 Test and Verification**.

**----End**

# 1.4 Preparation

This section describes the preparations that need to be completed in the CEC before integration development.

## 1.4.1 Preparing Resources

Before the integration, you need to complete the following preparations:

1. You have applied for tenant information from the CEC. The system O&M administrator has added tenant information for you and provided the following information for you.

   **Table 1-1** Parameters

   | Parameter | Description |
   | --- | --- |
   | TenantId | Tenant space ID generated by the system after the tenant space (that is, your CEC) is successfully created.<br>To obtain the tenant space ID, sign in to your tenant space and choose **Configuration Center** > **System Management** > **Tenant Information**. |
   | Account | Account for signing in to the CEC. |
   | Password | Password for signing in to the CEC. |

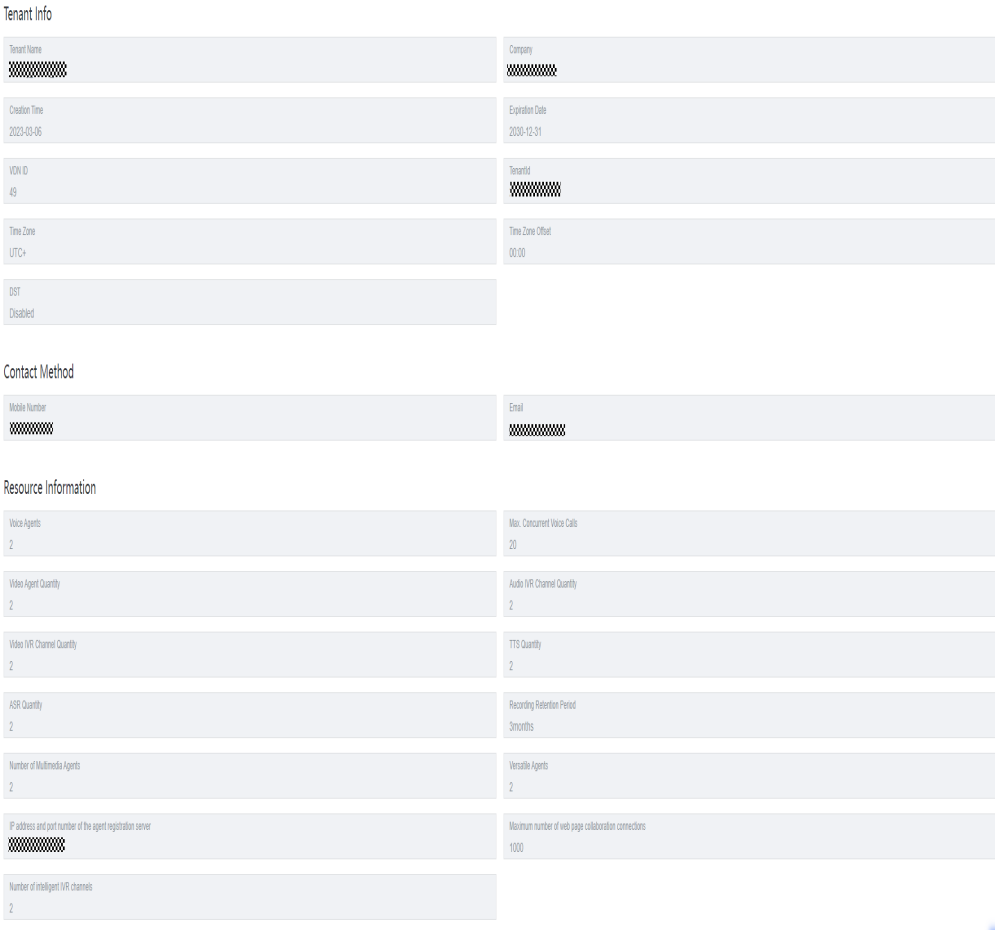
2. Use the tenant administrator account and password to sign in to your CEC and change the initial password on the sign-in page.
3. Perform the following steps to ensure that your tenant space has the multimedia agent feature:

   Sign in to the tenant space and choose **Configuration Center** > **System Management** > **Tenant Information**.

   Check the number of multimedia agents. If the number is 0, the multimedia agent feature is not enabled. Contact the Huawei operations administrator to enable the feature.

**Figure 1-5** Checking the multimedia agent feature




4. (Optional) To enable the click-to-dial function (a customer can make a voice or video call to an agent during an online text chat), contact the Huawei operations administrator.
5. (Optional) To enable the co-browsing function (a customer can share the web page with an agent or mark content on the web page during an online text chat), contact the Huawei operations administrator.
6. Obtain the JavaScript file address provided by the CEC, as shown in the following information. Replace the domain name with the actual one of the CEC.

   https://*servicestage.besclouds.com*/service-cloud/webclient/chat_client/js/thirdPartyClient.js?&t=*1595993533588*

## 1.4.2 Enabling the Online Customer Service Function in the CEC

**Step 1** Sign in to the CEC as a tenant administrator.

**Step 2** Add a multimedia skill queue.

Choose **Configuration Center** > **Employee Center** > **Skill Queue**. **Figure 1-6** shows the page for configuring a multimedia skill queue. If you need to enable the click-to-dial function, also configure a click-to-dial skill queue. **Figure 1-7** shows the page for configuring a click-to-dial skill queue.

**Step 3** Click **New** and set parameters based on **Table 1-2**.

**Figure 1-6** Page for configuring a multimedia skill queue

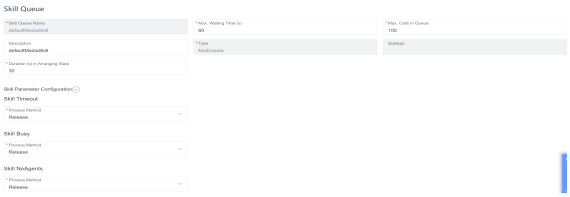



**Figure 1-7** (Optional) Page for configuring a click-to-dial skill queue

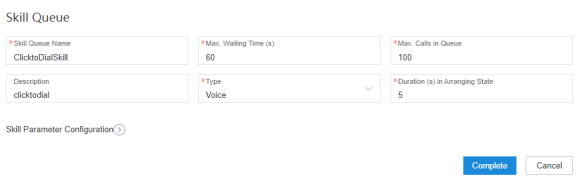



**Table 1-2** Parameters for configuring a skill queue

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Skill Queue Name | Yes | The value can contain a maximum of 20 characters and cannot contain spaces. |
| Max. Waiting Time (s) | Yes | The default value is **60**.<br>The unit is second.<br>The value ranges from 1 to 60000. |
| Max. Calls in Queue | Yes | The default value is **100**.<br>The value ranges from 1 to 10000. |
| Description | Yes | The value can contain a maximum of 50 characters. |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Type | Yes | The options are as follows:<br>● **Voice**: A voice skill queue handles voice businesses.<br>● **Multimedia**: A multimedia skill queue handles multimedia businesses.<br>● **Video**: A video skill queue handles video businesses.<br>● **Voice Click to Dial**: A voice click-to-dial skill queue is used together with multimedia businesses. During a text chat with an agent, a customer can directly make a voice call to the agent.<br>● **Video Click to Dial**: A video click-to-dial skill queue is used together with multimedia businesses. During a text chat with an agent, a customer can directly make a video call to the agent.<br>**NOTE**<br>Click-to-dial skill queues can be used only in the web channel.<br>The default value is **Voice**. |
| Duration (s) in Arranging State | Yes | Duration during which an agent is in wrap-up state after a call ends. The default value is **5**.<br>After this duration, the agent enters the idle state and can answer calls from customers.<br>The value ranges from 0 to 3600. |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Skill Parameter Configuration | No | Personalized configurations, which are the handling policies when a customer calls a skill queue and the call cannot be connected. The parameters are as follows:<br>● **Skill Timeout**<br>– **Process Method**: Handling policy when queuing times out because no idle agent can answer the call.<br>▪ **Release** (default)<br>▪ **Transfer**<br>– **Device Type**: If **Process Method** is set to **Transfer**, you need to configure the skill queue or IVR flow to which the call is transferred.<br>▪ **Skill Queue**<br>▪ **IVR**<br>● **Skill Busy**<br>– **Process Method**: Handling policy when the customer is queuing because no idle agent can answer the call, or the number of queuing customers exceeds the upper limit.<br>▪ **Release** (default)<br>▪ **Transfer**<br>– **Device Type**: If **Process Method** is set to **Transfer**, you need to configure the skill queue or IVR flow to which the call is transferred.<br>▪ **Skill Queue**<br>▪ **IVR**<br>● **Skill NoAgents**<br>– **Process Method**: Handling policy when no agent can answer the call because no agent is on duty.<br>▪ **Release** (default)<br>▪ **Transfer** |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| | | – **Device Type**: If **Process Method** is set to **Transfer**, you need to configure the skill queue or IVR flow to which the call is transferred.<br>▪ **Skill Queue**<br>▪ **IVR**<br>● **Queuing and waiting configuration**: When a customer needs to wait in a queue after making an inbound call, a voice can be played to optimize the customer waiting process.<br>– **Queuing Method**<br>▪ **Default Wait Tone**<br>▪ **Customizing the Wait Tone**<br>▪ **IVR**<br>● **Keeping and waiting configuration**: When a call needs to be held and the customer needs to wait, a voice can be played to optimize the customer waiting process.<br>– **Keeping Method**<br>▪ **Default Keeping Tone**<br>▪ **Customizing the Keeping Tone**<br>● **Skill AnswerMode**: After an agent answers a call from a customer, the employee ID of the agent can be played to the customer.<br>– **Answer Type**<br>▪ **Report employee ID**<br>▪ **Report no voice**<br>**NOTE**<br>Calls in voice, video, and click-to-dial skill queues can be transferred to IVRs or skill queues. Calls in multimedia skill queues can be transferred only to skill queues.<br>The skill queue to which a call is transferred and the skill queue that is created must belong to one type.<br>The waiting tones can be set only for voice and video skill queues. Click-to-dial skill queues are not supported. |

1. Click **Complete**.

**Step 4** Configure a multimedia called route.

1. Choose **Configuration Center** > **Access Configuration** > **Called Route**.
2. Click **New** to add parameter information for the VDN and click **Complete**, as shown in **Figure 1-8**. **Table 1-3** describes the parameters. If you need to enable the click-to-dial function, also configure a click-to-dial called route.

**Figure 1-8** Page for configuring a called route

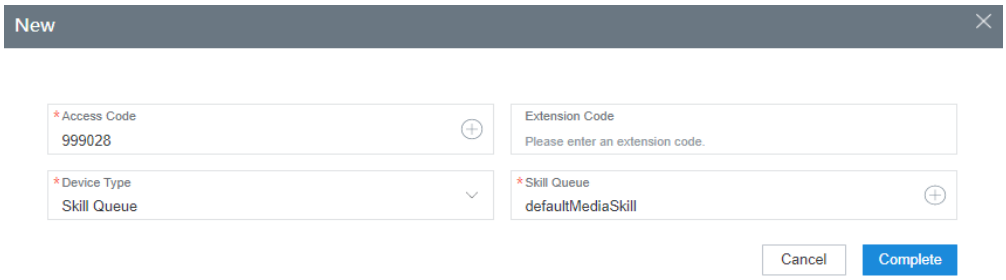



**Table 1-3** Parameters for configuring a called route

| Parameter | Mandatory or Not | Description |
|---|---|---|
| Access Code | Yes | Customer service hotline. Customers can dial the access code to connect to agents.<br>Click ⊕ to select an access code, for example, a multimedia access code, from the list in the dialog box. |
| Extension Code | No | To set one access code for multiple destination devices, you can configure extension codes.<br>For example, if the access code is 12345, you can add extension code 1 to route calls to skill queue A and extension code 2 to route calls to skill queue B. In this way, customers can dial 123451 to directly access skill queue A. |
| Device Type | Yes | Select **Skill Queue** to configure a called route of the skill queue type. |
| Skill Queue | Yes | Associate the skill queue created in **Step 3**.<br>**Skill Queue**: Click ⊕ to select a skill queue from the list in the dialog box. The type of the skill queue is the same as that of the access code. For example, if you select a multimedia access code, all available skill queues are of the multimedia type. |

**Step 5** Configure a business account and skill queue.

1. Choose **Configuration Center** > **Employee Center** > **Agent Management**.
2. Select an agent and click **Configure** in the **Operation** column. The agent information configuration page is displayed.
3. Associate a business account and skill queue with the agent. If you need to enable the click-to-dial function, also associate a click-to-dial skill queue.

**Figure 1-9** Page for configuring agent information

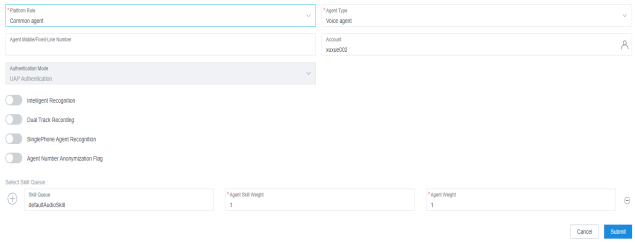



**Table 1-4** Parameters for configuring agent information

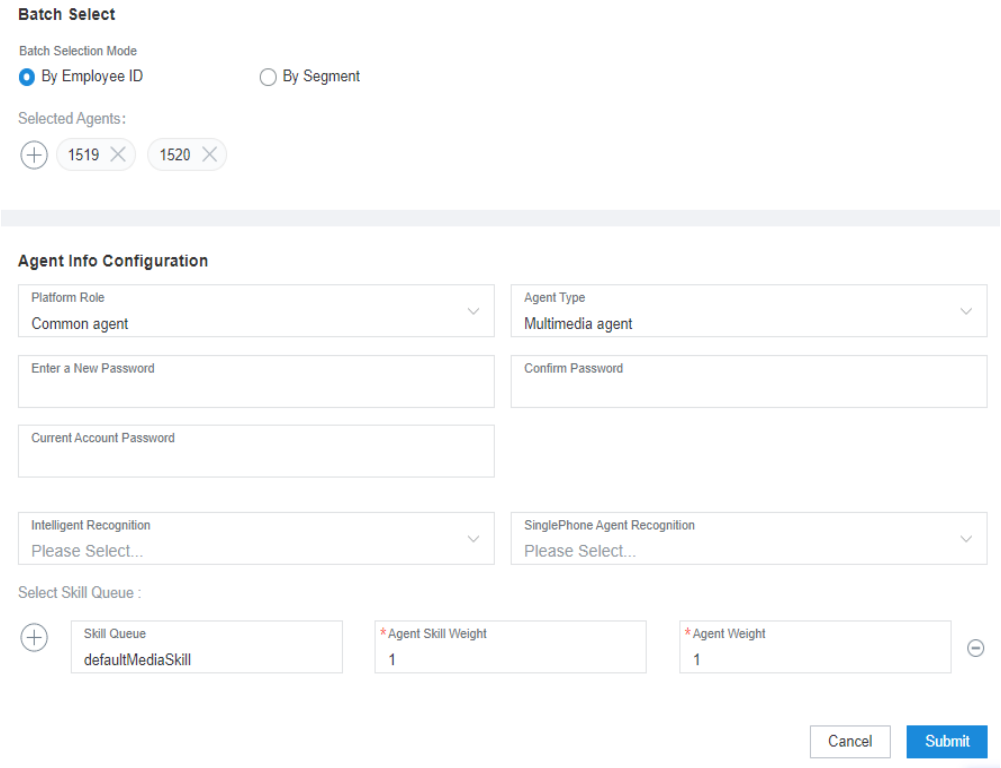
| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Platform Role | Yes | Agent role. This parameter is mandatory.<br>– **Common agent**: This role can answer or transfer inbound calls from customers.<br>– **Quality checker**: This role can intervene in calls between common agents and customers. For example, this role can perform operations, such as insertion, interception, and forcible busy state setting, to coach and supervise agents' handling of inbound calls.<br>– **Callout agent**: This role can answer, transfer, or reject inbound calls from customers. |
| Agent Type | Yes | Agent type based on the businesses that can be handled. This parameter is mandatory.<br>– **Audio agent**<br>– **Video agent**<br>– **Multimedia agent**<br>– **Versatile agent** |
| Agent Mobile/Fixed-Line Number | No | Mobile number or fixed-line phone number used by an agent. |
| Account | Yes | Employee account.<br>For details, see **Managing Employees**. |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Intelligent Recognition | No | Whether an agent is an intelligent agent. By default, this switch is turned off. In addition to basic voice control functions, intelligent agents support real-time ASR and related intelligent recommendation functions. Before turning on this switch, ensure that the number of agents for which intelligent recognition is enabled does not exceed the number of intelligent agents allocated when the tenant is created. |
| SinglePhone Agent Recognition | No | After this switch is turned on, an agent can dial a specified access code to access an IVR flow, press a key as prompted to enter the employee ID and password to sign in, and answer calls on a mobile phone. When this switch is turned on, system O&M personnel need to customize the single-phone agent process for the tenant based on the platform, and the tenant needs to provide number resources for accessing the single-phone agent process. |
| Agent Number Anonymization Flag | No | Flag for a third party to mark whether an agent has the anonymization feature. This is not a feature switch. The anonymization feature enables agents to customize the calling number displayed on the user side (the calling number displayed to the user) and the calling number displayed on the agent side (the calling number displayed to the customer manager). |
| Select Skill Queue | Yes | Skill queue of an agent. When selecting multiple skill queues, set them to the same media type except for versatile agents. For example, set them to **Audio agent** or **Multimedia agent**.<br>**NOTE**<br>– If **Agent Type** is set to **Video agent**, the corresponding number of video agents must have been applied during tenant resource application.<br>– If **Agent Type** is set to **Multimedia agent**, the corresponding number of multimedia agents must have been applied during tenant resource application.<br>– If **Agent Type** is set to **Versatile agent**, the corresponding number of versatile agents must have been applied during tenant resource application.<br>– To add more business accounts, choose **Configuration Center** > **Employee Center** > **Employee**. |

4. Click **Submit**. The business account and skill queue are associated with the agent ID.
5. (Optional) Click **Batch Configure**. On the agent information configuration page that is displayed, configure agent information in batches.

**Figure 1-10** Page for configuring agent information in batches




- **Batch Select**: Select agents to be configured by employee ID or employee ID segment.
- **Agent Info Configuration**: Set parameters by referring to **5**.

**Step 6** Configure the web channel.

1. Choose **Configuration Center** > **Access Configuration** > **Channel Configuration**.
2. Click **New**, set **Channel Access Code**, select **WEB**, and click **Next** to enter the page for configuring the web channel.

**NOTE**

The channel access code must be unique. The code can contain only letters, digits, and underscores (_), and must start with a letter or an underscore (_).

3. Set the web channel parameters shown in **Figure 1-11** based on **Table 1-5**. If you need to enable the click-to-dial function, configure **CTD Called Party Configuration** and **Click-to-Call Skill Queue** on this page.

**Figure 1-11** Web channel configuration page

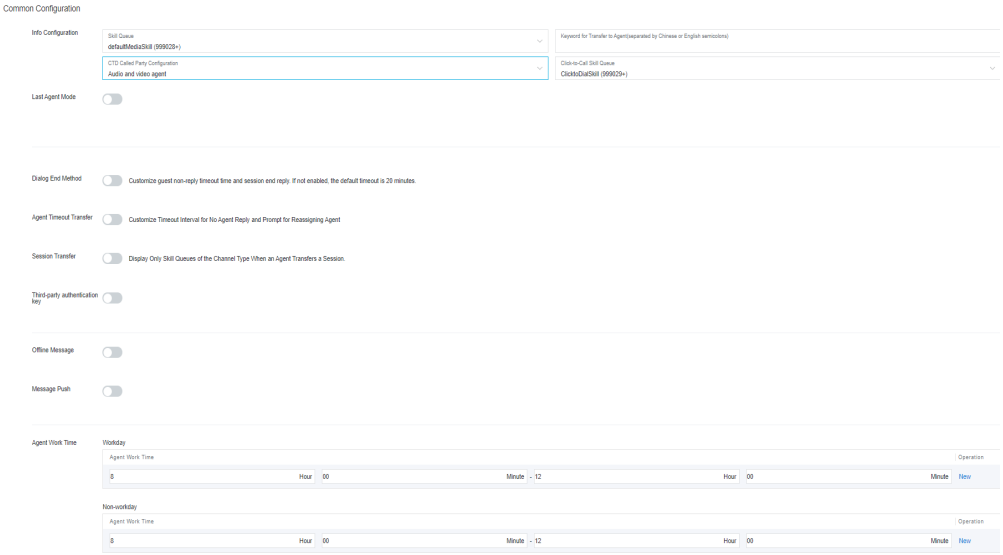



**Table 1-5** Web channel parameters

| Parameter | Description |
| --- | --- |
| **Common Configuration** | |
| Info Configuration | Set the following parameters:<br>– **Skill Queue**: The options are all multimedia called parties of the current tenant space. Select an option as required.<br>– **Keyword for Transfer to Agent**: Keywords for switching from chatbot service to manual service. After a customer enters any of the keywords on the HTML5 client, chatbot service is switched to manual service.<br>**NOTE**<br>If the intelligent chatbot is disabled, you do not need to set this parameter.<br>– **CTD Called Party Configuration**: The options are all voice and video agents and IVRs of the current tenant space.<br>– **Click-to-Call Skill Queue**: If **CTD Called Party Configuration** is set to **Audio and video agent**, you need to configure the related skill queue.<br>– **Click to obtain the IVR access code**: If **CTD Called Party Configuration** is set to **IVR**, you need to configure the IVR access code. |

| Parameter | Description |
| --- | --- |
| Dialog End Method | This function is disabled by default. If it is disabled, the default timeout period is 20 minutes. If it is enabled, set the following parameters:<br>– **Prompt Interval for No Reply (min)**: If a customer does not reply on the client within this period, the session is disconnected.<br>– **Conclusion**: The system sends an end reminder after the session is disconnected. |
| Third-party authentication key | This function is disabled by default. If it is enabled, set **secretKey**. This parameter is mandatory when the authorization signature authentication mode is used. |
| Agent Work Time | – **Workday**: A maximum of four working time segments (from 00:00 to 24:00) can be configured. By default, a time segment is displayed. You can click **New** to add a time segment.<br>– **Non-workday**: A maximum of four working time segments (from 00:00 to 24:00) can be configured. By default, a time segment is displayed. You can click **New** to add a time segment.<br>– **Non-Working Time Notification**: Notification displayed when a customer calls in non-working time. |
| Queue reminder | Set the following parameters:<br>– **Queue reminder interval (seconds)**: This parameter is mandatory. The default value is **10**.<br>– **Queue reminder content**: Notification displayed if a customer is in a queue after making an inbound call. |

| Parameter | Description |
| --- | --- |
| Chatbot Configuration | This function is disabled by default. If it is enabled, customers are preferentially connected to the chatbot. Set the following parameters:<br>– **Change Avatar**: Chatbot avatar.<br>– **Name**: Chatbot name.<br>– **Gender**: Chatbot gender.<br>– **Chatbot Access Code**: Chatbot access code configured in the intelligent IVR.<br>– **Default reply**: Customized reply displayed when the chatbot cannot recognize the intent of a customer.<br>– **Timeout reply**: Customized reply displayed when the session with a customer times out.<br>– **Prompt for transfer to agent**: Customized prompt message indicating that chatbot service is switched to manual service.<br>**NOTE**<br>If you want to enable the chatbot function to allow customers to chat with the chatbot, configure the chatbot based on the required chatbot type. For details, see **Configuring Intelligent IVR**. |
| Robot Assistant Configuration | This function is disabled by default. If it is enabled, the chatbot assistant is enabled on the agent side.<br>– **Assistant Access Code**: Chatbot access code configured in the intelligent IVR.<br>– **SilentAgent Skill Queue**: The options are all multimedia called parties of the current tenant space. Select an option as required. |

4. Click **Save And Proceed To The Next Step**. The **Integration instructions** page is displayed.

**Step 7** (Optional) On the **Integration instructions** tab page, click **Try**. On the page that is displayed, set customer information to simulate the dialog window on the customer side.

Verify that customers can chat with agents or the chatbot through the current channel.

1. Click **Try**, and click  in the lower right corner of the page that is displayed. The **Online Customer Service** dialog box is displayed. The online customer service has two chat modes:

   a. If **Connect to Chatbot** is enabled, customers are connected to and chat with the chatbot by default. If the chat content entered by a customer contains a keyword that can be recognized by the chatbot, the chatbot recognizes the keyword and replies to the customer.

b. If **Connect to Chatbot** is disabled, customers are automatically connected to and chat with online agents. Click **Sign In** on the CEC to sign in as a multimedia agent, and choose **Online Chat Workbench**. The workbench of the current session is displayed. After a customer is connected, you can chat with the customer online.

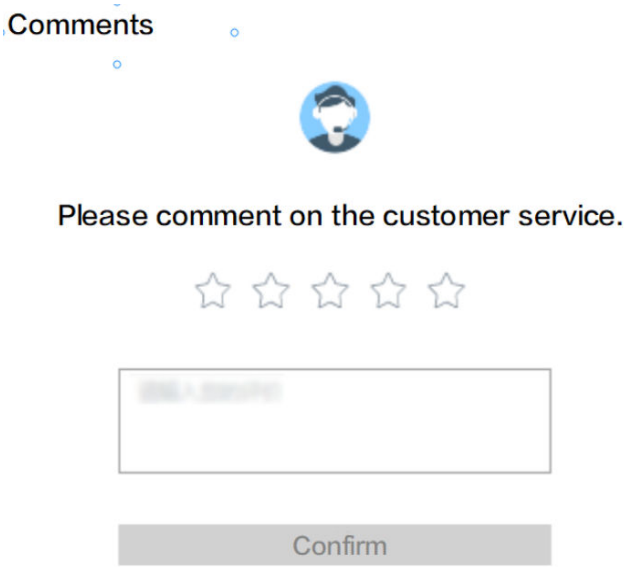
NOTE

When chatting with the chatbot, the customer can click **Transfer to Agent** or enter content that contains any of the keywords specified by **Keyword for Transfer to Agent** to switch from the chatbot to an agent. (**Keyword for Transfer to Agent** can be set in **Step 6**.) However, the customer cannot switch to the chatbot or another agent when chatting with an agent.

2. In the **Online Customer Service** dialog box, enter the chat content, click **Send**, and check the reply of the chatbot or agent.

3. (Optional) Click  and choose **Evaluation** to comment on the customer service, including the satisfaction rating and evaluation content, as shown in **Figure 1-12**. Click **Confirm** to submit the evaluation.

**Figure 1-12** Comments page




NOTE

Customers can evaluate the agents who chat with them. During or after a chat, the customer can evaluate the service of the agent at any time. The last evaluation is used.

**----End**

After the channel configuration is complete, record the channel ID, which will be used during token authentication API development and page integration.

| Configuration ID | Channel Access Code | Channel Type | Bind Skill Queue | Operation |
|---|---|---|---|---|
| 869525871671777281 | WECHAT001 | WECHAT | defaultMediaSkill (10001+) | Modify Delete |
| 853180664420335631 | wechat001 | WECHAT | defaultMediaSkill (10001+) | Modify Delete |
| 827635713993441281 | hyy | WECHAT | defaultMediaSkill (10001+) | Modify Delete |
| 817854979183314945 | WEBCHAT | WEB | defaultMediaSkill (10001+) | Modify Delete |
| 743032024951966724 | A20191231 | WECHAT | defaultMediaSkill (10001+) | Modify Delete |
| 740930049837911041 | a123456 | WECHAT | defaultMediaSkill (10001+) | Modify Delete |

# 1.5 Integration Development

This section describes how to integrate the web chat control in your system.

## 1.5.1 Developing a Token Generation Mechanism and Token Authentication API

You need to develop a token generation mechanism and token authentication API for the CEC to perform functions such as authentication. When receiving your JavaScript request, the CEC invokes this API to send the enterprise token, tenant space ID, and user account to your system for confirmation. After the confirmation, the CEC performs internal verification and request processing.

**NOTE**

A demo of the token generation mechanism and token authentication API is provided for your reference. You can download the demo from the Huawei Developer Forum (https://bbs.huaweicloud.com/forum/thread-192048-1-1.html). The demo provides the algorithm and verification logic. Use a secure algorithm in actual use. This demo cannot be directly used for production and is for demonstration only. Design the token generation mechanism and token authentication API based on your system security requirements.

### Procedure

**Step 1** Develop a token generation mechanism. You need to develop the mechanism based on the features and security requirements of your system.

**Step 2** Develop a token authentication API based on the specifications provided by the CEC. **Table 1-6** and **Table 1-7** describes the requirements of each request parameter and response parameter. The request URL is as follows:

POST http(s)://*ip*:*port*/rest/cc-messaging/v1/thirdparty/chatThirdPartyValidate (which can be customized by the enterprise)

**NOTE**

HTTP is an insecure protocol, which may bring risks to the system. Therefore, it is not recommended. The secure HTTPS is recommended. If HTTPS is used, you need to prepare a certificate in CER format to verify the validity of the HTTPS website.

**Table 1-6** Request body description

| Parameter | Type | Position | Mandatory or Not | Description |
|---|---|---|---|---|
| Token | String | Body | Yes | Verification information generated on the enterprise/partner side. |
| userName | String | Body | Yes | Username provided by the enterprise/partner.<br>The value must exist in the enterprise authentication system. |
| userId | String | Body | Yes | User ID provided by the enterprise/partner.<br>The value must exist in the enterprise authentication system. |
| tenantSpaceId | String | Body | Yes | Tenant space ID provided by the CEC. To obtain the tenant space ID, choose **Configuration Center** > **System Management** > **Tenant Information**. |

- Example

```
{
   "Token":"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
   "userName":"Tom",
   "userId":"10010001"
   "tenantSpaceId":"202007159031"
 }
```

**NOTE**

The preceding username and user ID are examples. Ensure that the enterprise customer exists.

**Table 1-7** Response description

| Parameter | Type | Position | Mandatory or Not | Description |
|---|---|---|---|---|
| retCode | Integer | Body | Yes | Interface result identifier.<br>**0** indicates success.<br>Other values indicate failure. |
| message | String | Body | Yes | Result information. |

- Example

```
{
    "retCode":0,
    "message":"validate success"
}
```

**Step 3** (Required if HTTPS is used) Submit the URL and certificate file of the authentication API to the CEC operations administrator. The operations administrator configures the URL and certificate file in the CEC.

**----End**

## 1.5.2 Developing an Integration Page

### 1.5.2.1 Integrating Core Code

#### Prerequisites

The Huawei Cloud operations administrator has configured the authentication API in the CEC.

#### Procedure

**Step 1** Introduce the JavaScript dependent framework first. In the following example, the jQuery and Axios components need to be introduced. The reference versions are jQuery 1.8.0.js and Axios.min.js. If the co-browsing function is required, introduce the cobrowse.js and cobrowseCommon.js components or release the cobrowse plugin for use. (For details about the development of the plugin, see section 3.1 "Guide to the Development and Usage of the cobrowse Plugin.") The cobrowse.js and cobrowseCommon.js components are introduced in CDN mode. Replace *ip:port* with the IP address and port number of the CEC, or directly with the domain name of the CEC, for example, **https://www.test.com/service-cloud/resource.root/cobrowse/sdk/cobrowse.js**. Figure 1 shows an example of the code. (Notes: 1. Co-browsing operations cannot be performed on a new tab page.) 2. When the cobrowse.js and cobrowseCommon.js components for co-browsing are introduced and the third-party page contains nested iFrames, co-browsing operations can be performed only in the iFrame where the JavaScript components are introduced. If an iFrame has a parent iFrame, co-browsing operations cannot be performed in the parent iFrame.

**Figure 1-13** Code example

```
<script type="text/javascript" src="https://ip:port/service-cloud/resource.root/cobrowse/sdk/cobrowse.js"></script>
<script type="text/javascript" src="https://ip:port//service-cloud/webclient/chat_client/js/cobrowseCommon.js"></script>
<script type="text/javascript" src="js/jquery-1.8.0.js"></script>
<script type="text/javascript" src="js/axios.min.js"></script>
```

The cobrowse.js and cobrowseCommon.js components are introduced in CDN mode. Replace *ip:port* with the IP address and port number of the CEC, or directly with the domain name of the CEC, for example, **https://www.test.com/service-cloud/resource.root/cobrowse/sdk/cobrowse.js**.

**Step 2** Obtain the JavaScript file address provided by the CEC, as shown in the following information. Replace the domain name with the actual one of the CEC.

https://***servicestage.besclouds.com***/service-cloud/webclient/chat_client/js/thirdPartyClient.js?&t=***1595993533588***

**Step 3** On the enterprise side, use Ajax to request to integrate the JavaScript script in the CEC. The following is the JavaScript code snippet. Variable definitions are described in the following table.

```
   // Define the $aicc_ContextPath variable. The variable name must be $aicc_ContextPath, and the value
is https://IP address:Port number/service-cloud/ or https://Domain name/service-cloud/.
 const $ContextPath = "https://ip:port/service-cloud"
   // Construct request parameters.
 let timestamp = new Date().getTime();
 let serviceUrl = $ContextPath+ "/webclient/chat_client/js/thirdPartyClient.js?"+"&t=" + timestamp;
   // The following variables need to be defined before being used. In this demo, the variables are defined
when axios.js is introduced. For details about the parameters, see Table 1-8.
 let thirdUserData = {};
 thirdUserData['thirdUserToken'] ="XXXXXXXX";
 thirdUserData['thirdUserName'] = "XXXXXXXX"
 thirdUserData['thirdUserId'] = "XXXXXXXX";
 thirdUserData['tenantSpaceId'] = "XXXXXXXX";
 thirdUserData['channelConfigId'] = "XXXXXXXX";
 thirdUserData['locale'] = "xx";
 thirdUserData['mapService'] = 'google';

   var importScript = (function (oHead) {
       function loadError(oError) {
           throw new URIError("The script " + oError.target.src + " is not accessible.");
       }
       return function (sSrc, fOnload) {
           var oScript = document.createElement("script");
           oScript.type = "text\/javascript";
           oScript.onerror = loadError;
           if (fOnload) { oScript.onload = fOnload; }
           oHead.appendChild(oScript);
           oScript.innerHTML = sSrc;
       }

   })(document.head || document.getElementsByTagName("body")[0]);

   function thirdValidate(thirdUserData) {
       let timestamp = new Date().getTime();
       let serviceUrl = $ContextPath + "/webclient/chat_client/js/newThirdPartyClient.js?"+"&t=" + timestamp;
   // Request newThirdPartyClient.js.
       var request = $.ajax({
           url: serviceUrl,
           type: "POST",
           data: JSON.stringify(thirdUserData),
           crossDomain: true,
           dataType:"text",
           xhrFields: {withCredentials: true},
           error: function (XMLHttpRequest, textStatus, errorThrown) { console.log(XMLHttpRequest.status);
console.log(XMLHttpRequest.readyState); console.log(textStatus); },
           success: function (data) {
               importScript(data)
           }
       });
   }

   applyChatWindow();
   function applyChatWindow() {

       thirdUserData['timestamp'] = timestamp;
       // Request the JavaScript file from the Service Cloud.
       axios({
           method: 'post',
           url: serviceUrl,
           data: JSON.stringify(thirdUserData),
           withCredentials: true
         })
```

```
        // Define JavaScript file variables and write them to the current web page.
        .then(resp => {
        if (resp && resp.status === 200) {
            let str = 'var configId = "' + thirdUserData.channelConfigId + '";var userId = "' +
thirdUserData.thirdUserId + '";var userName = "' + thirdUserData.thirdUserName
                + '";var tenantSpaceId = "' + thirdUserData.tenantSpaceId + '";var locale = "' +
thirdUserData.locale
                + '";var token="'+ thirdUserData.thirdUserToken+'";';
            importScript(str+resp.data);
        } else {
          this.$alert ("Link failed!");
          console.log(resp.status);
          console.log(resp);
        }
      });
  }
```

**----End**

**Table 1-8** Integration page parameters

| Parameter | Mandatory or Not | Description |
|---|---|---|
| $ContextPath | Yes | Variable part of the request address in **Step 2**, which is usually a domain name or an IP address and port number. Replace *IP address*:*Port number* with the public network domain name of the AICC.<br>You are advised to configure the domain name or the IP address and port number in the configuration file or configuration table. The value of **$ContextPath** is obtained from the configuration item. |
| serviceUrl | Yes | Complete path obtained in **Step 2**.<br>Generally, the value of **serviceUrl** is not a full address. Instead, it consists of the **$ContextPath** variable and the fixed part. Ensure that the value of **serviceUrl** is the same as the request address provided by the CEC. |
| thirdUserToken | Yes | Native verification information of an enterprise, which is the same as the value of **Token** in the enterprise verification API. |
| tenantSpaceId | Yes | Tenant space ID. Choose **Configuration Center** > **System Management** > **Tenant Information** to view the value. |
| thirdUserId | Yes | Enterprise user ID. The value must be the same as that of **userId** in **Table 1-6**. |
| thirdUserName | Yes | Enterprise username. The value must be the same as that of **userName** in **Table 1-6**. |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| channelConfigId | Yes | Channel configuration ID. After the operations in Step 7 are complete, choose **Configuration Center** > **Access Configuration** > **Channel Configuration** to view the value. |
| locale | Yes | Tenant space language provided by the CEC.<br>● **zh**: Chinese<br>● **en**: English |
| mapService | No | User map service.<br>● **tencent**<br>● **google**<br>The default value is **tencent**.<br>**NOTE**<br>Tencent Maps does not support locations outside China. |

### 1.5.2.2 Example: JavaScript Page Integration Code

| | |
| --- | --- |
| **Environment Requirement** | - |
| **Reference Library** | jQuery 1.8.0.js/Axios.min.js |
| **Integration Example** | index.html |

**NOTICE**

- The demo described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take sufficient measures to ensure that personal data is fully protected.
- The demo described in this document is for demonstration only. Commercial use of the demo is prohibited.
- Information in this document is for reference only and does not constitute any offer or commitment.

## index.html

```
<!DOCTYPE html>
<!--
  Tutorial.
  Check available devices.
-->
<html>
```

```
<head>
    <meta http-equiv="x-ua-compatible" content="IE=edge">
    <meta charset="UTF-8">
    <title>Online customer service</title>
    <!-- If the co-browsing function is required, introduce the cobrowse.js and cobrowseCommon.js
components. Replace ip:port with the IP address and port number of the CEC, or directly with the domain
name of the CEC. -->
    <script type="text/javascript" src="https://ip:port/service-cloud/resource.root/cobrowse/sdk/
cobrowse.js"></script>
    <script type="text/javascript" src="https://ip:port//service-cloud/webclient/chat_client/js/
cobrowseCommon.js"></script>
    <script type="text/javascript" src="js/jquery-1.8.0.js"></script>
    <script type="text/javascript" src="js/axios.min.js"></script>
</head>

<body>
    <!-- Check that browser is not IE -->
    <script>
        var ua = window.navigator.userAgent;
        if (ua.indexOf('MSIE ') > 0 || ua.indexOf('Trident/') > 0) {
            alert("Internet Explorer is not supported. Please use Chrome or Firefox");
        }
    </script>
    <script>
    // Define the $aicc_ContextPath variable. The variable name must be $aicc_ContextPath, and the value
is https://ip:port/service-cloud/ or https://Domain name/service-cloud/.
    const $ContextPath = "https://ip:port/service-cloud"
    // Construct request parameters.
    let timestamp = new Date().getTime();
    let serviceUrl = $ContextPath+ "/webclient/chat_client/js/thirdPartyClient.js?"+"&t=" + timestamp;
    // The following variables need to be defined before being used. In this demo, the variables are defined
when axios.js is introduced. For details about the parameters, see Table 1-8.
    let thirdUserData = {};
    thirdUserData['thirdUserToken'] ="XXXXXXXX";
    thirdUserData['thirdUserName'] = "XXXXXXXX"
    thirdUserData['thirdUserId'] = "XXXXXXXX";
    thirdUserData['tenantSpaceId'] = "XXXXXXXX";
    thirdUserData['channelConfigId'] = "XXXXXXXX";
    thirdUserData['locale'] = "xx";

    var importScript = (function (oHead) {
        function loadError(oError) {
            throw new URIError("The script " + oError.target.src + " is not accessible.");
        }
        return function (sSrc, fOnload) {
            var oScript = document.createElement("script");
            oScript.type = "text\/javascript";
            oScript.onerror = loadError;
            if (fOnload) { oScript.onload = fOnload; }
            oHead.appendChild(oScript);
            oScript.innerHTML = sSrc;
        }

    })(document.head || document.getElementsByTagName("body")[0]);

    function thirdValidate(thirdUserData) {
        let timestamp = new Date().getTime();
        let serviceUrl = $ContextPath + "/webclient/chat_client/js/newThirdPartyClient.js?"+"&t=" + timestamp;
    // Request newThirdPartyClient.js.
        var request = $.ajax({
            url: serviceUrl,
            type: "POST",
            data: JSON.stringify(thirdUserData),
            crossDomain: true,
            dataType:"text",
            xhrFields: {withCredentials: true},
            error: function (XMLHttpRequest, textStatus, errorThrown) { console.log(XMLHttpRequest.status);
console.log(XMLHttpRequest.readyState); console.log(textStatus); },
            success: function (data) {
```

```
                importScript(data)
            }
        });
    }

    applyChatWindow();
    function applyChatWindow() {

        thirdUserData['timestamp'] = timestamp;
        // Request the JavaScript file from the Service Cloud.
        axios({
            method: 'post',
            url: serviceUrl,
            data: JSON.stringify(thirdUserData),
            withCredentials: true
          })
          // Define JavaScript file variables and write them to the current web page.
          .then(resp => {
          if (resp && resp.status === 200) {
              let str = 'var configId = "' + thirdUserData.channelConfigId + '";var userId = "' +
thirdUserData.thirdUserId + '";var userName = "' + thirdUserData.thirdUserName
                    + '";var tenantSpaceId = "' + thirdUserData.tenantSpaceId + '";var locale = "' +
thirdUserData.locale
                    + '";var token="'+ thirdUserData.thirdUserToken+'";';
              importScript(str+resp.data);
          } else {
            this.$alert ("Link failed!");
            console.log(resp.status);
            console.log(resp);
          }
        });
    }

</script>

    <!--  HTML components of simple GUI -->
    <div id="status_line">
    </div>
</body>

</html>
```

# 1.6 Test and Verification

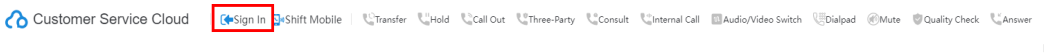
After page integration is complete, you need to test and verify that the web chat control can be used properly. The following uses Google Chrome as an example. During the verification, you can use the Nginx server to simulate a third party to invoke the web chat control or use other access methods that you are familiar with.

**Step 1** Enable the Nginx server locally (for details about the Nginx version, see **nginx/Windows-1.22.0**) and configure information, including the service address and certificate, in the **nginx.conf** file.

**Step 2** Enter the server address and send a request to simulate a third party to invoke the web chat control.

**Step 3** Click **Sign In** to sign in to the CEC as an agent in the multimedia skill queue, as shown in **Figure 1-14**. On the online chat workbench that is displayed, set the status to **Idle**.

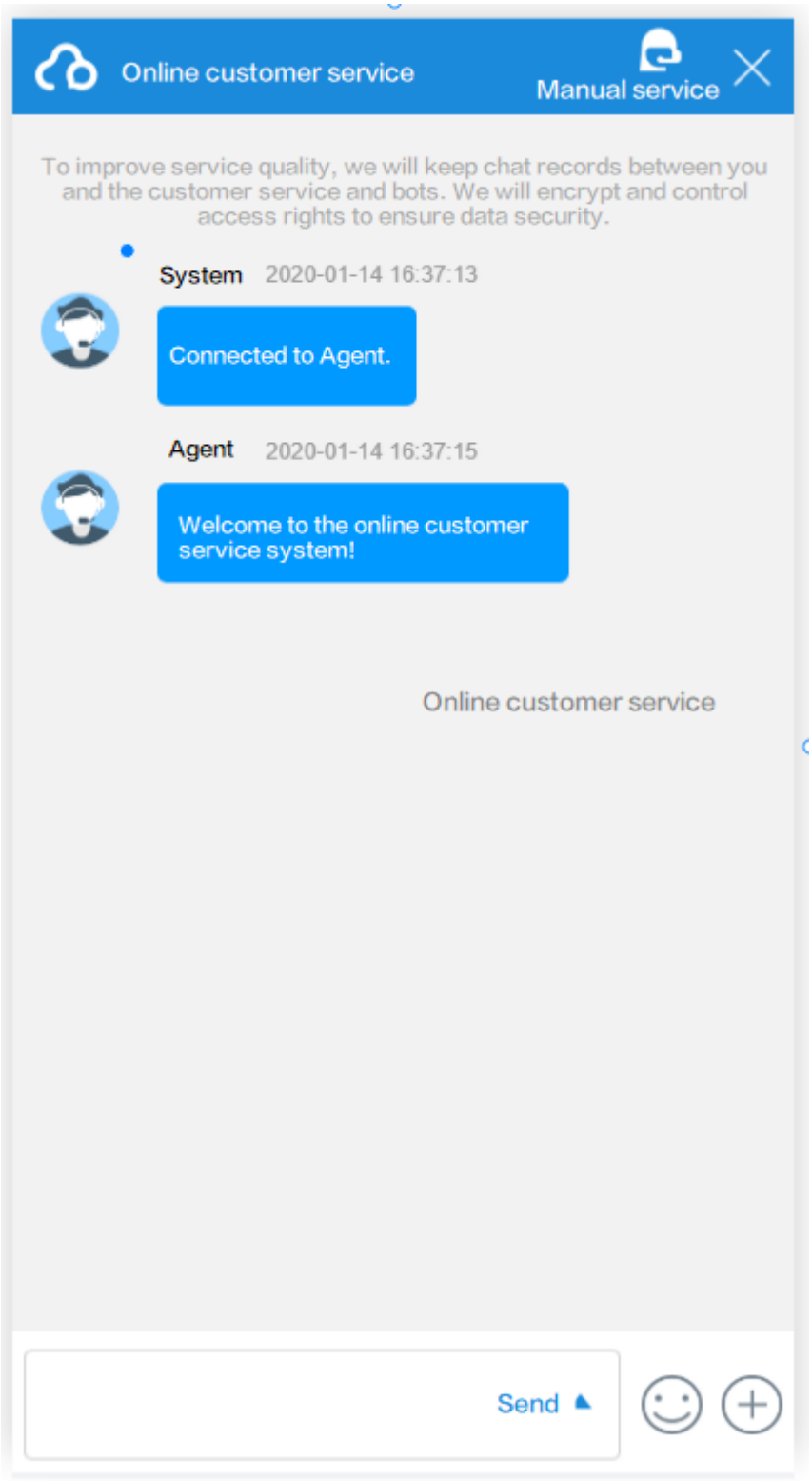
**Figure 1-14** Sign In button




**Step 4** On the enterprise customer page, press **F12** to open the console, choose **Network**, and refresh the page.

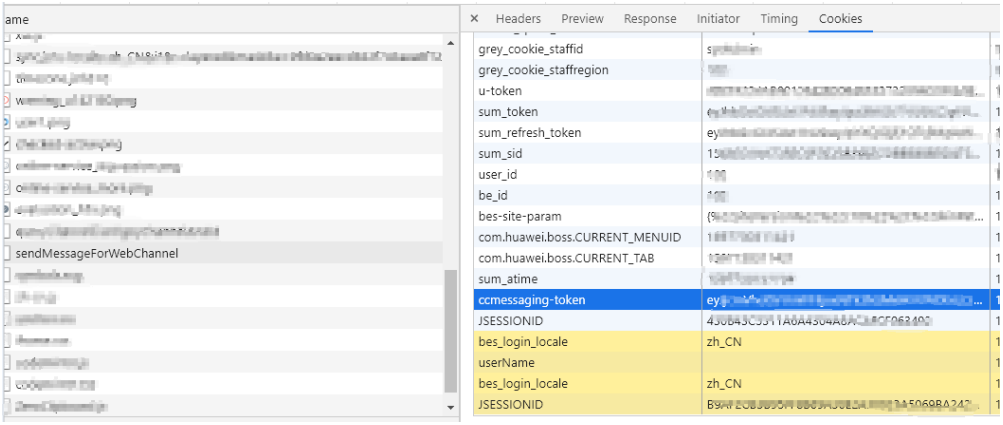
Click the **thirdPartyClient.js** request displayed on the console and click **Response** on the right. If any content is returned, a blue circle icon is displayed in the lower right corner. Click the icon. If a message indicating that an agent is connected is displayed, as shown in **Figure 1-15**, the invocation is successful.

**Figure 1-15** Dialog box on the customer side



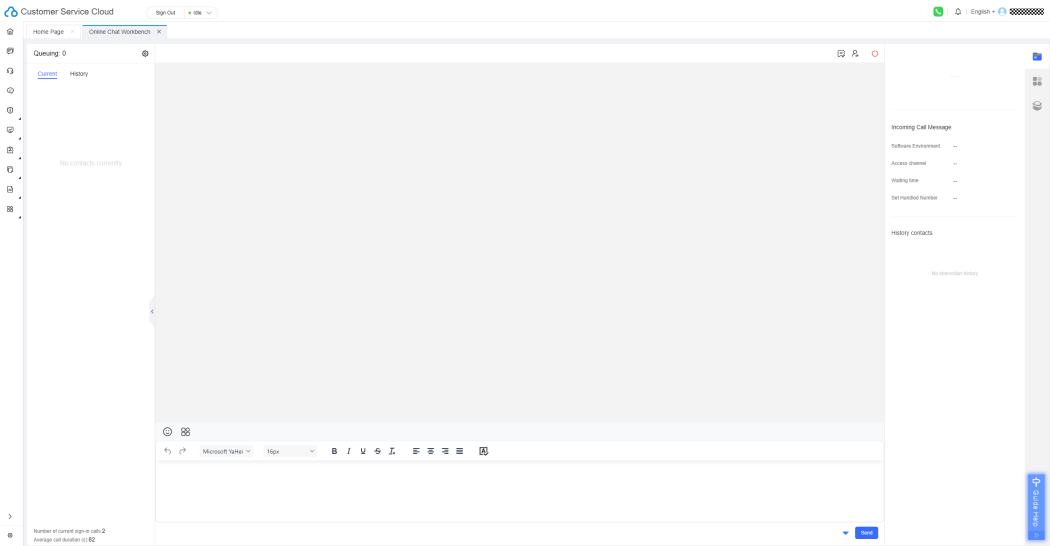

**Step 5** On the browser console, click the **Application** tab, expand **Storage** > **Cookies** > *Domain name* on the left, and check whether **ccmessaging-token** is written into cookies. If the field shown in **Figure 1-16** is displayed, the integration is successful.

**Figure 1-16** ccmessaging-token in cookies of the HTML page



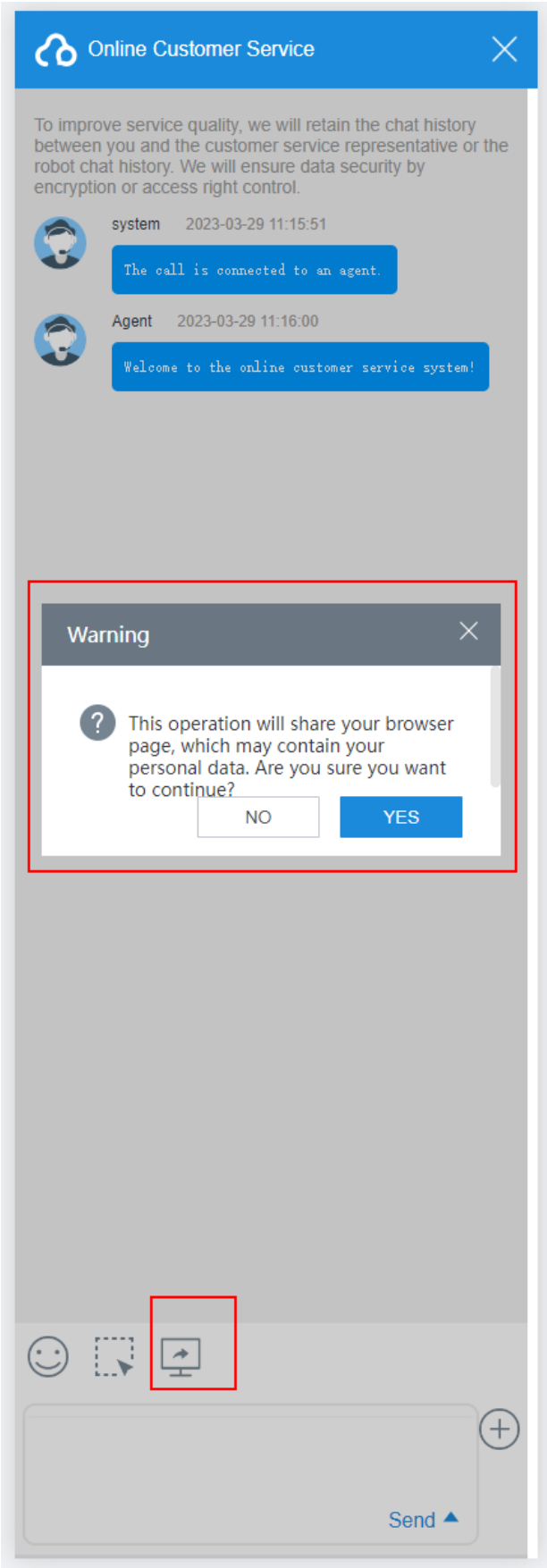

**Step 6** On the workbench shown in **Figure 1-17**, check whether the agent is in the occupied state and whether the agent receives the request from the client. If the agent can talk with the customer, the integration is successful.

**Figure 1-17** Agent workbench



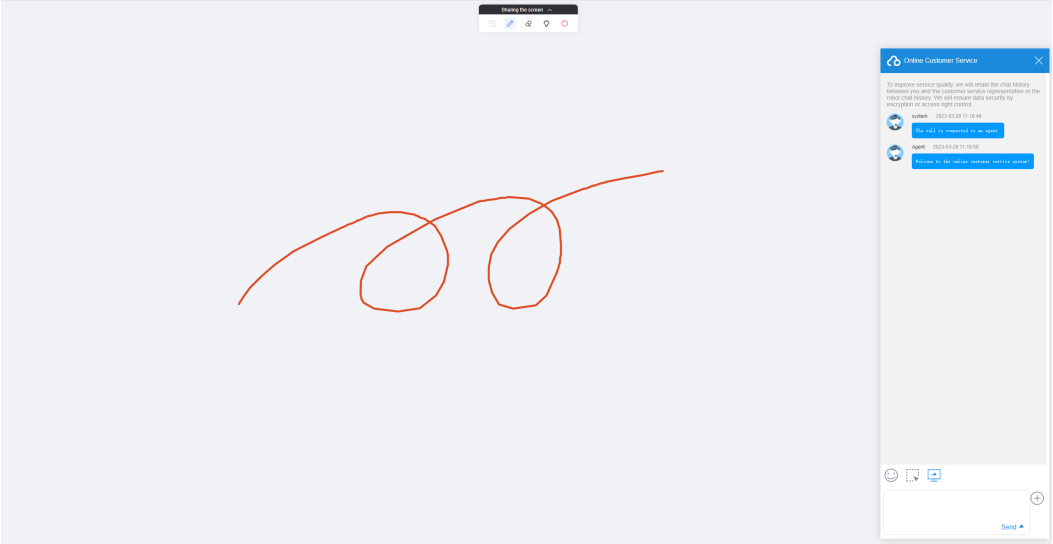

**Step 7** If the co-browsing feature is enabled for the tenant, a customer can initiate co-browsing in the dialog box on the customer side. After the customer clicks the co-browsing button, a confirmation dialog box is displayed. The customer can confirm the initiation.

**Figure 1-18** Dialog box on the customer side



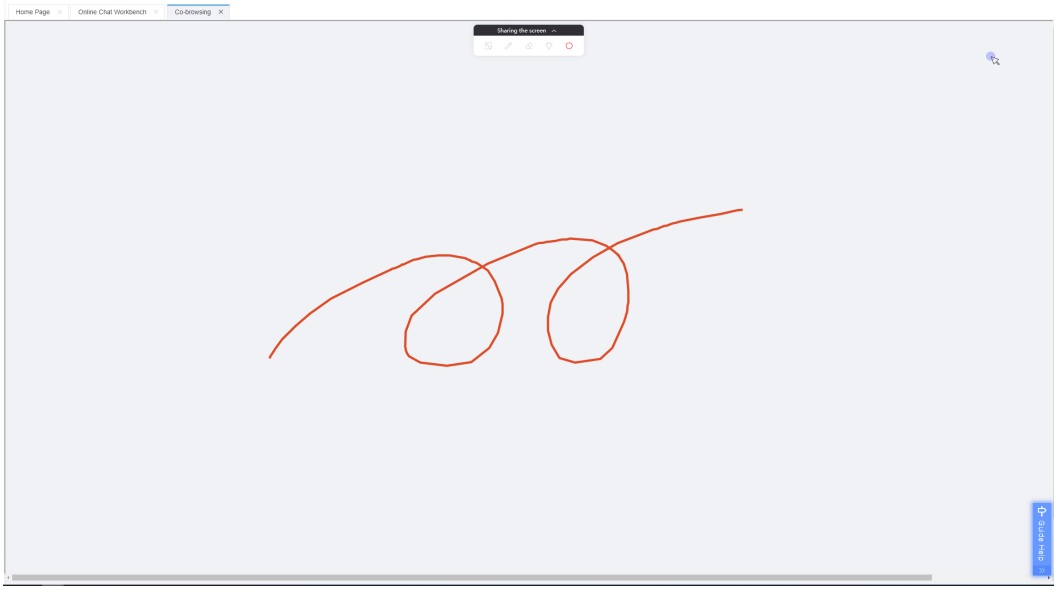
**Step 8** After the agent accepts the co-browsing request on the agent workbench, the customer can share the current page, mark or highlight content on the page, or request the agent's remote control of the page. When the co-browsing icon shown in **Figure 1-19** is displayed in the lower right corner of the page, co-browsing is successfully initiated.

**Figure 1-19** Co-browsing page on the customer side




**Step 9** On the agent workbench, the agent can view the customer's current page, view the content marked or highlighted by the customer, or request the remote control of the customer's page.

**Figure 1-20** Co-browsing page on the agent workbench




**Step 10** When the web page is scrolled, the marker cannot move with the scrolling of the web page, as shown in **Figure 1-21** and **Figure 1-22**. If the marker needs to move with the scrolling of the web page in a business scenario, clear the marker, scroll the page, and mark content again.

**Figure 1-21** Circled web page content

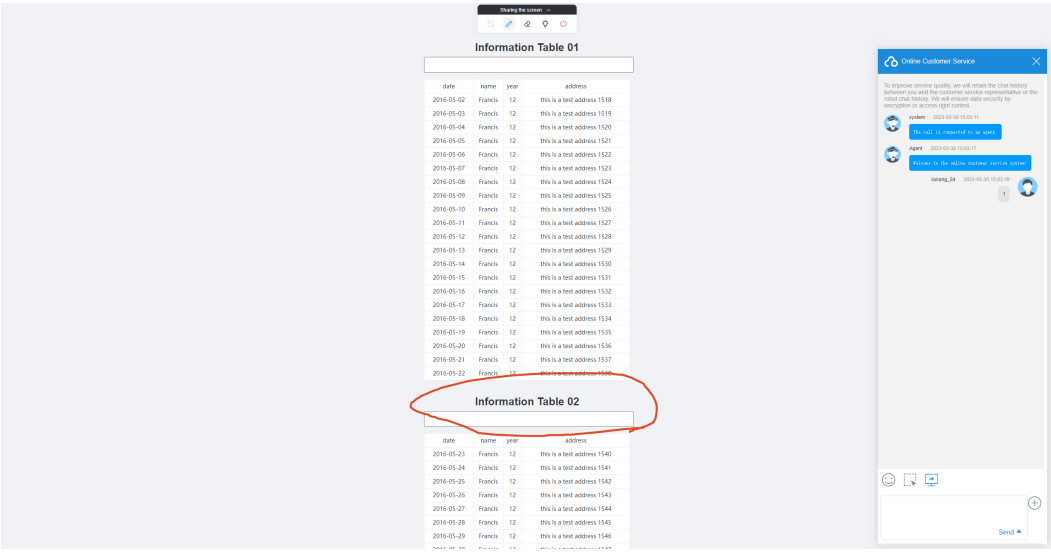
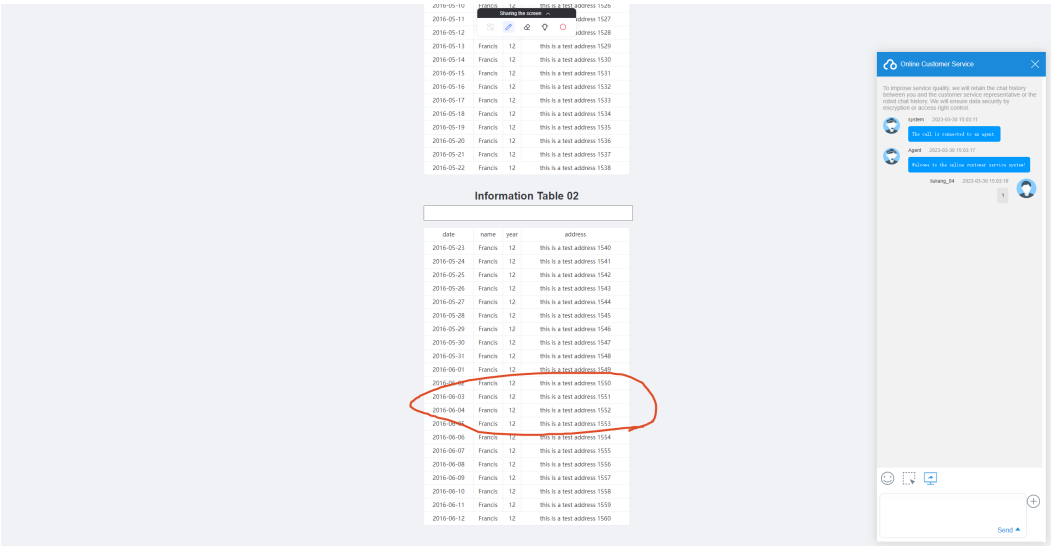



**Figure 1-22** Circle that cannot move with the scrolling of the web page




**NOTE**

Ensure that only one agent in the multimedia skill queue signs in to your tenant space. Otherwise, the system may route the session to another agent based on the routing rules. As a result, you may not receive the customer request.

**----End**

# 1.7 FAQs

# 1.7.1 How Can I Resolve the Reported Cross-domain Error When the xmlHttpRequest Requests the URL of the CEC?

## Symptom

The following error information is displayed:

```
Access to XMLHttpRequest at "requested js" from origin xx has been blocked by CROS policy: No 'Access-Control-Allow—Origin' header is present on the requested response;
```

## Solution

This is a cross-domain error: The website of the integrator does not allow requests for resources that are not provided by the local domain due to security restrictions. You can use the reverse proxy of a load balancing application (such as Nginx).

**Figure 1-23** Principles of address mapping on a load balancing application


When a third-party page uses JavaScript to invoke a service under the local domain name, the service is bypassed to the domain name of the CEC.

The CEC identifies the request only when it identifies **service-cloud**. Therefore, the request URL of the third-party page must contain **service-cloud**.

The following uses Nginx as an example to describe the overall configuration:

**Step 1** Add the first-layer service address to the **nginx.conf** file as follows:

```
location /demo/ {
    proxy_set_header Host $host;
    set $Real $proxy_add_x_forwarded_for;
    if ($Real ~ (\d+)\.(\d+)\.(\d+)\.(\d+)\,(.*)){
        set $Real $1.$2.$3.$4;
    }
    proxy_set_header X-Real-IP $Real;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    proxy_pass https://servicestage.besclouds.com/;
}
```

**NOTE**

- *demo* is an example value. The integrator can replace the value based on requirements.
- Replace *servicestage.besclouds.com* with the address provided by the CEC.

```
location /service-cloud/ {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    proxy_request_buffering off;
```

```
    proxy_pass https://servicestage.besclouds.com/service-cloud;
}
```

**Step 2** Use the following address to send a JavaScript request in the **1.5.2 Developing an Integration Page**. Replace *location.protocol* with the domain name of the integrator and the service name configured in the previous step, for example, **demo**.

```
const $ContextPath = location.protocol/service-cloud
let serviceUrl = $ContextPath+ "/webclient/chat_client/js/thirdPartyClient.js?"+"&t=" + timestamp;
let thirdUserData = {};
......
```

**----End**

# 2 Lightweight Web Chat Control Integration (Authorization Mode)



## 2.1 Overview

The lightweight web chat control is a customer-side online customer service product that can be quickly integrated into websites. **Currently, only web integration on PCs is supported.** You can use our product to quickly build an intelligent online customer service system in the CEC SaaS (Software as a Service) system to chat with customers online on web pages.

The lightweight web chat control provides the function of chatting with customers online on web pages. The features are as follows:

- It is lightweight, easy to be integrated into your web pages, and does not occupy the main pages of portals and workbenches.
- It is easy to operate. Customers can easily find the window to chat with the customer service personnel on the page. The customer service personnel can send texts, emoticons, pictures, and quick responses on the agent client interface.

**NOTE**

Before using this document, you need to learn about the following two customer authentication modes:

1. Token mode: Used in scenarios where access customers need to be authenticated.

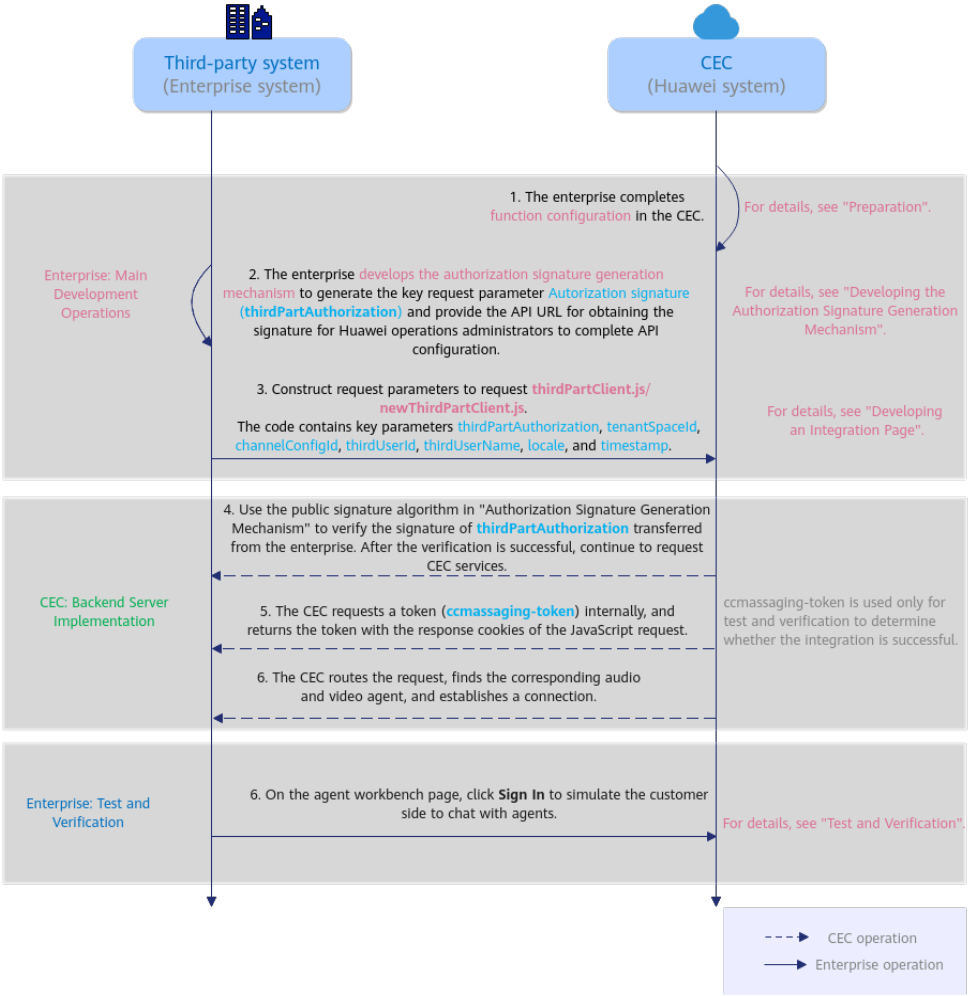
2. Authorization mode: Guest mode that is used in scenarios where access customers are not authenticated.

Select the development guide based on your requirements. If you want to use the token mode, go to **1 Lightweight Web Chat Control Integration (Token Mode)**.

# 2.2 Integration Principle

**Figure 2-1** shows the implementation principle.

**Figure 2-1** Integration principle of the web chat control in authorization mode

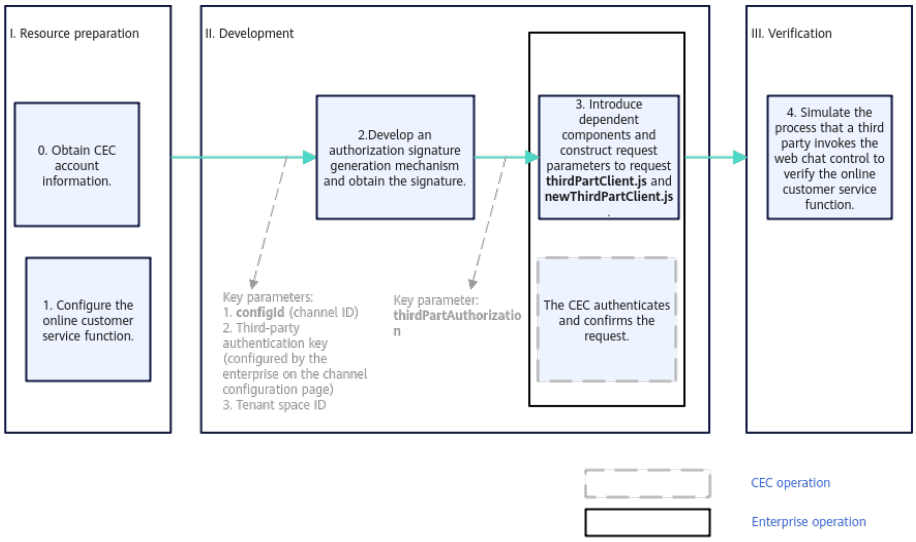



# 2.3 Integration Procedure

The lightweight web chat control can be quickly and efficiently integrated into your pages. You can perform integration development by referring to **Figure 2-2**.

**Figure 2-2** Integration development procedure




**Step 1** Before the development, prepare resources, obtain sign-in information, and configure the online customer service function. For details, see **2.4 Preparation**.

**Step 2** Develop an authorization signature generation mechanism and perform authorization signature authentication on your server. For details, see **2.5.1 Developing an Authorization Signature Generation Mechanism**.

**Step 3** Introduce related dependent frameworks, construct request parameters to request **thirdPartyClient.js** and **newThirdPartyClient.js**, and return the authorization signature, tenant space ID, and enterprise user ID and username to the CEC for authentication. After the authentication is successful, the lightweight access of the CEC web page chat capability can be implemented. For details, see **2.5.2 Developing an Integration Page**.

**Step 4** Test and verify the functions of the web chat control. Initiate an online chat in the customer-side window to check the function of chatting with agents. For details, see **2.6 Test and Verification**.

**----End**

# 2.4 Preparation

This section describes the preparations that need to be completed in the CEC before integration development.

## 2.4.1 Preparing Resources

Before the integration, you need to complete the following preparations:

1. You have applied for tenant information from the CEC. The system O&M administrator has added tenant information for you and provided the following information for you.

**Table 2-1** Parameters provided by the Huawei O&M administrator

| Parameter | Description |
| --- | --- |
| **TenantId** | Tenant space ID generated by the system after the tenant space (that is, your CEC) is successfully created.<br>To obtain the tenant space ID, sign in to your tenant space and choose **Configuration Center** > **System Management** > **Tenant Information**. |
| **Account** | Account for signing in to the CEC. |
| **Password** | Password for signing in to the CEC. |

In addition, you need to obtain and configure parameters including **configId**, **accessKey**, and **secretKey**.

**Table 2-2** Parameters to be obtained from or configured in the CEC

| Parameter | Description |
| --- | --- |
| configId | Channel ID. After the operations in **2.4.2 Enabling the Online Customer Service Function in the CEC** are complete, choose **Configuration Center** > **Access Configuration** > **Channel Configuration** to view the value. |
| accessKey | Developer ID. You are advised to use the value of **configId** for this parameter. |
| secretKey | Key for signature authentication. Choose **Configuration Center** > **Access Configuration** > **Channel Configuration**, click **New** or **Modify**, and find **Third-party authentication key** to configure the key. (Rule: The key contains a maximum of 43 characters, including only digits and letters.) |

NOTE

- For security purposes, **secretKey** must be obtained from the server and cannot be transferred on the frontend.
- **secretKey** is the unique key for signature authentication. Keep it secure.

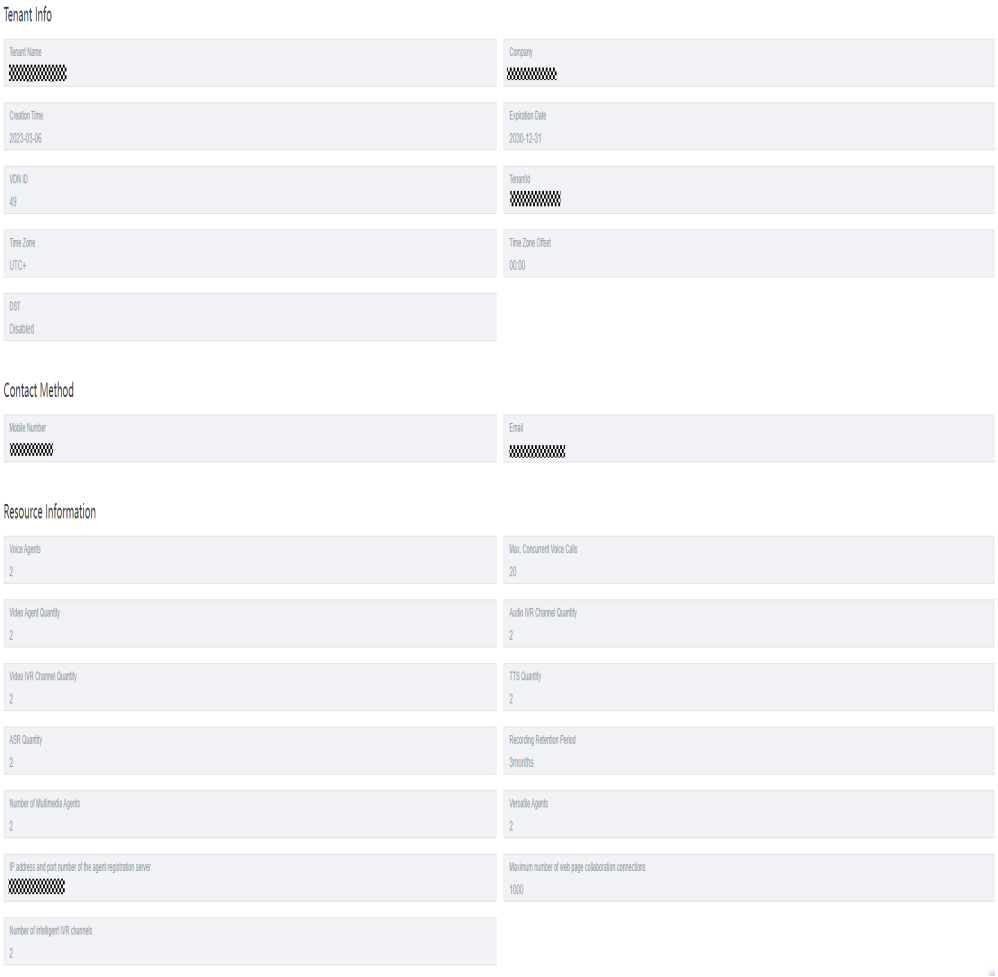
2. Perform the following steps to ensure that your tenant space has the multimedia agent feature:

   Sign in to the tenant space and choose **Configuration Center** > **System Management** > **Tenant Information**.

   Check the number of multimedia agents. If the number is 0, the multimedia agent feature is not enabled. Contact the Huawei operations administrator to enable the feature.

**Figure 2-3** Checking the multimedia agent feature




3. (Optional) To enable the co-browsing function (a customer can share the web page with an agent or mark content on the web page during an online text chat), contact the Huawei operations administrator.

## 2.4.2 Enabling the Online Customer Service Function in the CEC

**Step 1** Sign in to the CEC as a tenant administrator.

**Step 2** Add a multimedia skill queue.

Choose **Configuration Center** > **Employee Center** > **Skill Queue**. **Figure 2-4** shows the page for configuring a multimedia skill queue. If you need to enable the click-to-dial function, also configure a click-to-dial skill queue. **Figure 2-5** shows the page for configuring a click-to-dial skill queue.

**Step 3** Click **New** and set parameters based on **Table 2-3**.

**Figure 2-4** Page for configuring a multimedia skill queue

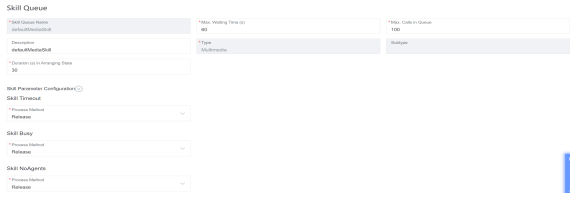



**Figure 2-5** (Optional) Page for configuring a click-to-dial skill queue

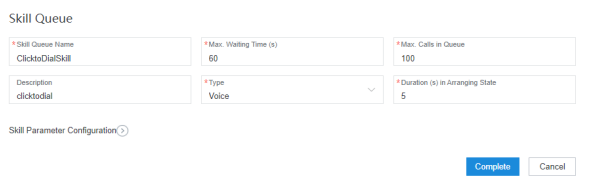



**Table 2-3** Parameters for configuring a skill queue

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Skill Queue Name | Yes | The value can contain a maximum of 20 characters and cannot contain spaces. |
| Max. Waiting Time (s) | Yes | The default value is **60**.<br>The unit is second.<br>The value ranges from 1 to 60000. |
| Max. Calls in Queue | Yes | The default value is **100**.<br>The value ranges from 1 to 10000. |
| Description | Yes | The value can contain a maximum of 50 characters. |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Type | Yes | The options are as follows:<br>● **Voice**: A voice skill queue handles voice businesses.<br>● **Multimedia**: A multimedia skill queue handles multimedia businesses.<br>● **Video**: A video skill queue handles video businesses.<br>● **Voice Click to Dial**: A voice click-to-dial skill queue is used together with multimedia businesses. During a text chat with an agent, a customer can directly make a voice call to the agent.<br>● **Video Click to Dial**: A video click-to-dial skill queue is used together with multimedia businesses. During a text chat with an agent, a customer can directly make a video call to the agent.<br>**NOTE**<br>Click-to-dial skill queues can be used only in the web channel.<br>The default value is **Voice**. |
| Duration (s) in Arranging State | Yes | Duration during which an agent is in wrap-up state after a call ends. The default value is **5**.<br>After this duration, the agent enters the idle state and can answer calls from customers.<br>The value ranges from 0 to 3600. |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Skill Parameter Configuration | No | Personalized configurations, which are the handling policies when a customer calls a skill queue and the call cannot be connected. The parameters are as follows:<br>● **Skill Timeout**<br>– **Process Method**: Handling policy when queuing times out because no idle agent can answer the call.<br>▪ **Release** (default)<br>▪ **Transfer**<br>– **Device Type**: If **Process Method** is set to **Transfer**, you need to configure the skill queue or IVR flow to which the call is transferred.<br>▪ **Skill Queue**<br>▪ **IVR**<br>● **Skill Busy**<br>– **Process Method**: Handling policy when the customer is queuing because no idle agent can answer the call, or the number of queuing customers exceeds the upper limit.<br>▪ **Release** (default)<br>▪ **Transfer**<br>– **Device Type**: If **Process Method** is set to **Transfer**, you need to configure the skill queue or IVR flow to which the call is transferred.<br>▪ **Skill Queue**<br>▪ **IVR**<br>● **Skill NoAgents**<br>– **Process Method**: Handling policy when no agent can answer the call because no agent is on duty.<br>▪ **Release** (default)<br>▪ **Transfer** |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| | | – **Device Type**: If **Process Method** is set to **Transfer**, you need to configure the skill queue or IVR flow to which the call is transferred.<br>▪ **Skill Queue**<br>▪ **IVR**<br>● **Queuing and waiting configuration**: When a customer needs to wait in a queue after making an inbound call, a voice can be played to optimize the customer waiting process.<br>– **Queuing Method**<br>▪ **Default Wait Tone**<br>▪ **Customizing the Wait Tone**<br>▪ **IVR**<br>● **Keeping and waiting configuration**: When a call needs to be held and the customer needs to wait, a voice can be played to optimize the customer waiting process.<br>– **Keeping Method**<br>▪ **Default Keeping Tone**<br>▪ **Customizing the Keeping Tone**<br>● **Skill AnswerMode**: After an agent answers a call from a customer, the employee ID of the agent can be played to the customer.<br>– **Answer Type**<br>▪ **Report employee ID**<br>▪ **Report no voice**<br>**NOTE**<br>Calls in voice, video, and click-to-dial skill queues can be transferred to IVRs or skill queues. Calls in multimedia skill queues can be transferred only to skill queues.<br>The skill queue to which a call is transferred and the skill queue that is created must belong to one type.<br>The waiting tones can be set only for voice and video skill queues. Click-to-dial skill queues are not supported. |

1. Click **Complete**.

**Step 4** Configure a multimedia called route.

1. Choose **Configuration Center** > **Access Configuration** > **Called Route**.
2. Click **New** to add parameter information for the VDN and click **Complete**, as shown in **Figure 2-6**. **Table 2-4** describes the parameters. If you need to enable the click-to-dial function, also configure a click-to-dial called route.

**Figure 2-6** Page for configuring a called route

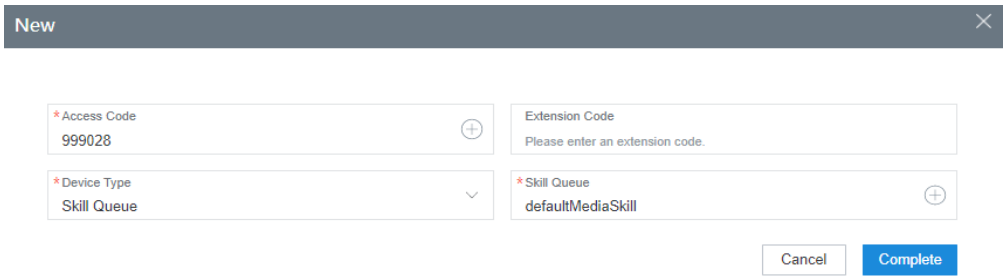



**Table 2-4** Parameters for configuring a called route

| Parameter | Mandatory or Not | Description |
|---|---|---|
| Access Code | Yes | Customer service hotline. Customers can dial the access code to connect to agents.<br>Click ⊕ to select an access code, for example, a multimedia access code, from the list in the dialog box. |
| Extension Code | No | To set one access code for multiple destination devices, you can configure extension codes.<br>For example, if the access code is 12345, you can add extension code 1 to route calls to skill queue A and extension code 2 to route calls to skill queue B. In this way, customers can dial 123451 to directly access skill queue A. |
| Device Type | Yes | Select **Skill Queue** to configure a called route of the skill queue type. |
| Skill Queue | Yes | Associate the skill queue created in **Step 3**.<br>**Skill Queue**: Click ⊕ to select a skill queue from the list in the dialog box. The type of the skill queue is the same as that of the access code. For example, if you select a multimedia access code, all available skill queues are of the multimedia type. |

**Step 5** Configure a business account and skill queue.

1. Choose **Configuration Center** > **Employee Center** > **Agent Management**.
2. Select an agent and click **Configure** in the **Operation** column. The agent information configuration page is displayed.
3. Associate a business account and skill queue with the agent. If you need to enable the click-to-dial function, also associate a click-to-dial skill queue.

**Figure 2-7** Page for configuring agent information

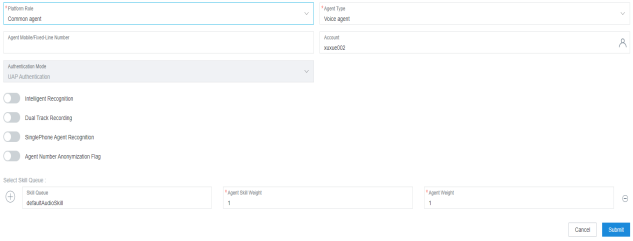


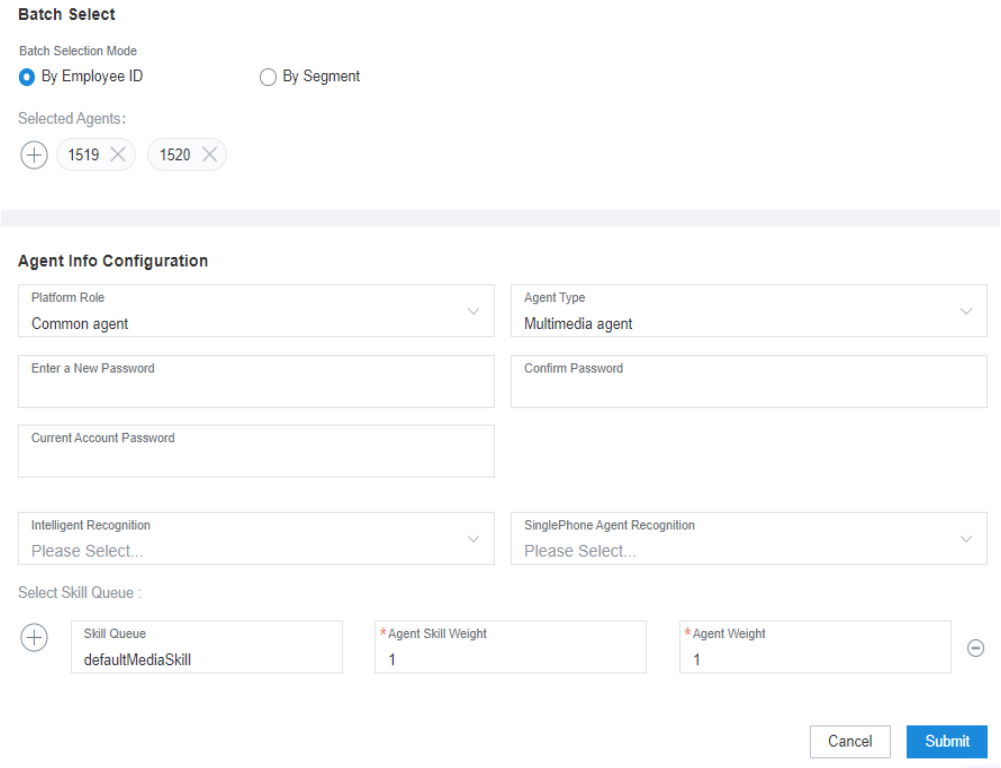

**Table 2-5** Parameters for configuring agent information

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Platform Role | Yes | Agent role. This parameter is mandatory.<br>– **Common agent**: This role can answer or transfer inbound calls from customers.<br>– **Quality checker**: This role can intervene in calls between common agents and customers. For example, this role can perform operations, such as insertion, interception, and forcible busy state setting, to coach and supervise agents' handling of inbound calls.<br>– **Callout agent**: This role can answer, transfer, or reject inbound calls from customers. |
| Agent Type | Yes | Agent type based on the businesses that can be handled. This parameter is mandatory.<br>– **Audio agent**<br>– **Video agent**<br>– **Multimedia agent**<br>– **Versatile agent** |
| Agent Mobile/Fixed-Line Number | No | Mobile number or fixed-line phone number used by an agent. |
| Account | Yes | Employee account.<br>For details, see **Managing Employees**. |

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| Intelligent Recognition | No | Whether an agent is an intelligent agent. By default, this switch is turned off. In addition to basic voice control functions, intelligent agents support real-time ASR and related intelligent recommendation functions. Before turning on this switch, ensure that the number of agents for which intelligent recognition is enabled does not exceed the number of intelligent agents allocated when the tenant is created. |
| SinglePhone Agent Recognition | No | After this switch is turned on, an agent can dial a specified access code to access an IVR flow, press a key as prompted to enter the employee ID and password to sign in, and answer calls on a mobile phone. When this switch is turned on, system O&M personnel need to customize the single-phone agent process for the tenant based on the platform, and the tenant needs to provide number resources for accessing the single-phone agent process. |
| Agent Number Anonymization Flag | No | Flag for a third party to mark whether an agent has the anonymization feature. This is not a feature switch. The anonymization feature enables agents to customize the calling number displayed on the user side (the calling number displayed to the user) and the calling number displayed on the agent side (the calling number displayed to the customer manager). |
| Select Skill Queue | Yes | Skill queue of an agent. When selecting multiple skill queues, set them to the same media type except for versatile agents. For example, set them to **Audio agent** or **Multimedia agent**.<br>**NOTE**<br>– If **Agent Type** is set to **Video agent**, the corresponding number of video agents must have been applied during tenant resource application.<br>– If **Agent Type** is set to **Multimedia agent**, the corresponding number of multimedia agents must have been applied during tenant resource application.<br>– If **Agent Type** is set to **Versatile agent**, the corresponding number of versatile agents must have been applied during tenant resource application.<br>– To add more business accounts, choose **Configuration Center** > **Employee Center** > **Employee**. |

4. Click **Submit**. The business account and skill queue are associated with the agent ID.
5. (Optional) Click **Batch Configure**. On the agent information configuration page that is displayed, configure agent information in batches.

**Figure 2-8** Page for configuring agent information in batches




- **Batch Select**: Select agents to be configured by employee ID or employee ID segment.
- **Agent Info Configuration**: Set parameters by referring to **5**.

**Step 6** Configure the web channel.

1. Choose **Configuration Center** > **Access Configuration** > **Channel Configuration**.
2. Click **New**, set **Channel Access Code**, select **WEB**, and click **Next** to enter the page for configuring the web channel.

**NOTE**

The channel access code must be unique. The code can contain only letters, digits, and underscores (_), and must start with a letter or an underscore (_).

3. Set the web channel parameters shown in **Figure 2-9** based on **Table 2-6**. If you need to enable the click-to-dial function, configure **CTD Called Party Configuration** and **Click-to-Call Skill Queue** on this page.

**Figure 2-9** Web channel configuration page

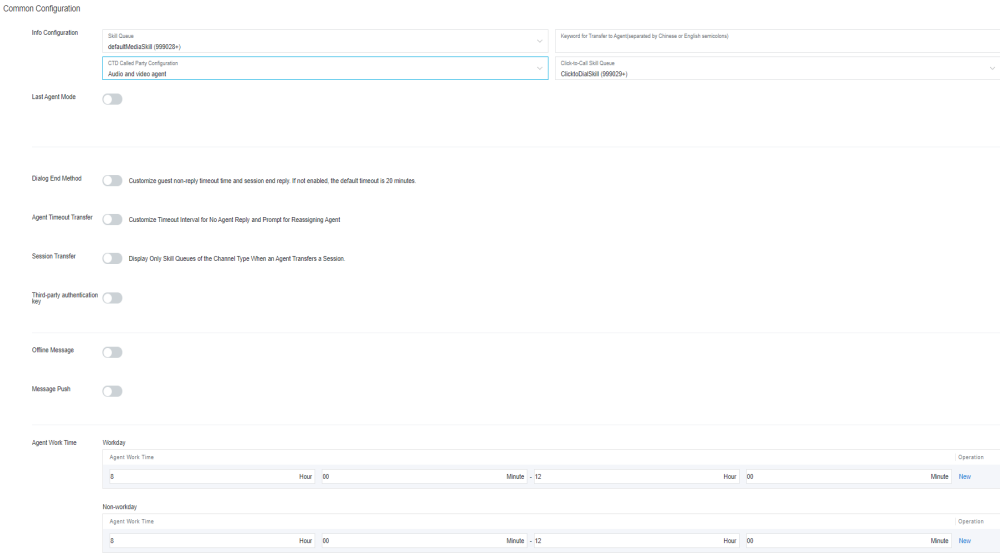



**Table 2-6** Web channel parameters

| Parameter | Description |
| --- | --- |
| **Common Configuration** | |
| Info Configuration | Set the following parameters:<br>– **Skill Queue**: The options are all multimedia called parties of the current tenant space. Select an option as required.<br>– **Keyword for Transfer to Agent**: Keywords for switching from chatbot service to manual service. After a customer enters any of the keywords on the HTML5 client, chatbot service is switched to manual service.<br>**NOTE**<br>If the intelligent chatbot is disabled, you do not need to set this parameter.<br>– **CTD Called Party Configuration**: The options are all voice and video agents and IVRs of the current tenant space.<br>– **Click-to-Call Skill Queue**: If **CTD Called Party Configuration** is set to **Audio and video agent**, you need to configure the related skill queue.<br>– **Click to obtain the IVR access code**: If **CTD Called Party Configuration** is set to **IVR**, you need to configure the IVR access code. |

| Parameter | Description |
| --- | --- |
| Dialog End Method | This function is disabled by default. If it is disabled, the default timeout period is 20 minutes. If it is enabled, set the following parameters:<br>– **Prompt Interval for No Reply (min)**: If a customer does not reply on the client within this period, the session is disconnected.<br>– **Conclusion**: The system sends an end reminder after the session is disconnected. |
| Third-party authentication key | This function is disabled by default. If it is enabled, set **secretKey**. This parameter is mandatory when the authorization signature authentication mode is used. |
| Agent Work Time | – **Workday**: A maximum of four working time segments (from 00:00 to 24:00) can be configured. By default, a time segment is displayed. You can click **New** to add a time segment.<br>– **Non-workday**: A maximum of four working time segments (from 00:00 to 24:00) can be configured. By default, a time segment is displayed. You can click **New** to add a time segment.<br>– **Non-Working Time Notification**: Notification displayed when a customer calls in non-working time. |
| Queue reminder | Set the following parameters:<br>– **Queue reminder interval (seconds)**: This parameter is mandatory. The default value is **10**.<br>– **Queue reminder content**: Notification displayed if a customer is in a queue after making an inbound call. |

| Parameter | Description |
| --- | --- |
| Chatbot Configuration | This function is disabled by default. If it is enabled, customers are preferentially connected to the chatbot. Set the following parameters:<br>– **Change Avatar**: Chatbot avatar.<br>– **Name**: Chatbot name.<br>– **Gender**: Chatbot gender.<br>– **Chatbot Access Code**: Chatbot access code configured in the intelligent IVR.<br>– **Default reply**: Customized reply displayed when the chatbot cannot recognize the intent of a customer.<br>– **Timeout reply**: Customized reply displayed when the session with a customer times out.<br>– **Prompt for transfer to agent**: Customized prompt message indicating that chatbot service is switched to manual service.<br>**NOTE**<br>If you want to enable the chatbot function to allow customers to chat with the chatbot, configure the chatbot based on the required chatbot type. For details, see **Configuring Intelligent IVR**. |
| Robot Assistant Configuration | This function is disabled by default. If it is enabled, the chatbot assistant is enabled on the agent side.<br>– **Assistant Access Code**: Chatbot access code configured in the intelligent IVR.<br>– **SilentAgent Skill Queue**: The options are all multimedia called parties of the current tenant space. Select an option as required. |

4. Click **Save And Proceed To The Next Step**. The **Integration instructions** page is displayed.

**Step 7** (Optional) On the **Integration instructions** tab page, click **Try**. On the page that is displayed, set customer information to simulate the dialog window on the customer side.

Verify that customers can chat with agents or the chatbot through the current channel.

1. Click **Try**, and click  in the lower right corner of the page that is displayed. The **Online Customer Service** dialog box is displayed. The online customer service has two chat modes:

   a. If **Connect to Chatbot** is enabled, customers are connected to and chat with the chatbot by default. If the chat content entered by a customer contains a keyword that can be recognized by the chatbot, the chatbot recognizes the keyword and replies to the customer.

b. If **Connect to Chatbot** is disabled, customers are automatically connected to and chat with online agents. Click **Sign In** on the CEC to sign in as a multimedia agent, and choose **Online Chat Workbench**. The workbench of the current session is displayed. After a customer is connected, you can chat with the customer online.

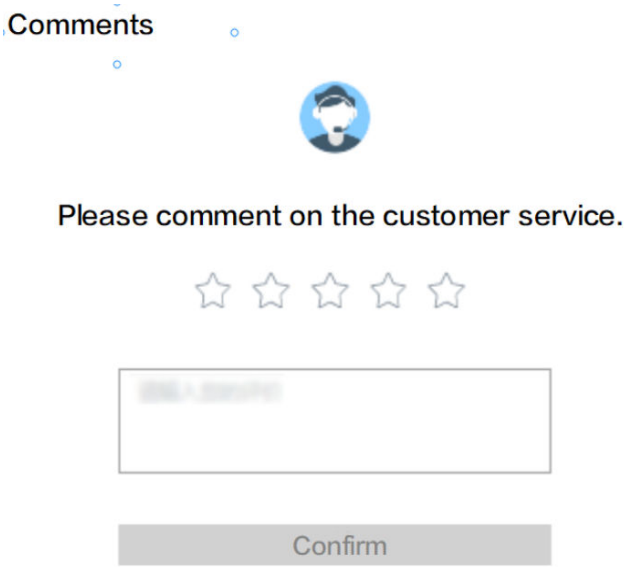
NOTE

When chatting with the chatbot, the customer can click **Transfer to Agent** or enter content that contains any of the keywords specified by **Keyword for Transfer to Agent** to switch from the chatbot to an agent. (**Keyword for Transfer to Agent** can be set in **Step 6**.) However, the customer cannot switch to the chatbot or another agent when chatting with an agent.

2. In the **Online Customer Service** dialog box, enter the chat content, click **Send**, and check the reply of the chatbot or agent.

3. (Optional) Click  and choose **Evaluation** to comment on the customer service, including the satisfaction rating and evaluation content, as shown in **Figure 2-10**. Click **Confirm** to submit the evaluation.

**Figure 2-10** Comments page




NOTE

Customers can evaluate the agents who chat with them. During or after a chat, the customer can evaluate the service of the agent at any time. The last evaluation is used.

**----End**

After the channel configuration is complete, record the channel ID, which will be used during token authentication API development and page integration.

| Configuration ID | Channel Access Code | Channel Type | Bind Skill Queue | Operation |
|---|---|---|---|---|
| 869525871671777281 | WECHAT001 | WECHAT | defaultMediaSkill (10001+) | Modify Delete |
| 853180664420335631 | wechat001 | WECHAT | defaultMediaSkill (10001+) | Modify Delete |
| 827635713993441281 | hyy | WECHAT | defaultMediaSkill (10001+) | Modify Delete |
| 817854979183314945 | WEBCHAT | WEB | defaultMediaSkill (10001+) | Modify Delete |
| 743032024951966724 | A20191231 | WECHAT | defaultMediaSkill (10001+) | Modify Delete |
| 740930049837911041 | a123456 | WECHAT | defaultMediaSkill (10001+) | Modify Delete |

# 2.5 Integration Development

## 2.5.1 Developing an Authorization Signature Generation Mechanism

You need to develop an authentication mechanism for generating an authorization signature. The CEC needs to use the generated authorization signature for authentication.

Before the development, you need to learn about the internal rules for generating an authorization signature, as shown in **Figure 2-11**.

**Figure 2-11** Main rules of the authorization signature generation mechanism


After learning about the preceding principles and rules, perform the following steps to develop the authorization signature authentication mechanism:

**Step 1** Generate **SignedHeaders** by traversing the header names involved in encoding in the HTTP header.

1. Change the header names to lowercase, that is, invoke the lowerCase() function. For details about the tool class SignerUtils, see 1. The following is an example:

```
private Map<String, String> lowerCaseSignedHeaders(Map<String, String> signedHeaders) {
    if ((null == signedHeaders) || signedHeaders.isEmpty()) {
        throw new IllegalArgumentException("signedHeaders cann't be null.");
    }
    Map<String, String> headers = new HashMap<>(SignerUtils.HASH_MAP_INITIALIZATION_SIZE);
    for (Map.Entry<String, String> e : signedHeaders.entrySet()) {
        String name = e.getKey();
        String value = e.getValue();
        headers.put(name.toLowerCase(Locale.ENGLISH), value.trim());
    }
    return headers;
}
```

2. Use semicolons (;) to separate the header names in 1-1 to generate a record. Do not add a semicolon (;) to the last field.
3. Sort all records in 1-1 in alphabetical order and combine them into a large string in sequence. The following is an example:

```
private String appendSignedHeaders(StringBuilder buffer) {
    int start = buffer.length();
     Set<String> headerNames = new TreeSet<>(this.signedHeaders.keySet());
    for (String name : headerNames) {
        buffer.append(name).append(';');
    }
    buffer.deleteCharAt(buffer.length() - 1);
     int end = buffer.length();
    String signedHeadersStr = buffer.substring(start, end);
    return signedHeadersStr;
}
```

**NOTE**

Encode the following header:

**Content-Length="*****"**

**Content-Type="application/json;charset=UTF-8"**

**Step 2** Generate **authStringPrefix**.

Combine the following fields with slashes (/): **authVersion**, **accessKey**, **timestamp**, and **SignedHeaders**. The format is as follows:

```
authStringPrefix="auth-v2/{accessKey}/{timestamp}/{SignedHeaders}";
```

**NOTE**

- **auth-v2**: Authentication version number. In this version, the value is fixed to **auth-v2**.
- **accessKey**: The third-party system uses **configId** (channel ID) as the unique ID.
- **timestamp**: Time when the third-party system initiates a service. The value is a string. The time string is formatted to the *yyyy-MM-dd'T'HH:mm:ss.SSS'Z* format.
- **SignedHeaders**: Header names involved in encoding in the HTTP header, which is generated in Step 1.

**Step 3** Generate **signingKey**.

Encrypt the value of **authStringPrefix** generated in Step 2 using SHA256HEX. **SecretKey** is the key configured by the third-party system on the channel configuration page. The following is an example of the SHA256HEX algorithm. For details about the tool class SignerUtils, see 1.

```
public static String sha256Hex(String key, String toSigned) throws
NoSuchAlgorithmException,InvalidKeyException, UnsupportedEncodingException {
```

```
    Mac mac = Mac.getInstance("HmacSHA256");
    mac.init(new SecretKeySpec(key.getBytes(SignerUtils.CHARSET), "HmacSHA256"));
    String digit = new String(SignerUtils.encodeHex(mac.doFinal(toSigned.getBytes(SignerUtils.CHARSET))));
    return digit;
}
```

**Step 4** Generate **CanonicalHeaders**.

NOTE

The encoding rule is the same as that of **SignedHeaders**, but header value encoding is added.

1. Traverse the header names involved in encoding in the HTTP header and change the header names to lowercase. That is, invoke the lowerCase() function. For details, see **Step 1**.
2. Invoke the normalize function to format the converted lowercase string. For details about the tool class PathUtils, see **2**.

```
/**
 * normalize
 * @param value Payload information
 * @return builder
 */
public static String normalize(String value) {
    try {
        StringBuilder builder = new StringBuilder(PathUtils.DEFAULT_CAPACIT);
        for (byte b : value.getBytes(PathUtils.CHARSET)) {
            if (PathUtils.URI_UNRESERVED_CHARACTERS.get(b & 0xFF)) {
                builder.append((char) b);
            } else {
                builder.append(PathUtils.PERCENT_ENCODED_STRINGS[b & 0xFF]);
            }
        }
        return builder.toString();
    } catch (UnsupportedEncodingException e) {
        throw new RuntimeException(e);
    }
}
```

3. Sort the records in **4-1** in alphabetical order.
4. Traverse the sorted records and add the string "\n" between the records to form a large string. Do not add the string "\"n to the last record.

**Step 5** Generate **canonicalRequest**.

Combine the **HttpMethod**, **HttpURI**, **SignedHeaders**, **CanonicalHeaders**, and **NormalizePath** fields with **"\n"**. Do not add **"\n"** to the last record. For details about the tool class PathUtils, see **2**.

```
private String canonicalRequest() {
    StringBuilder buffer = new StringBuilder(PathUtils.DEFAULT_CAPACITY);
    buffer.append(this.httpMethod).append(System.lineSeparator());
    buffer.append(this.uri).append(System.lineSeparator());
      this.appendSignedHeaders(buffer);
    buffer.append(System.lineSeparator());
      this.appendCanonicalHeaders(buffer);
    buffer.append(System.lineSeparator());
     if (this.isNotEmpty(this.payload))
     {
        buffer.append(PathUtils.normalize(this.payload));
     }
     return buffer.toString();
 }
```

The format is as follows:

```
CanonicalRequest = $HttpMethod + "\n" + $HttpURI+ "\n" + SignedHeaders($HttpHeaders) + "\n" +
CanonicalHeaders ($HttpHeaders) + "\n" + NormalizePath($HttpBody)
```

**NOTE**

- The **CanonicalRequest** parameter is described as follows:

  **$HttpMethod**: GET, PUT, and POST requests defined in the HTTPS protocol. The value must be all in uppercase.

  **$HttpURI**: API request URI. The value must start with a slash (/). If the value does not start with a slash (/), add it. An example value is **/service-cloud/webclient/chat_client/js/newThirdPartyClient.js**. The value **/** indicates an empty path.

  **SignedHeaders**: **SignedHeaders** generated in **Step 1**.

  **CanonicalHeaders**: **CanonicalHeaders** generated in **Step 4**.

  **NormalizePath**: Body after formatting.

- Only the following parameters in **NormalizePath** are encoded:

  **thirdUserName**: Enterprise username.

  **thirdUserId**: Enterprise user ID.

  **tenantSpaceId**: Tenant space ID provided by the enterprise.

  **channelConfigId**: Enterprise access channel ID.

**Step 6** Generate **signature**. Encrypt **CanonicalRequest** generated in **Step 5** using SHA256HEX. The encryption key is **signingKey** generated in **Step 3**.

**Step 7** Generate **Authorization** (signature). Combine **authStringPrefix** generated in **Step 2** and **signature** generated in **Step 6** with a slash (/). The format is as follows:

```
Authorization:$authStringPrefix/$Signature
```

**----End**

## Reference

The authentication mechanism for generating an authorization signature involves the tool classes SignerUtils and PathUtils. Their formats are as follows:

1. SignerUtils

```
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;

public class SignerUtils {
    private static final int HASH_MAP_INITIALIZATION_SIZE = 5;
    private static final int ONE_CHAR_BITS_NUM = 4;
    private static final String CHARSET = "UTF-8";
    private static final char[] DIGITS_LOWERS = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a',
        'b', 'c', 'd', 'e', 'f'};
    private SignerUtils() {
    }

    private static char[] encodeHex(final byte[] data) {
        final int le = data.length;
        final char[] outs = new char[le << 1];
        for (int i = 0, j = 0; i < le; i++) {
            outs[j++] = SignerUtils.DIGITS_LOWERS[(0xF0 & data[i]) >>> ONE_CHAR_BITS_NUM];
            outs[j++] = SignerUtils.DIGITS_LOWERS[0x0F & data[i]];
        }
        return outs;
    }
}
```

2. PathUtils

```
import java.io.UnsupportedEncodingException;
import java.util.BitSet;
import java.util.Locale;
import java.util.concurrent.CompletionException;

public class PathUtils {
    private static final String CHARSET = "UTF-8";
    private static final int NUM_256 = 256;
    private static final int DEFAULT_CAPACITY = 16;
    private static final BitSet URI_UNRESERVED_CHARACTERS = new BitSet();
    private static final String[] PERCENT_ENCODED_STRINGS = new String[NUM_256];
    static {
        for (int i = 97; i <= 122; i++) {
            PathUtils.URI_UNRESERVED_CHARACTERS.set(i);
        }
        for (int i = 65; i <= 90; i++) {
            PathUtils.URI_UNRESERVED_CHARACTERS.set(i);
        }
        for (int i = 48; i <= 57; i++) {
            PathUtils.URI_UNRESERVED_CHARACTERS.set(i);
        }
        PathUtils.URI_UNRESERVED_CHARACTERS.set(45);
        PathUtils.URI_UNRESERVED_CHARACTERS.set(46);
        PathUtils.URI_UNRESERVED_CHARACTERS.set(95);
        PathUtils.URI_UNRESERVED_CHARACTERS.set(126);

        for (int i = 0; i < PathUtils.PERCENT_ENCODED_STRINGS.length; i++) {
            PathUtils.PERCENT_ENCODED_STRINGS[i] = String.format(Locale.ROOT, "%%%02X", new
Object[]{Integer.valueOf(i)});
        }
    }
    private PathUtils() {}
}
```

## 2.5.2 Developing an Integration Page

### 2.5.2.1 Integrating Core Code

**Step 1** Introduce the required JavaScript framework. In the following example, the jQuery component needs to be introduced. The reference version is jQuery 3.4.1. If the co-browsing function is required, introduce the cobrowse.js and cobrowseCommon.js components or release the cobrowse plugin for use. (For details about the development of the plugin, see section 3.1 "Guide to the Development and Usage of the cobrowse Plugin.") Introduce the cobrowse.js and cobrowseCommon.js components in CDN mode. Replace *ip:port* with the IP address and port number of the CEC, or directly with the domain name of theCEC, for example, **https://www.test.com/service-cloud/resource.root/cobrowse/sdk/cobrowse.js**. Figure 2-12 shows an example of the code.

**Figure 2-12** JavaScript framework reference example

```
<script type="text/javascript" src="https://ip:port/service-cloud/resource.root/cobrowse/sdk/cobrowse.js"></script>
<script type="text/javascript" src="https://ip:port//service-cloud/webclient/chat_client/js/cobrowseCommon.js"></script>
<script type="text/javascript" src="js/jquery.js"></script>
```

**NOTE**

1. Co-browsing operations cannot be performed on a new tab page.
2. When the cobrowse.js and cobrowseCommon.js components for co-browsing are introduced and the third-party page contains nested iFrames, co-browsing operations can be performed only in the iFrame where the JavaScript components are introduced. If an iFrame has a parent iFrame, co-browsing operations cannot be performed in the parent iFrame.

**Step 2** Generate the authorization signature on the server by following the instructions provided in **2.5.1 Developing an Authorization Signature Generation Mechanism**.

**Step 3** On the frontend of the enterprise system, send an Ajax request to obtain the authorization signature generated on the server.

```
function testAjax() {
    let timestamp = new Date().getTime();
    let serviceUrl = "authorizationService";
    let thirdUserData = {};
    thirdUserData['thirdUserName'] = userName;
    thirdUserData['thirdUserId'] = userId;
    thirdUserData['tenantSpaceId'] = tenantSpaceId;
    thirdUserData['channelConfigId'] = configId;
    thirdUserData['locale'] = locale;
    thirdUserData['mapService'] = tencent;
    thirdUserData['timestamp'] = timestamp;
    var request = $.ajax({
        url: serviceUrl,
        method: "POST",
        contentType: "application/json",
        data: JSON.stringify(thirdUserData),
    });
    request.done(function(message){
        if(message){
            thirdUserData['thirdPartAuthorization'] = message;
            thirdValidate(thirdUserData);
        }
        return  message;
    });
```

**Step 4** After obtaining the signature, use the Ajax to request the JavaScript script to be integrated.

```
const $ContextPath = "https://ip:port/service-cloud";
function thirdValidate(thirdUserData) {
  let timestamp = new Date().getTime();
  let serviceUrl = $ContextPath + "/webclient/chat_client/js/newThirdPartyClient.js?"+"&t=" + timestamp;
      var request = $.ajax({
        url: serviceUrl,
        method: "POST",
        data: JSON.stringify( ),
        crossDomain: true,
        dataType:"text",
        xhrFields: {withCredentials: true},
        error: function (XMLHttpRequest, textStatus, errorThrown)
            { console.log(XMLHttpRequest.status); console.log(XMLHttpRequest.readyState);
console.log(textStatus); },
        success: function (data) {
            importScript(data)
        }
});
```

**Table 2-7** Parameters for developing an integration page

| Parameter | Mandatory or Not | Description |
| --- | --- | --- |
| $ContextPath | Yes | Actual domain name. Replace **https://***ip:port/* with the public network domain name of the CEC. |
| thirdPartAuthorization | Yes | Authorization signature generated by the enterprise, which must be the same as the signature in **Step 7**. |
| tenantSpaceId | Yes | Tenant space ID. Choose **Configuration Center** > **System Management** > **Tenant Information** to view the value. |
| thirdUserId | Yes | Enterprise user ID. The value must be the same as that of **thirdUserId** in **Step 5**. |
| thirdUserName | Yes | Enterprise username. The value must be the same as that of **thirdUserName** in **Step 5**. |
| channelConfigId | Yes | Channel ID. After the operations in **Step 6** are complete, choose **Configuration Center** > **Access Configuration** > **Channel Configuration** to view the value. |
| locale | Yes | Language information of third-party tenant space.<br>● **zh**: Chinese<br>● **en**: English |
| timestamp | Yes | Timestamp. The time string is formatted to the *yyyy-MM-dd'T'HH:mm:ss.SSS'Z* format. |
| mapService | No | User map service.<br>● **tencent**<br>● **google**<br>The default value is **tencent**.<br>**NOTE**<br>Tencent Maps does not support locations outside China. |

**----End**

### 2.5.2.2 Example: JavaScript Page Integration Code

| | |
| --- | --- |
| **Environment Requirement** | - |
| **Reference Library** | jquery.js |

| Download Link | index.html |
| --- | --- |

**NOTICE**

- The demo described in this document may involve the use of personal data. You are advised to comply with relevant laws and regulations and take sufficient measures to ensure that personal data is fully protected.
- The demo described in this document is for demonstration only. Commercial use of the demo is prohibited.
- Information in this document is for reference only and does not constitute any offer or commitment.

## index.html

```
<!DOCTYPE html>
<html>

<head>
    <!-- If the co-browsing function is required, introduce the cobrowse.js and cobrowseCommon.js
components. Replace ip:port with the IP address and port number of the CEC, or directly with the domain
name of the CEC. -->
    <script type="text/javascript" src="https://ip:port/service-cloud/resource.root/cobrowse/sdk/
cobrowse.js"></script>
    <script type="text/javascript" src="https://ip:port//service-cloud/webclient/chat_client/js/
cobrowseCommon.js"></script>
    <!-- 0. Introduce the jQuery component. -->
    <script type="text/javascript" src="jquery.js"></script>
</head>

<body>
    <script>
    // 1. Define variables, which will be used on the chat box page. For details about the parameters, see
Table 2-7.
        const userId = "XXXXXXXX";                  // user ID (value of thirdUserId in the table)
        const userName = "XXXXXXXXXXX";                // user nickname (value of thirdUserName in the table)
        const tenantSpaceId ="XXXXXXXXXX";                // tenant space ID (tenant id)
        const configId = "XXXXXXXXXXXXXXXXXX";            // channel ID (wWEBchannelConfig id).
        const locale = "zh";                        // language (locale, which can be zh or en)
        const $ContextPath = "https://ip:port/service-cloud";        // request URL

    // 2. Construct request parameters.
        let serviceUrl = $ContextPath + "/webclient/chat_client/js/newThirdPartyClient.js?&t="+new
Date().getTime();
        let thirdUserData = {};
        thirdUserData['thirdUserName'] = userName;
        thirdUserData['thirdUserId'] = userId;
        thirdUserData['tenantSpaceId'] = tenantSpaceId;
        thirdUserData['channelConfigId'] = configId;
        thirdUserData['locale'] = locale;
    thirdUserData['timestamp'] = new Date().getTime();                              // timestamp
    thirdUserData['secretKey'] = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';          // authentication key
        getAuthorization(thirdUserData);                                          // authorization signature

        // 3. Request newThirdPartyClient.js.
        var request = $.ajax({
            url: serviceUrl,
            method: "POST",
            data: JSON.stringify(thirdUserData),
            xhrFields: { withCredentials: true },
            success: function (data) {
```

```
            // 4. Create a script tag and run the newThirdPartyClient.js script.
            importScript(data);
        },
        error: function (XMLHttpRequest, textStatus) {
            alert("unauthorized,validate failed")
        }
    });

// 5. Load and run the JavaScript on the integration page.
    var importScript = (function (oHead) {
        function loadError(oError) {
            throw new URIError("The script " + oError.target.src + " is not accessible.");
        }
        return function (sSrc, fOnload) {
            var oScript = document.createElement("script");
            oScript.type = "text/javascript";
            oScript.onerror = loadError;
            if (fOnload) { oScript.onload = fOnload; }
            oHead.appendChild(oScript);
            oScript.innerHTML = sSrc;
        }

    })(document.head || document.getElementsByTagName("body")[0]);
// Obtain the authorization signature based on parameters.
function getAuthorization(thirdUserData){
    $.ajax({
        url:"/webchat/authorizationService",
        method:"post",
        data:JSON.stringify(thirdUserData),
        async:false,
        success: function (data) {
            thirdUserData['thirdPartAuthorization'] = data;
            }
        });
    }
</script>
</body>

</html>
```

## 2.6 Test and Verification

After page integration is complete, you need to test and verify that the web chat control can be used properly. The following uses Google Chrome as an example. During the verification, you can use the Nginx server to simulate a third party to invoke the web chat control or use other access methods that you are familiar with.

**Step 1** Enable the Nginx server locally (for details about the Nginx version, see **nginx/Windows-1.22.0**) and configure information, including the service address and certificate, in the **nginx.conf** file.

**Step 2** Enter the server address and send a request to simulate a third party to invoke the web chat control.

**Step 3** Sign in to the CEC as an agent in the multimedia skill queue, select **Sign In** on the top of the page, and set the state to **Idle**.

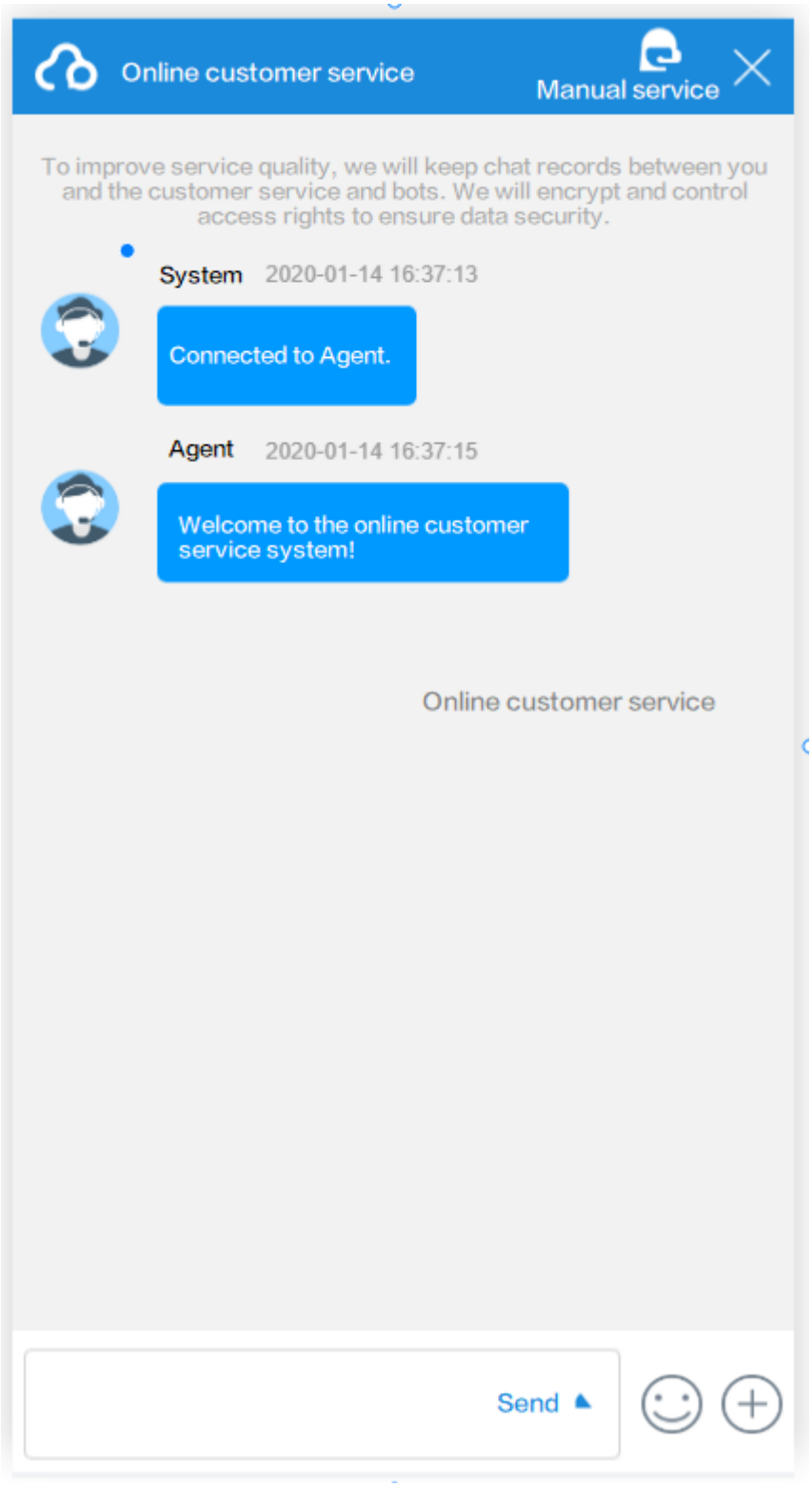
**Step 4** On the enterprise customer page, press **F12** to open the console, choose **Network**, and refresh the page.

Click the **thirdPartyClient.js** request displayed on the console and click **Response** on the right. If any content is returned, a blue circle icon is displayed in the lower

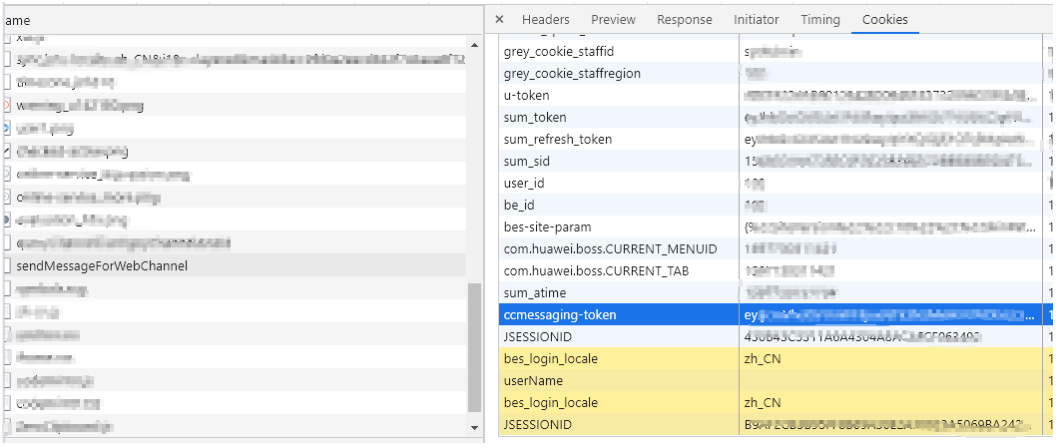
right corner. Click the icon. If a message indicating that an agent is connected is displayed, the invocation is successful.

**Figure 2-13** Online customer service



**Step 5** On the browser console, choose **Application**, choose **Storage** > **Cookies** > *Your domain name* on the left, and check whether **ccmessaging-token** is written into cookies.

**Figure 2-14** Checking ccmessaging-token

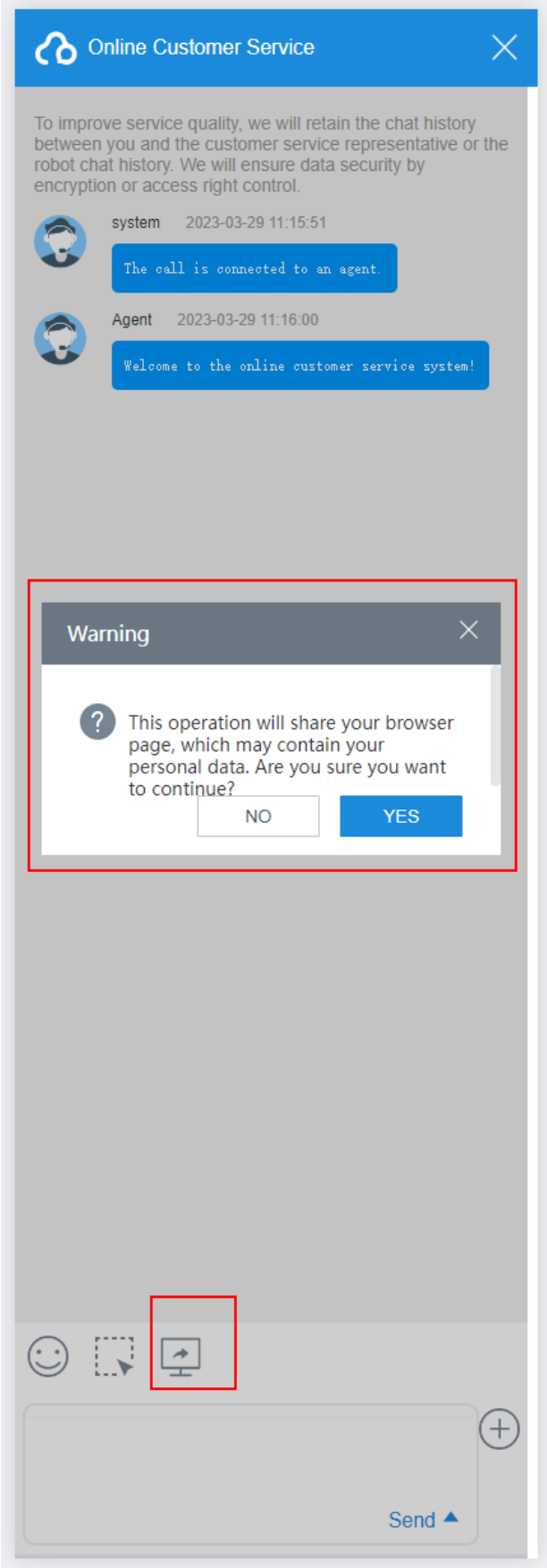



**Step 6** On the agent page of the CEC, check whether the agent is in the occupied state and receives the request sent by the client.

**Step 7** If the co-browsing feature is enabled for the tenant, a customer can initiate co-browsing in the dialog box on the customer side. After the customer clicks the co-browsing button, a confirmation dialog box is displayed. The customer can confirm the initiation.

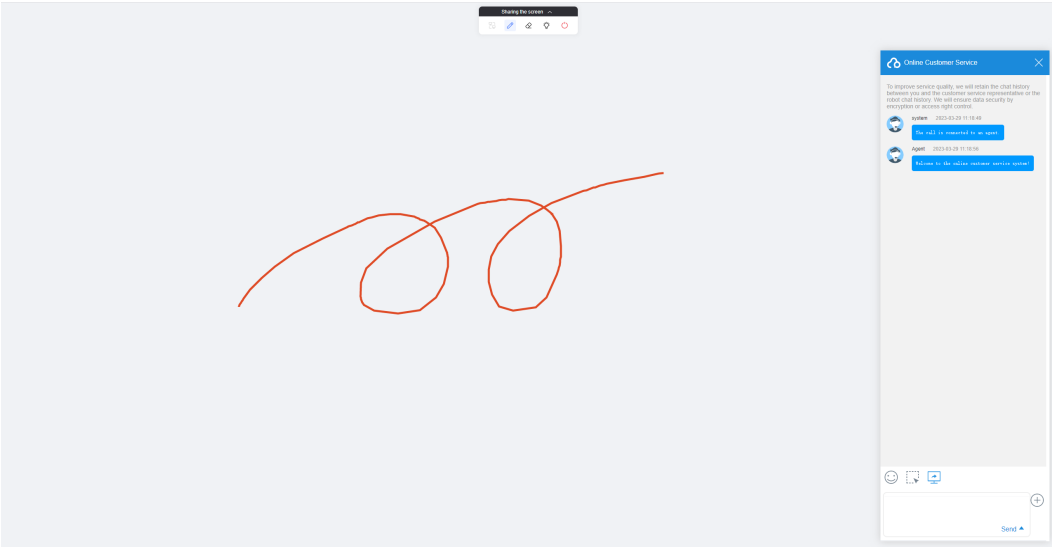
  

**Figure 2-15** Dialog box on the customer side



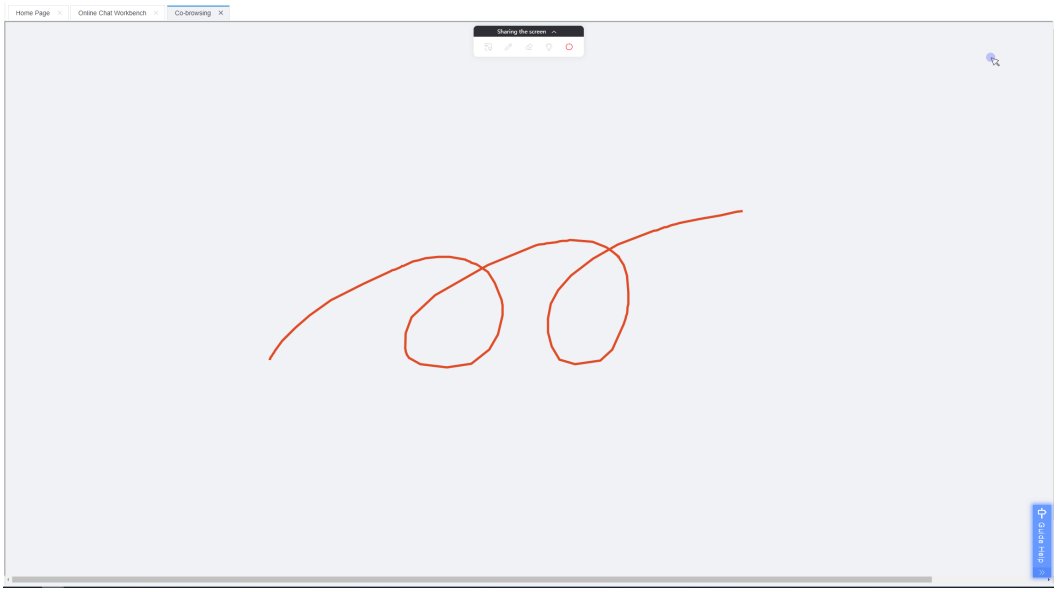
**Step 8** After the agent accepts the co-browsing request on the agent workbench, the customer can share the current page, mark or highlight content on the page, or request the agent's remote control of the page. When the co-browsing icon shown in **Figure 2-16** is displayed in the lower right corner of the page, co-browsing is successfully initiated.

**Figure 2-16** Co-browsing page on the customer side




**Step 9** On the agent workbench, the agent can view the customer's current page, view the content marked or highlighted by the customer, or request the remote control of the customer's page.

**Figure 2-17** Co-browsing page on the agent workbench




**Step 10** When the web page is scrolled, the marker cannot move with the scrolling of the web page, as shown in **Figure 2-18** and **Figure 2-19**. If the marker needs to move with the scrolling of the web page in a business scenario, clear the marker, scroll the page, and mark content again.

**Figure 2-18** Circled web page content

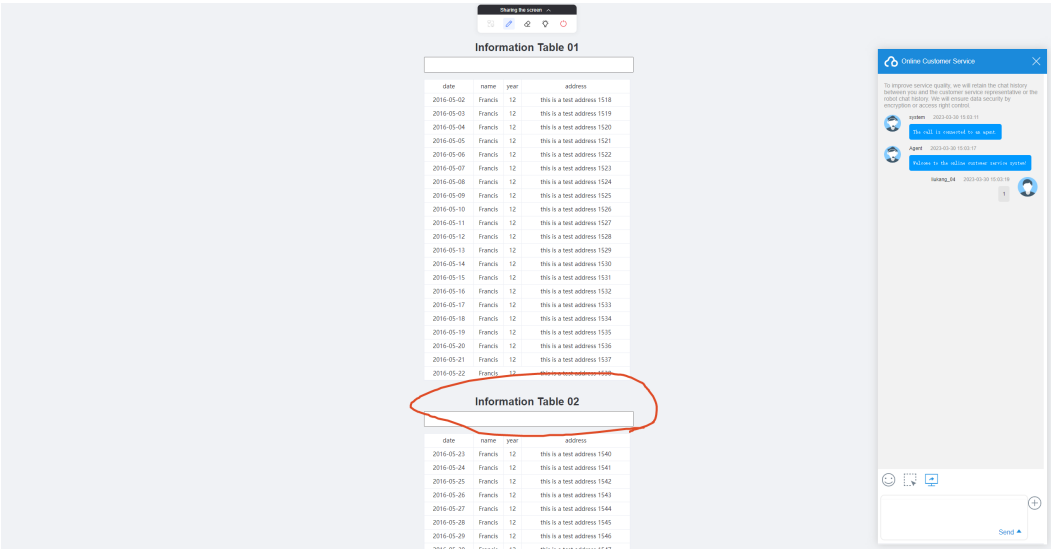
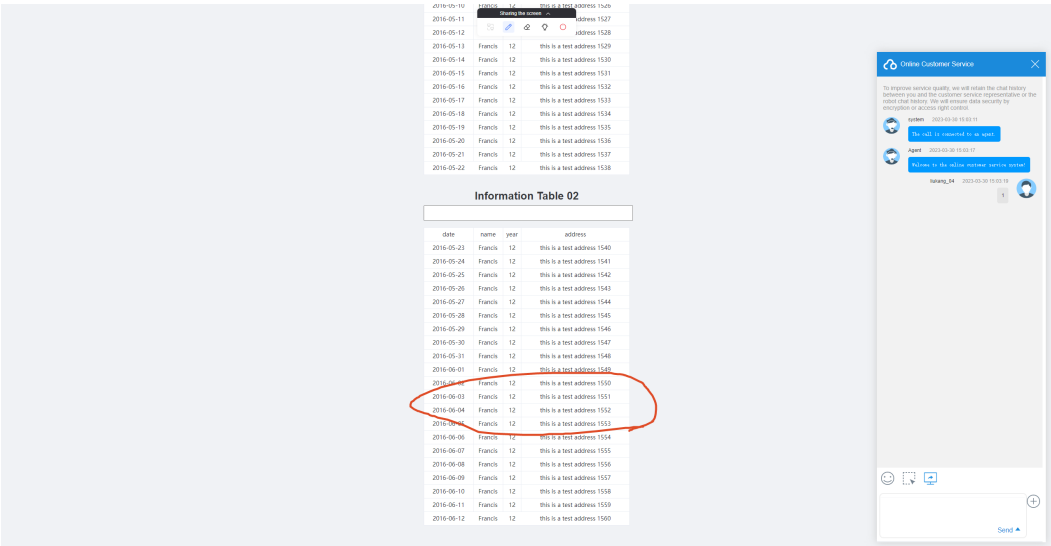



**Figure 2-19** Circle that cannot move with the scrolling of the web page




**Step 11** If the third-party web page contains an animation effect, the agent workbench cannot display the animation effect during the co-browsing.

**NOTE**

Ensure that only one agent in the multimedia skill queue signs in to your tenant space. Otherwise, the system may route the session to another agent based on the routing rules. As a result, you may not receive the customer request.

**----End**

# 2.7 FAQs

## 2.7.1 How Can I Resolve the Reported Cross-domain Error When the xmlHttpRequest Requests the URL of the CEC?

### Symptom

The following error information is displayed:

```
Access to XMLHttpRequest at "requested js" from origin xx has been blocked by CROS policy: No 'Access-Control-Allow—Origin' header is present on the requested response;
```

### Solution

This is a cross-domain error: The website of the integrator does not allow requests for resources that are not provided by the local domain due to security restrictions. You can use the reverse proxy of a load balancing application (such as Nginx).

**Figure 2-20** Principles of address mapping on a load balancing application


When a third-party page uses JavaScript to invoke a service under the local domain name, the service is bypassed to the domain name of the CEC.

The CEC identifies the request only when it identifies **service-cloud**. Therefore, the request URL of the third-party page must contain **service-cloud**.

The following uses Nginx as an example to describe the overall configuration:

**Step 1** Add the first-layer service address to the **nginx.conf** file as follows:

```
location /demo/ {
    proxy_set_header Host $host;
    set $Real $proxy_add_x_forwarded_for;
    if ($Real ~ (\d+)\.(\d+)\.(\d+)\.(\d+)\,(.*)){
        set $Real $1.$2.$3.$4;
    }
    proxy_set_header X-Real-IP $Real;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    proxy_pass https://servicestage.besclouds.com/;
}
```

**NOTE**

- *demo* is an example value. The integrator can replace the value based on requirements.
- Replace *servicestage.besclouds.com* with the address provided by the CEC.

```
location /service-cloud/ {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    proxy_request_buffering off;
```

```
    proxy_pass https://servicestage.besclouds.com/service-cloud;
}
```

**Step 2** Use the following address to send a JavaScript request in the **1.5.2 Developing an Integration Page**. Replace *location.protocol* with the domain name of the integrator and the service name configured in the previous step, for example, **demo**.

```
const $ContextPath = location.protocol/service-cloud
let serviceUrl = $ContextPath+ "/webclient/chat_client/js/thirdPartyClient.js?"+"&t=" + timestamp;
let thirdUserData = {};
......
```

**----End**