Huawei Cloud Meeting

Developer Guide

Issue 03

Date 2024-04-23

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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.

Contents

1 Openness of Huawei Cloud Meeting

Introduction

Both Huawei Cloud Meeting server and clients are open and available for integration. You can integrate diverse capabilities of Huawei Cloud Meeting in the following three ways:

- Server integration: Huawei Cloud Meeting provides open REST APIs for enterprise, user, corporate directory, and meeting management, meeting control, and dashboard. Third-party service systems (including service backends and terminal apps) can call related APIs to implement their own service logic.
- Client SDK integration: Developers can use the open client SDK of Huawei Cloud Meeting to integrate capabilities such as creating and joining meetings as well as performing meeting control. The SDK has integrated the in-meeting UI. Third-party applications only need to call a few APIs to integrate video capabilities of Huawei Cloud Meeting into their applications.
- Client scheme integration: Third-party applications (including web pages and terminal apps) can use parameters defined in URL protocol to directly start the Huawei Cloud Meeting client and join meetings.

Figure 1-1 Integrating Huawei Cloud Meeting capabilities

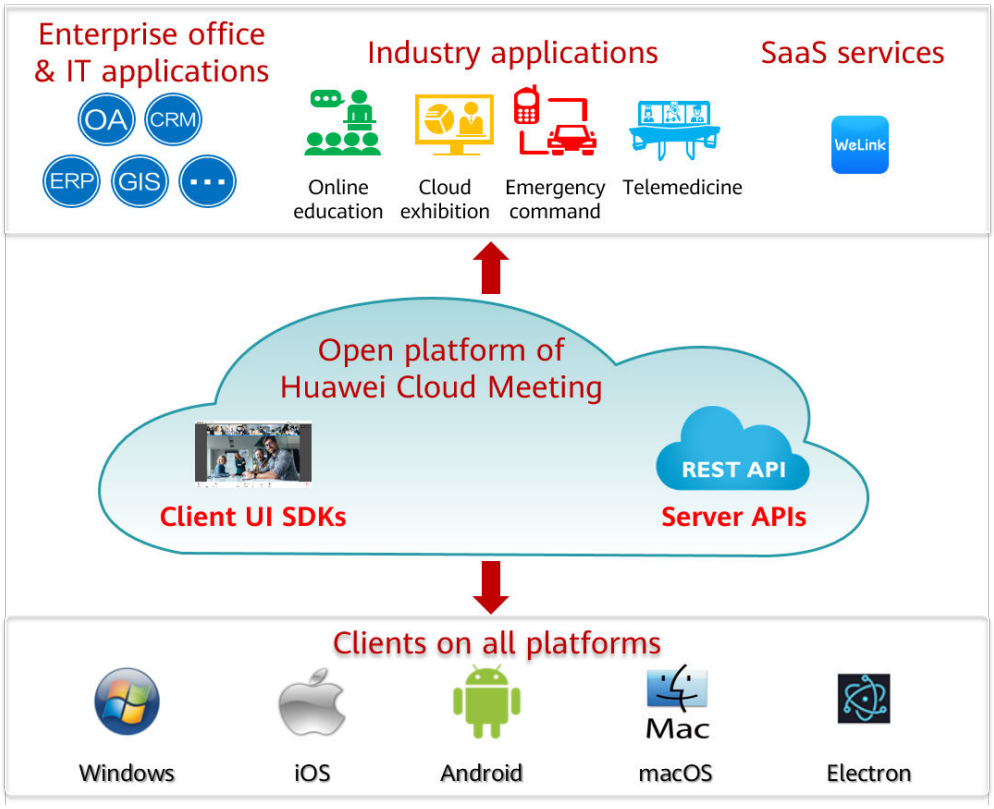


Table 1-1 Capabilities available for integration

Category	Capability	Details
Server capabilities	Enterprise management	• SP administrators can add, delete, modify, and query enterprises. • SP administrators can manage enterprise resources.
	User management	• Adding, deleting, modifying, and querying departments • Adding, deleting, modifying, and querying users • User password management • Corporate directory query
	Cloud meeting room management	• Allocating, modifying, reclaiming, and querying cloud meeting rooms
	Hard terminal management	• Activating, querying, and deleting hard terminals

Category	Capability	Details
	Meeting management	• Creating, modifying, and querying meetings (including recurring meetings) • Historical meeting query
	Meeting control	• Inviting, hanging up, and muting participants during a meeting • Setting continuous presence during a meeting
	Meeting notification push	**Single meeting** • Participant status • Loudest speaker • Meeting quality • Live captions • Meeting status **Enterprise-level meetings** • Meeting start status • Meeting end status • Meeting close status
	Dashboard	• Meeting QoS query • Statistics
	Bulletin board management	• Publication management • Program management • Material management
Open capabilities of client UI SDKs	Android UI SDK	• Creating meetings and joining meetings after login or anonymously • Scheduling, editing, and canceling meetings and obtaining meeting details and the list of cloud meeting rooms • Point-to-point calls
	iOS UI SDK	• Creating meetings and joining meetings after login or anonymously • Scheduling meetings • Point-to-point calls

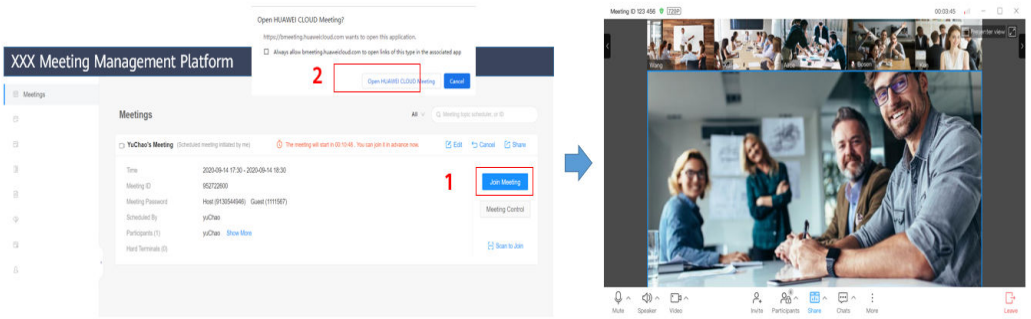
Category	Capability	Details
	Windows UI SDK	• Creating meetings and joining meetings after login or anonymously • Scheduling, editing, and canceling meetings and obtaining meeting details and the list of cloud meeting rooms • Point-to-point calls
	macOS UI SDK	• Creating meetings and joining meetings after login or anonymously • Scheduling, editing, and canceling meetings and obtaining meeting details and the list of cloud meeting rooms • Point-to-point calls
	Electron UI SDK	• Creating meetings and joining meetings after login or anonymously • Scheduling, editing, and canceling meetings and obtaining meeting details and the list of cloud meeting rooms • Point-to-point calls
Open capabilities of the client scheme	Huawei Cloud Meeting Android app	• Starting the app • Starting and logging in to the app • Joining a meeting • Starting the app and joining meetings
	Huawei Cloud Meeting iOS app	• Starting the app • Starting and logging in to the app • Joining a meeting • Starting the app and joining meetings
	Huawei Cloud Meeting Windows client	• Starting the app • Starting and logging in to the app • Joining a meeting • Starting the app and joining meetings
	Huawei Cloud Meeting macOS client	• Starting the app • Starting and logging in to the app • Joining a meeting • Starting the app and joining meetings

Integration Scenario 1: Third-Party Systems Schedule Meetings on Huawei Cloud Meeting and Start the Client to Join Meetings

A third-party application can call REST APIs of the Huawei Cloud Meeting server to schedule a meeting. Application users only need to click the URL provided by the application to start the Huawei Cloud Meeting client and join the meeting.

Application scenario: Third parties want to manage and schedule meetings in their own service systems and allow users to join meetings using Huawei standard clients.

Figure 1-2 Terminal scheme integration



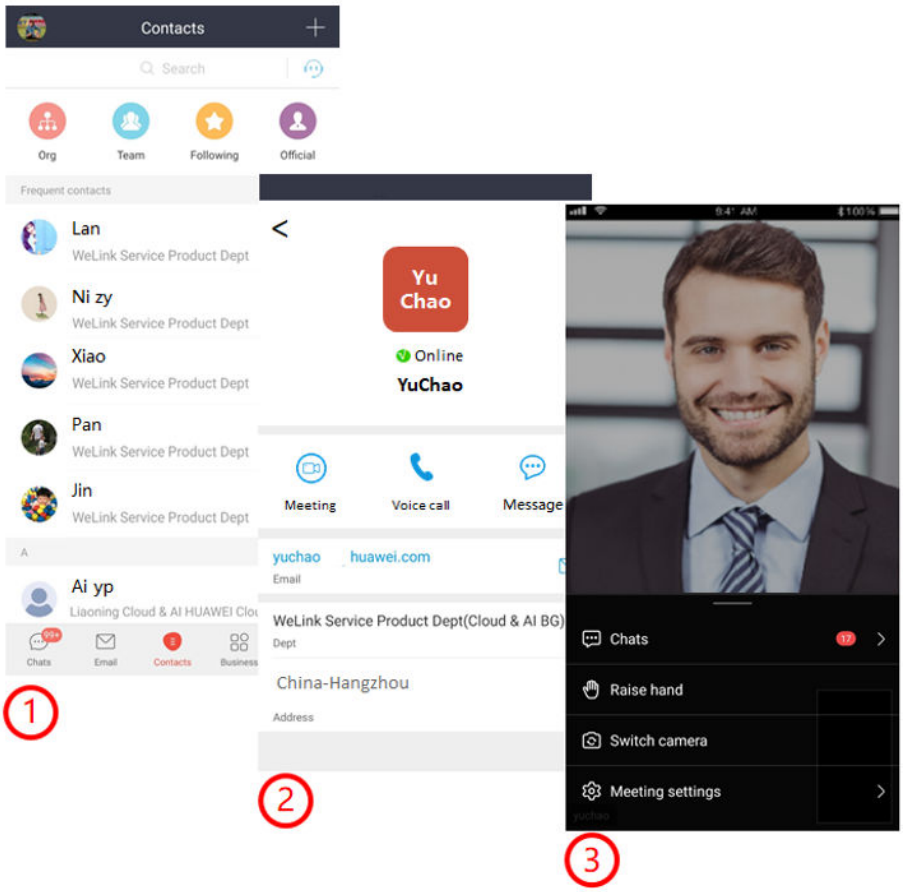
For details about REST APIs, see the **Server API Reference**. For details about the URL for starting the client, see **Terminal Scheme Application Integration**.

Integration Scenario 2: Third-Party Applications Quickly Integrate Huawei Cloud Meeting Client Capabilities Using UI SDK

A third-party application calls the client UI SDK APIs to quickly integrate meeting functions. You only need to develop an entry on your application, then you are free to call corresponding APIs to schedule, create, and join meetings. Meeting control operations and GUI are ready for you to use directly.

Application scenario: Third parties want to integrate the meeting capability into their apps. The UI provided by the UI SDK meets their requirements.

Figure 1-3 Third-party application integrating Huawei Cloud Meeting capabilities using terminal SDK



In the preceding figure, 1, 2, and 3 are screenshots from the same application. 1 and 2 are provided by the application itself. 3 is provided by the SDK. For details about the SDK, see **Client SDK Reference**.

2 Integrating Huawei Cloud Meeting Accounts

Huawei Cloud Meeting involves two account systems, namely Huawei Cloud accounts and Huawei Cloud Meeting accounts. Huawei Cloud Meeting accounts are divided into three types based on role: SP administrator, enterprise administrator, and enterprise user.

- Huawei Cloud accounts are used to purchase Huawei Cloud Meeting resources, log in to the Huawei Cloud console, request app IDs, and debug APIs in **API Explorer**.
- Huawei Cloud Meeting accounts can be used in the following scenarios:
  - SP administrators manage enterprises and enterprise resources.
  - Enterprise administrators manage departments, users, cloud meeting rooms, and meetings, and control meetings.
  - Enterprise users manage and control their own meetings.

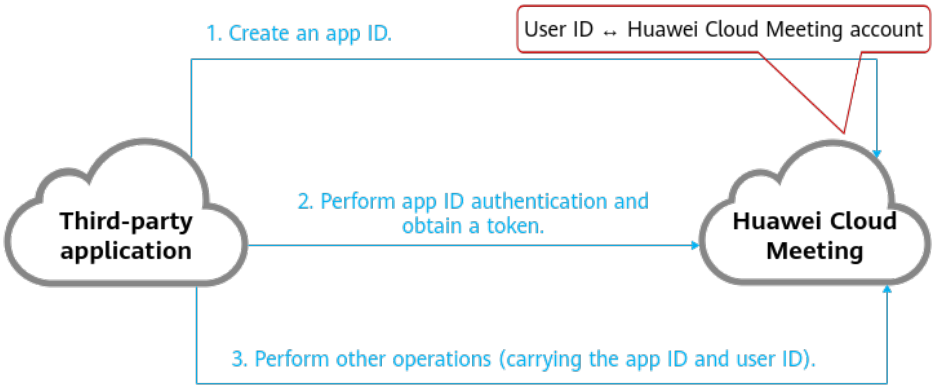
Third-party applications can integrate Huawei Cloud Meeting account system using app ID authentication or account and password authentication.

NOTE

- For details about how to request an enterprise administrator account, see **Preparations**. For details about how to request an app ID, see **Requesting an App ID**.

App ID Authentication (Recommended)

Figure 2-1 Integration using app ID authentication



Step 1 Create an app ID for the third-party application. You only need to do this for once. For details, see **Requesting an App ID**.

Step 2 Use the app ID and a third-party user ID to call the authentication API to obtain the access token.

Step 3 When calling other management APIs to manage a specific user (for example, the API for inviting a participant), carry the access token and third-party user ID.

----End

In this authentication mode, third-party applications are unaware of Huawei Cloud Meeting accounts. The Huawei Cloud Meeting system manages the binding and unbinding between third-party user IDs and Huawei Cloud Meeting accounts.

Account and Password Authentication

Figure 2-2 Integrating using account and password authentication

Step 1 If you want to integrate the Huawei Cloud Meeting account system using account and password authentication, call the REST APIs of the server to create accounts

and manage the binding relationships between third-party user IDs and Huawei Cloud Meeting accounts in the third-party system.

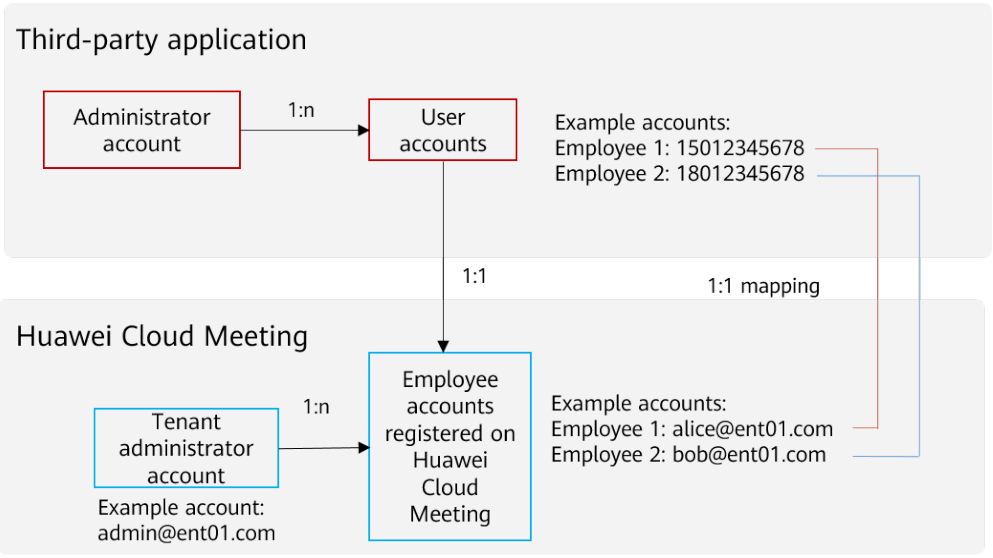
Step 2 Use the provisioned Huawei Cloud Meeting account and password to call the authentication API to obtain the access token.

Step 3 When calling other management APIs to manage a specific user (for example, the API for inviting a participant), carry the access token and Huawei Cloud Meeting account.

----End

In this authentication mode, third-party user IDs are bound with Huawei Cloud Meeting accounts in 1:1 mode on the third-party system.

Figure 2-3 Relationships between a third-party account and a Huawei Cloud Meeting account



NOTE

Third-party user IDs can be bound with Huawei Cloud Meeting accounts in n:1 mode as long as the third-party system can manage account binding and unbinding on itself.

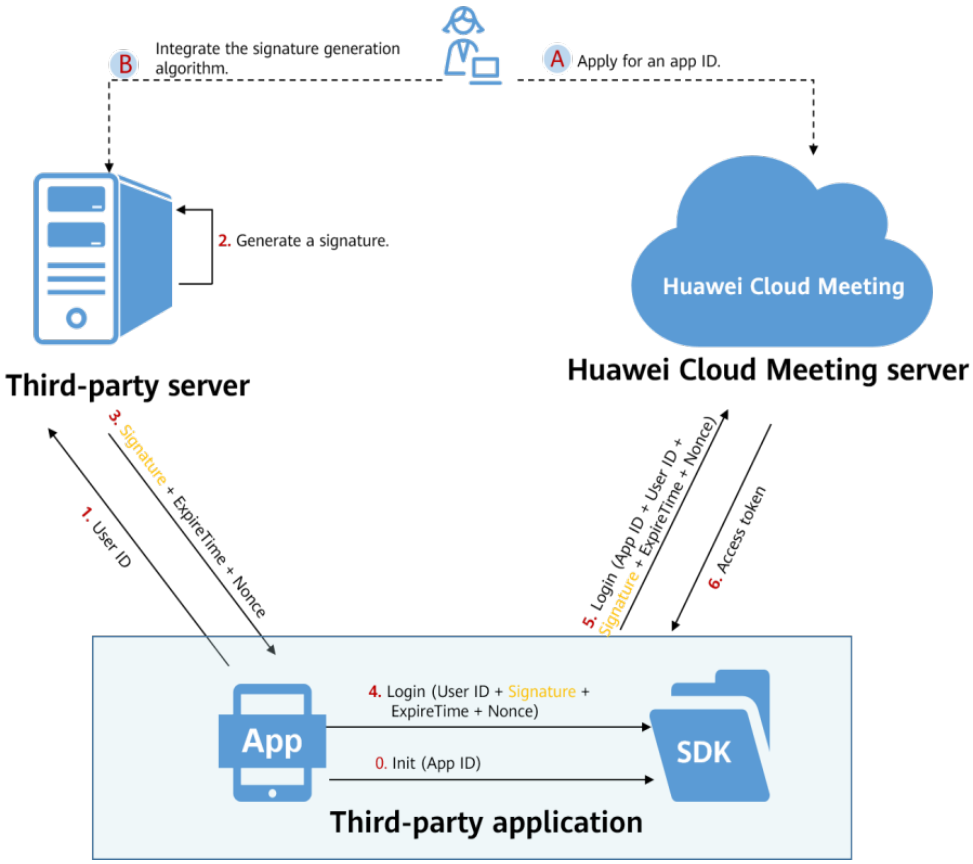
3 Introduction to App ID Authentication

Authentication using app IDs is supported to make integrating Huawei Cloud Meeting easier and more secure for third-party applications. An app ID identifies an application. An app ID can be used on a third-party desktop terminal, mobile terminal, and web application at the same time.

How App ID Authentication Works

1. **App ID authentication on a third-party client**

   Figure 3-1 App ID authentication process on a third-party client



   Prerequisites:

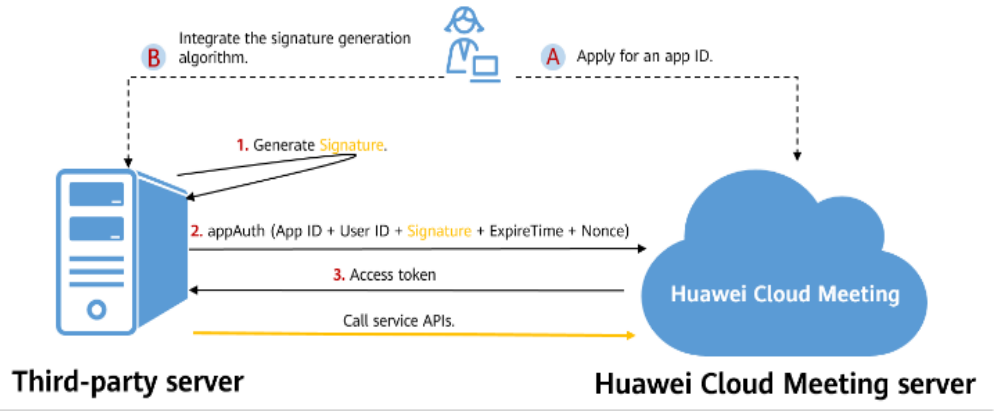
a. You have requested and obtained an app ID and appKey on the **Huawei Cloud Meeting console**.
b. You have integrated the signature generation algorithm on your own server. For details, see **Signature Generation Algorithms for Third-Party Service Integration**.
c. The app ID is passed during SDK initialization. For details, see section "Initialization" in **Client SDK Reference**.

Process:

a. The third-party client sends the third-party user ID to the third-party server.
b. The third-party server generates a signature for authentication based on the app ID, user ID, and application key.
c. The third-party server returns the value of **Signature**, **ExpireTime**, and **Nonce**.
d. The third-party client calls the login API of the client SDK. The parameters include **User ID**, **Signature**, **ExpireTime**, and **Nonce**.
e. The client SDK sends an authentication request to the Huawei Cloud Meeting server.
f. After the authentication, the SDK obtains the access token. The token is inaccessible to the third-party client and is maintained and updated by the SDK.

2. **App ID authentication on a third-party server**

Figure 3-2 App ID authentication process on a third-party server



Prerequisites:

a. You have requested and obtained an app ID and appKey on the **Huawei Cloud Meeting console**.
b. You have integrated the signature generation algorithm on your own server. For details, see **Signature Generation Algorithms for Third-Party Service Integration**.

Process:

a. The third-party server uses the third-party user ID, app ID, appKey, **ExpireTime**, and **Nonce** required in API calling to generate a signature for authentication.

b. The third-party server calls the app ID authentication API (a REST API) of the Huawei Cloud Meeting server.
c. After the authentication is successful, a token is returned.
d. The third-party server uses the access token to call other service APIs.

NOTE

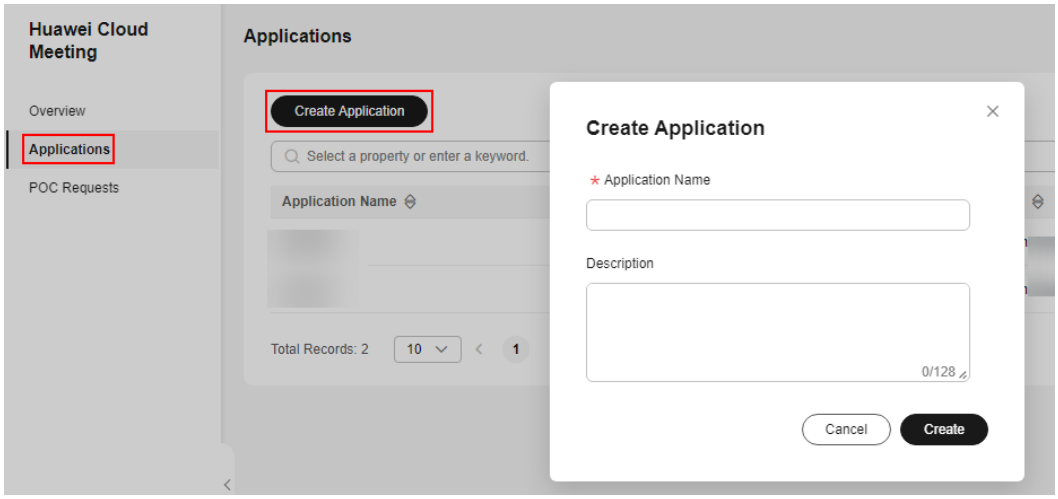
1. The validity period of an access token is 12 to 24 hours.
2. The **User ID** parameter is optional when you are generating a signature or calling the **appAuth** API. If the **User ID** parameter is not specified, the creator of the enterprise, namely, the enterprise owner is used by default.
3. The third-party application must ensure each user ID is unique in the enterprise. If the third party is a service provider and provides an application to multiple enterprises, both **User ID** and **Corp ID** are mandatory when you call the APIs.
4. If you want to use the Huawei Cloud Meeting contacts, information, such as email address, name, and phone number, can be carried during login authentication. The information will be written into Huawei Cloud Meeting contacts.

Requesting an App ID

Step 1 Use your Huawei Cloud account to log in to the **Huawei Cloud Meeting console**. Ensure that you have subscribed to Huawei Cloud Meeting using the Huawei Cloud account or have bound the Huawei Cloud account to a Huawei Cloud Meeting enterprise administrator account. For details, see **Preparations** in **Development Process**.

Step 2 In the navigation pane, choose **Applications** and click **Create Application**. In the displayed dialog box, enter the name and description of the third-party application.

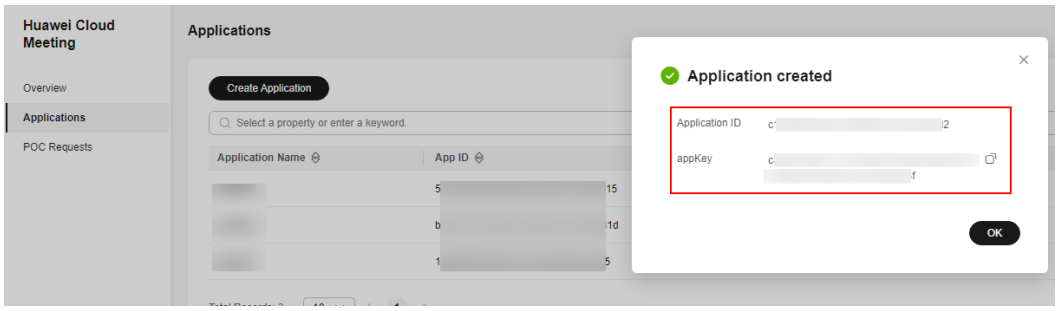
Figure 3-3 Creating an application



Step 3 Click **Create**.

An app ID and appKey are generated.

Figure 3-4 Application created



----End

NOTE

1. An appKey is the key used to generate the authentication signature. The key must be properly kept on the third-party server. Otherwise, meeting resources may be stolen.
2. appKeys must be stored on third-party servers rather than third-party clients. appKeys stored on clients can be easily obtained through decompilation.
3. appKeys can only be reset. There is no way to retrieve lost keys. When a key is lost, generate a new one. The original key will expire in one month.

Signature Generation Algorithms for Third-Party Service Integration

If an application is used only in a single enterprise, the algorithm for generating an authentication signature is as follows:

```
Signature = HexEncode(HMAC-SHA256((App ID + ":" + User ID + ":" + ExpireTime + ":" + Nonce), appKey))
```

If an application is developed by a service provider and used in multiple enterprises, the algorithms for generating an authentication signature in different scenarios are as follows:

- When enterprise users manage their meetings:
  ```
  Signature = HexEncode(HMAC-SHA256((App ID + ":" + CorpID  + ":" + User ID + ":" + ExpireTime + ":" + Nonce), appKey))
  ```
- When enterprise administrators manage enterprise resources:
  ```
  Signature = HexEncode(HMAC-SHA256((App ID + ":" + CorpID  + "::" + ExpireTime + ":" + Nonce), appKey))
  ```

  NOTE

  An enterprise administrator can also have a user ID. The user ID must have the administrator permissions.
- When service provider administrators manage service provider resources (for example, creating enterprises or allocating resources to enterprises):
  ```
  Signature = HexEncode(HMAC-SHA256((App ID + ":::" + ExpireTime + ":" + Nonce), appKey))
  ```

In the preceding algorithms:

1. The input data of HMAC-SHA256 is the values of **App ID**, **CorpID** (optional), **User ID** (optional), **ExpireTime**, and **Nonce**. The values must be separated by colons (:). For example, **d5e17*****489e:alice@ent01:1604020600:EycLQs*********nINuU1EBpQ**

NOTE

In SP mode, the colons (:) cannot be omitted even if **User ID** and **Corp ID** are left empty.

2. The key of HMAC-SHA256 is the value of appKey. For example, **tZAe********q32T**.
3. The binary number generated by HMAC-SHA256 needs to be converted into a hexadecimal string (**HexEncode**). The signature generated by the preceding data and key is **2a8c780c**********5a6c44f3c1b3c2455d**.
4. **ExpireTime**: expiration timestamp of the authentication signature. The unit is second. For example, if the current system timestamp is 1604020000 and you want to set the validity period of the signature to 10 minutes, the value of **ExpireTime** is **1604020600** (1604020000 + 10 x 60).
5. **Nonce**: a random character string, which must be different each time when the authentication signature is calculated. The string has 32 to 64 characters.

NOTE

1. Because the authentication signature has a validity period, the difference between the system time of the third-party server and that of the Huawei Cloud Meeting server cannot be too large and must be smaller than the validity period specified by **ExpireTime**, for example, 10 minutes in the preceding example. The time of the Huawei Cloud Meeting server is synchronized with the standard NTP time.

2. When **ExpireTime** is set to **0**, the signature does not expire. To prevent replay attacks, do not set **ExpireTime** to **0**.

Source code of the signature generation algorithm in Java:

```
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class HmacSHA256 {

    // Hexadecimal character set
    private final static char[] DIGEST_ARRAYS = {
        '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'
    };

     /**
     * Function: signature generation algorithm
     * Input parameters:
     *          1. data: HMAC-SHA256 input data
     *          2. key: appKey
     * Output parameter: a hexadecimal character string generated using the value of HMAC-SHA256
     */
    public static String encode(String data, String key) {
        byte[] hashByte;
        try {
            Mac sha256HMAC = Mac.getInstance("HmacSHA256");
            SecretKeySpec secretKey = new SecretKeySpec(key.getBytes("UTF-8"), "HmacSHA256");
            sha256HMAC.init(secretKey);

            hashByte = sha256HMAC.doFinal(data.getBytes("UTF-8"));
        } catch (NoSuchAlgorithmException | UnsupportedEncodingException | InvalidKeyException e) {
            return null;
        }

        return bytesToHex(hashByte);
```



```
    }

    /**
     * Function: converting an array in bytes into a hexadecimal string
     * Input parameter:
     *          1. bytes: byte array to be converted
     * Output parameter: hexadecimal character string
     */
    private static String bytesToHex(byte[] bytes) {
        StringBuffer hexStr = new StringBuffer();
        for (int i = 0; i < bytes.length; i++) {
            hexStr.append(DIGEST_ARRAYS[bytes[i] >>> 4 & 0X0F]);
            hexStr.append(DIGEST_ARRAYS[bytes[i] & 0X0F]);
        }

        return hexStr.toString();
    }
}
```

Source code of the signature generation algorithm in Python:

```
import hmac
from hashlib import sha256

class HmacSHA256:
      def __init__(self, sig_data, sig_key):
         self.data = sig_data
         self.key = sig_key

      def encode(self):
         try:
            sig_data = self.data.encode('utf-8')
            secret_key = self.key.encode('utf-8')
            signature = hmac.new(secret_key, sig_data, digestmod=sha256).hexdigest()
         except Exception as e:
            print (e)
            raise e
          return signature
```

Source code of the signature generation algorithm in C++:

```
#include <openssl/hmac.h>
#include <string.h>
#include <iostream>
using namespace std;

const int HMAC_ENCODE_SUCCESS = 0;
const int HMAC_ENCODE_FAIL = -1;
const int HMAC_SHA256_STR_LEN = 65;

/**
* Function: converting a byte stream into a hexadecimal character string
* Input parameters:
*          1. input: byte stream pointer
*          2. len: number of bytes in a byte stream
* Output parameters:
*          3. output: output buffer
*          6. output_length: length of the signature character string
* Returned value:
*          0: failed
*          Integer greater than 0: length of the character string after conversion
*/
int Byte2HexStr(char * output, unsigned char * input, unsigned int len)
{
    if ((NULL == output) || (NULL == input))
    {
        return 0;
    }
```



```
    unsigned int i = 0;
    for (i = 0; i < len; i++)
    {
        sprintf_s(output + 2*i, HMAC_SHA256_STR_LEN-2*i, "%x%x", (input[i] >> 4) & 0x0F, input[i]&0x0F);
    }

    *(output + 2*i) = '\0';

    return 2*i;
}


/**
* Function: signature generation algorithm
* Input parameters:
*          1. key: appKey
*          2. key_length: length of the appKey
*          3. input: HMAC-SHA256 input data
*          4. input_length: length of the character string in the HMAC-SHA256 input data
* Output parameters:
*          5. output: output buffer
*          6. output_length: length of the signature character string
* Returned value:
*          0: successful
*          -1: failed
*/
int HmacEncode(const char * key, unsigned int key_length,
    const char * input, unsigned int input_length,
    char output[HMAC_SHA256_STR_LEN], unsigned int &output_length) {

    // Calculate the byte stream of HMAC_SHA256.
    const EVP_MD * engine = EVP_sha256();

    unsigned char * byte_output = (unsigned char*)malloc(EVP_MAX_MD_SIZE);
    if (NULL == byte_output)
    {
        output_length = 0;
        return HMAC_ENCODE_FAIL;
    }
    unsigned int  byte_output_length = 0;

    HMAC_CTX *ctx;
    ctx = HMAC_CTX_new();

    HMAC_Init(ctx, key, strlen(key), engine);
    HMAC_Update(ctx, (unsigned char*)input, strlen(input));
    HMAC_Final(ctx, byte_output, &byte_output_length);

    HMAC_CTX_free(ctx);

    // Convert the HMAC_SHA256 byte stream into a hexadecimal character string.
    int ret = Byte2HexStr(output, byte_output, byte_output_length);
    free(byte_output);

    if (0 == ret)
    {
        output_length = 0;
        return HMAC_ENCODE_FAIL;
    }
    else
    {
        output_length = ret;
        return HMAC_ENCODE_SUCCESS;
    }

}
```


> **NOTE**
>
> The signature generation algorithm in C++ is based on OpenSSL 1.1.0 or later. Compile and install the OpenSSL library first. For details, see **OpenSSL**.

Source code of the signature generation algorithm in JavaScript:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/crypto-js.js"> </script>
<script>
    /**
     * 1. Do not include the signature generation algorithm in the frontend code. Otherwise, the appKey may be leaked, causing enterprise resource theft.
     * 2. The sample code is used only for debugging.
     */
    function genSignature(data,appKey){
        var sign = CryptoJS.HmacSHA256(data,appKey).toString();
        return sign
    }
</script>
```

App ID Authentication on Third-Party Clients Integrated with the UI SDK

When a third-party client initializes the SDK, **App ID** needs to be passed.

- For details about the initialization API of the Android SDK, see Android SDK **Initialization** in the *Client SDK Reference*.
- For details about the initialization API of the iOS SDK, see iOS SDK **Initialization** in *Client SDK Reference*.
- For details about the initialization API of the Windows SDK, see Windows SDK **Initialization** in *Client SDK Reference*.
- For details about the initialization API of the macOS SDK, see macOS SDK **Initialization** in *Client SDK Reference*.

When a third-party client logs in, obtain **Signature**, **ExpireTime**, and **Nonce** from the third-party server and call the login API of the SDK to complete authentication.

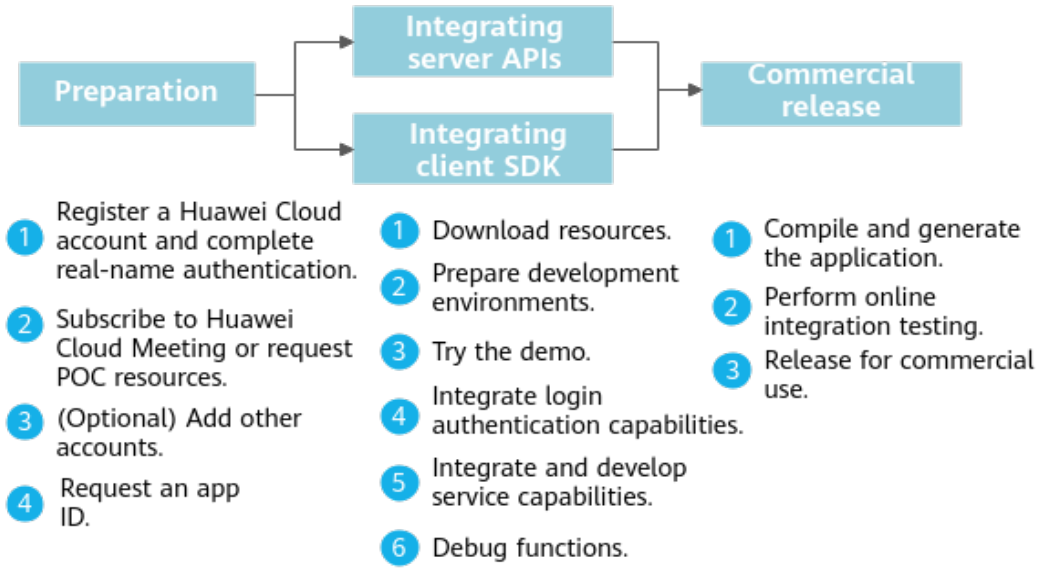
- For details about the login API of the Android SDK, see Android SDK **Login** in *Client SDK Reference*.
- For details about the login API of the iOS SDK, see iOS SDK **Login** in *Client SDK Reference*.
- For details about the login API of the Windows SDK, see Windows SDK **Login** in *Client SDK Reference*.
- For details about the login API of the macOS SDK, see macOS SDK **Login** in *Client SDK Reference*.

App ID Authentication on Third-Party Servers

Third-party servers authenticate app IDs through the app ID authentication API and obtain the access token. For details, see section **Authenticating an App ID** in *Server API Reference*.

4 Development Process

Figure 4-1 Development process



Preparations

1. Register a HUAWEI ID and enable Huawei Cloud services.

   Visit the **Huawei Cloud website**, click **Register**, and register an account as prompted. After the registration is successful, complete enterprise real-name authentication as soon as possible. For details, see **Enterprise Real-Name Authentication**.

2. Create a Huawei Cloud Meeting enterprise administrator account.

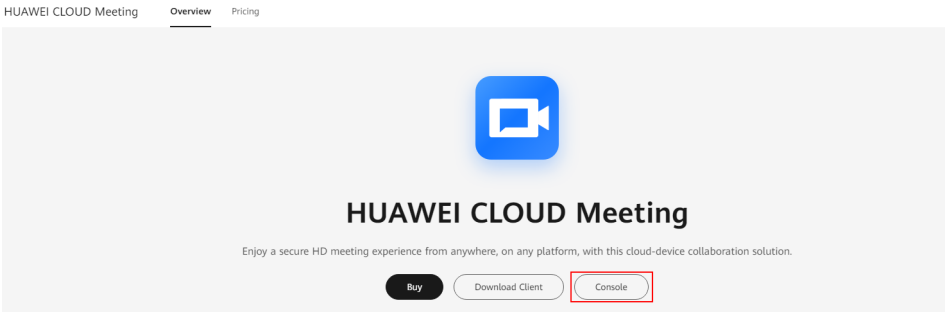
   – **Official commercial use**

   Subscribe to the Huawei Cloud Meeting service. For details about subscription, see **Subscription Process**.

   – **Free trial**

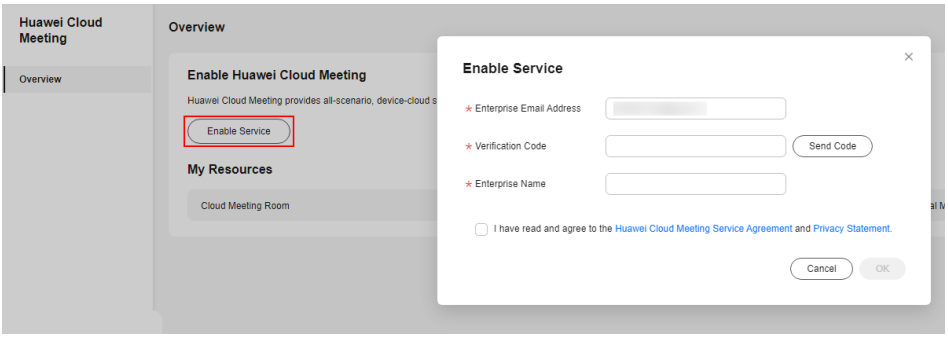
   a. Log in to Huawei Cloud using the account registered in step 1, go to **Huawei Cloud Meeting website**, and click **Console**.

Figure 4-2 Accessing the console



b. Click **Enable Service**.

Figure 4-3 Enabling the Huawei Cloud Meeting service



c. Choose **POC Requests** in the navigation pane, click **Create POC Request**, and enter the enterprise name, contact information, and application scenario.

Figure 4-4 Requesting POC resources

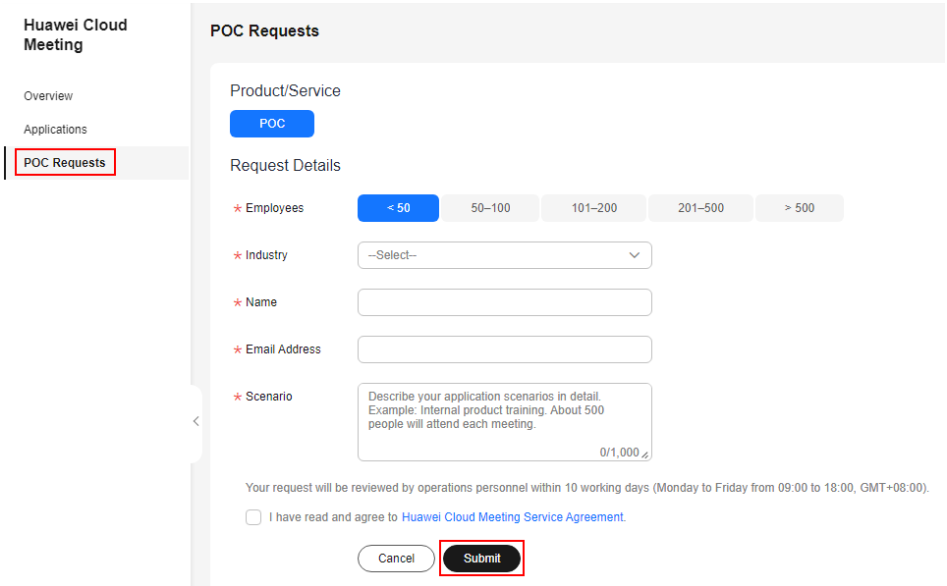
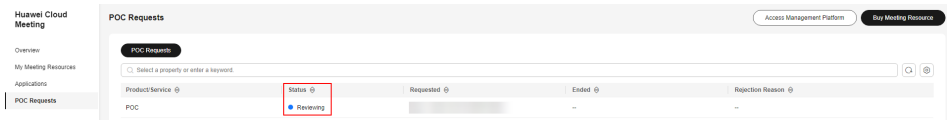


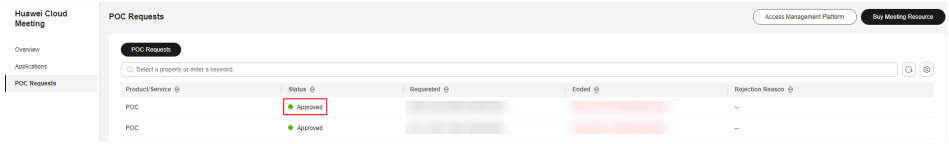
Figure 4-5 Waiting for approval



NOTE

- To request secondary development resources, contact Huawei sales personnel.

Figure 4-6 Request approved

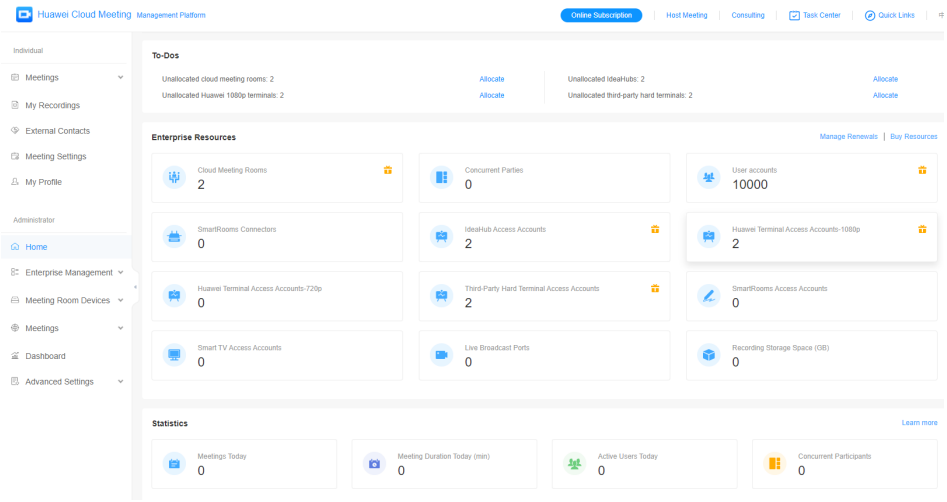


3. (Optional) Add other accounts.

   Click **Access Management Platform** to access the Huawei Cloud Meeting Management Platform.

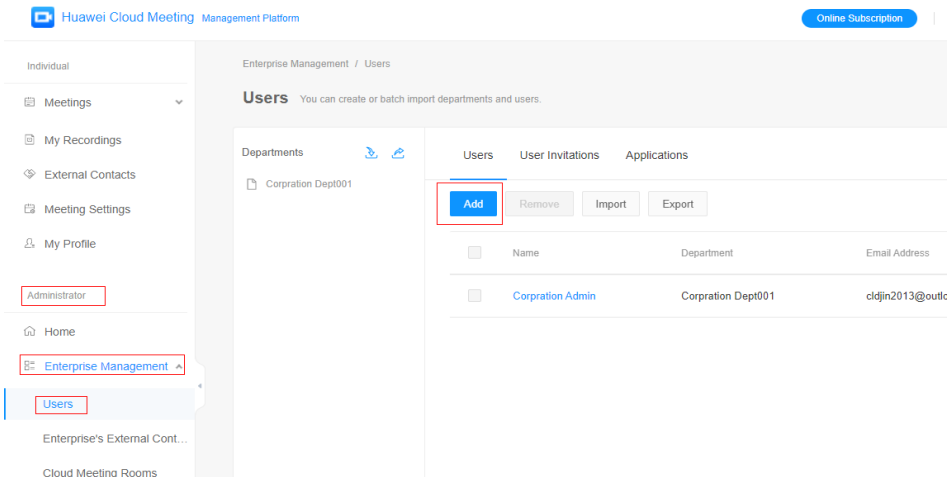
Figure 4-7 Accessing the Management Platform

Figure 4-8 Huawei Cloud Meeting Management Platform



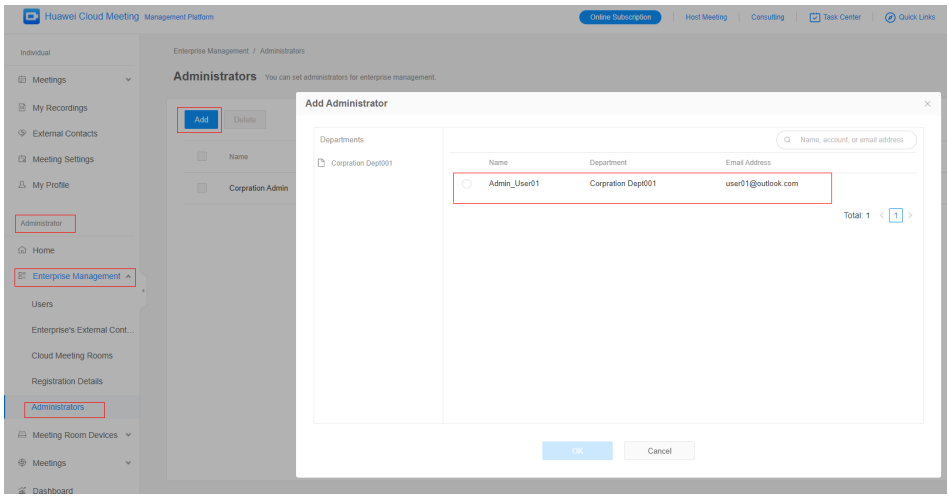
In the navigation pane, choose **Administrator** > **Enterprise Management** > **Users** to add meeting users.

Figure 4-9 Adding meeting users



To set a common meeting user as an enterprise administrator, choose **Administrator** > **Enterprise Management** > **Administrators** and click **Add**.

Figure 4-10 Adding an administrator



4. Request an app ID.

   For details about app ID authentication and how to request an app ID, see **Introduction to App ID Authentication**.

Integrating Server REST APIs

1. Download the server integration development tool and secondary development documents.

   When integrating server APIs, see *Developer Guide* as well as **Server API Reference** for more details.

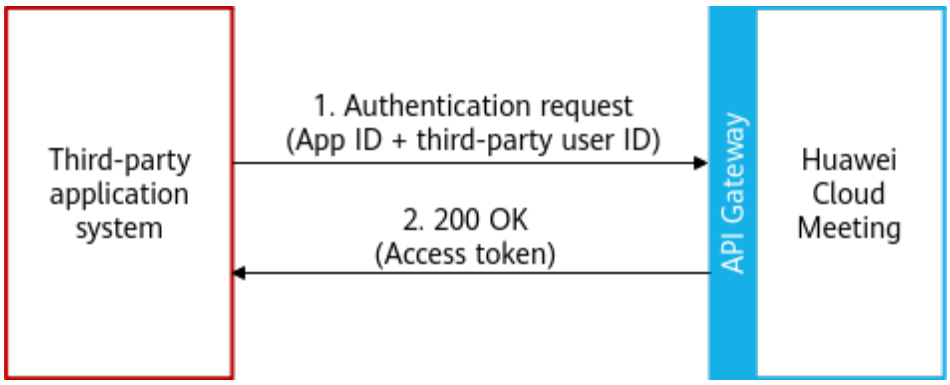
2. Prepare the development environment.

   Subscribe to the Huawei Cloud Meeting service and call the server APIs using a service account. The access address is https://api-intl.meeting.huaweicloud.com.

3. Integrate the login authentication.

Before integrating Huawei Cloud Meeting capabilities, the third-party system needs to call the authentication API to authenticate the third-party application account on the Huawei Cloud Meeting server. For details, see section **Login Authentication** in *Server API Reference*.

Figure 4-11 Server authentication integration



4. Call REST APIs for integration development.

   After the authentication, you can integrate the APIs of the Huawei Cloud Meeting server into the third-party application system.

   For details about the development method, see **Server API Reference**.

5. Debug functions.

   After the development, you can access the Huawei Cloud Meeting debugging environment (https://api-intl.meeting.huaweicloud.com) to debug functions.

   Each server REST API provided by the Huawei Cloud Meeting service can be debugged on **API Explorer** of Huawei Cloud.

Integrating the Client UI SDK

1. Download the client SDK development package and secondary development documents.

   When integrating the client SDK, see *Developer Guide* as well as **Client SDK Reference** for more details.

   Obtain the SDK development package by referring to **Downloading SDKs** in *Client SDK Reference*. The demo source code is in the SDK package.

2. Try the demo.

   Huawei Cloud Meeting provides a demo installation package for each platform SDK. You can download the demo to quickly run and debug functions.

   For details, see **Downloading Demo Installation Packages** of each platform SDK in *Client SDK Reference*.

3. Prepare environments.

   - Development environment
     - Prepare development tools and environments for your platform. For details, see section "Getting Started" of the corresponding platform SDK in the *Client SDK Reference*.
   - Debugging environment

After subscribing to the Huawei Cloud Meeting service or requesting a free trial account, call the client SDK.

4. Call the SDK for integration development.

   You can integrate the SDK APIs of the corresponding platform to your platform.

   For details about the development method, see **Client SDK Reference**.

5. Debug functions.

   After the development, you can access the Huawei Cloud Meeting debugging environment to debug functions.

Commercial Release

After developing and debugging applications, you can release the applications to end users.

5 Best Practices

Open Application Practice Overview

Huawei Cloud Meeting provides open server APIs and open client SDKs for you to flexibly integrate Huawei Cloud Meeting capabilities into your applications. This section describes a few typical scenarios of Huawei Cloud Meeting integration.

Figure 5-1 Open application practice overview

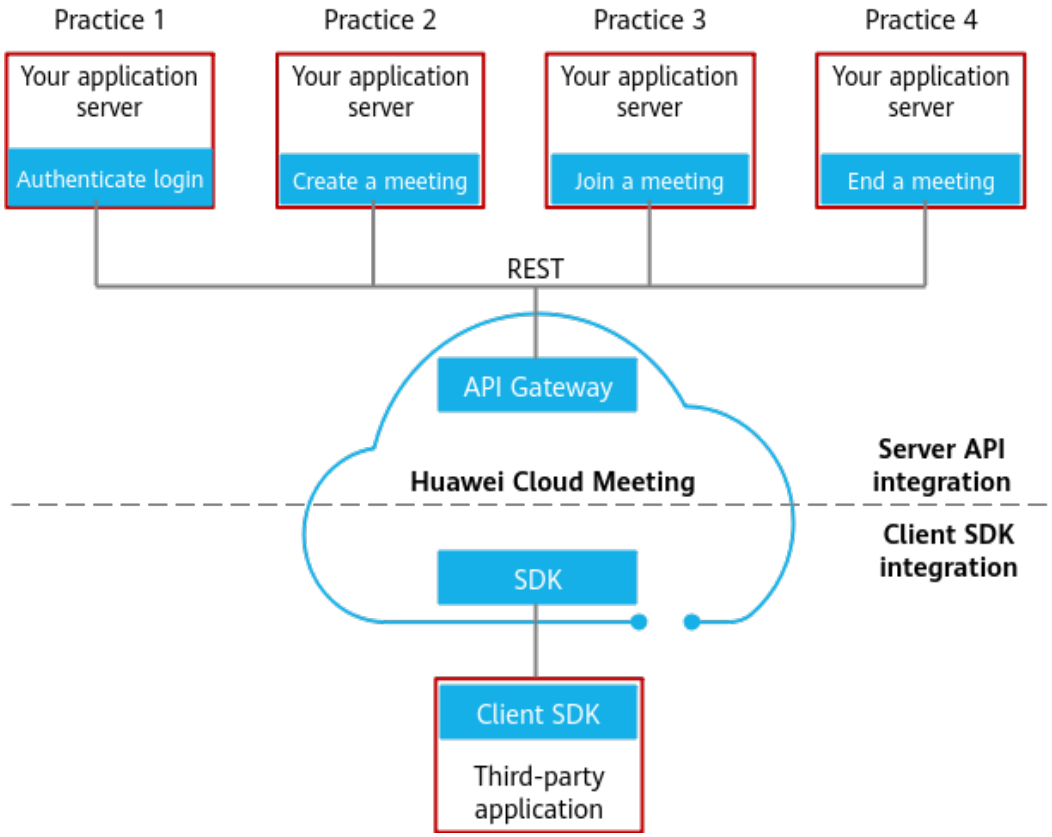


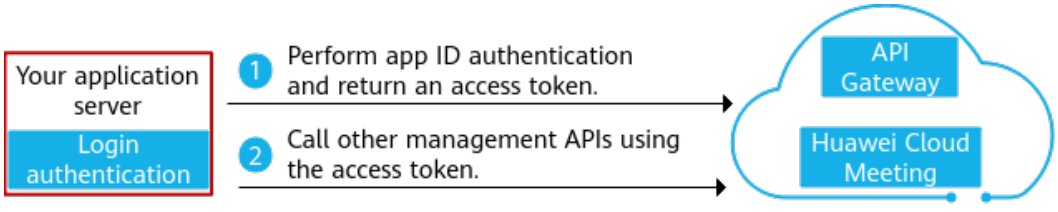
Table 5-1 Server API integration practice

Practice	Scenario
Practice 1: authenticating login	Before calling a server API, log in to the server by calling the app ID authentication API and obtain an access token.
Practice 2: creating a meeting	Creating a meeting on the server: You can integrate your existing management system with meeting management APIs of Huawei Cloud Meeting and call the server APIs to create a meeting. Creating a meeting on a client: You can create an instant or scheduled meeting on a client.
Practice 3: joining a meeting	Answering a call from the server to join a meeting: The server calls the server API to add you to a meeting. You will receive a call and join the meeting after answering the call. Proactively joining a meeting on a client: You can join a meeting by entering the meeting ID and password or clicking the link in the meeting notification email, on the Management Platform, or in the WeChat or WeChat applet.
Practice 4: leaving a meeting	Ending a meeting on the server: You can call the server API to end a meeting. Leaving or ending a meeting on a client: You can end or leave a meeting on the in-meeting screen.

Best Practice 1: Authenticating Login

When integrating Huawei Cloud Meeting server APIs, you are advised to use the app ID authentication mode. After the authentication is successful, you can obtain an access token. After the first authentication is successful, the Huawei Cloud Meeting backend automatically allocates a Huawei Cloud Meeting account to each third-party user.

Figure 5-2 App ID authentication



After the login is successful, Huawei Cloud Meeting automatically allocates an account to the third-party application. The binding relationship is maintained by Huawei Cloud Meeting. You do not need to pay attention to Huawei Cloud Meeting accounts.

Table 5-2 Mapping between third-party accounts and Huawei Cloud Meeting accounts

Third-Party Account	Huawei Cloud Meeting Account
15012345678	Auto-94b91e94ce3f4ddab9ecfe7625418e60
18012345678	Auto-7700ebc37cf580a680cdfde4b34b41d0

Best Practice 2: Creating a Meeting

- Your application server calls Huawei Cloud Meeting server APIs to schedule a meeting or create an instant meeting.
- Your application calls the client SDK APIs to schedule a meeting or create an instant meeting.

Figure 5-3 Integration solution for creating a meeting on the server

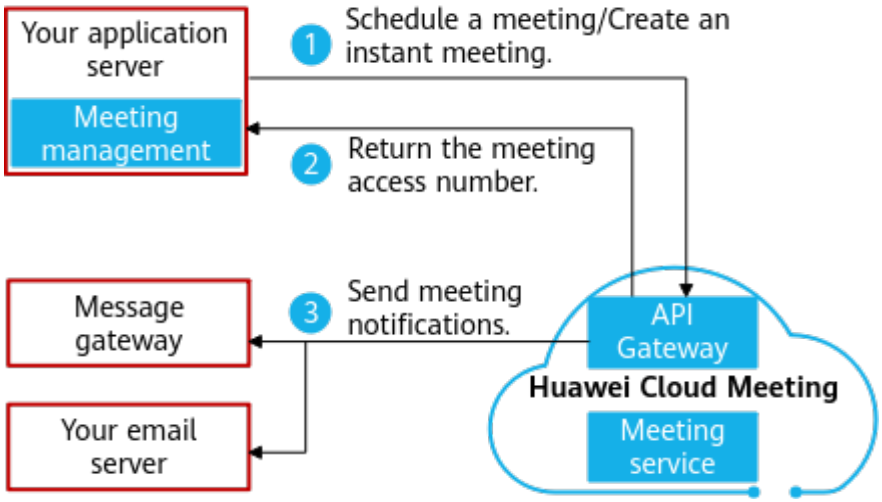
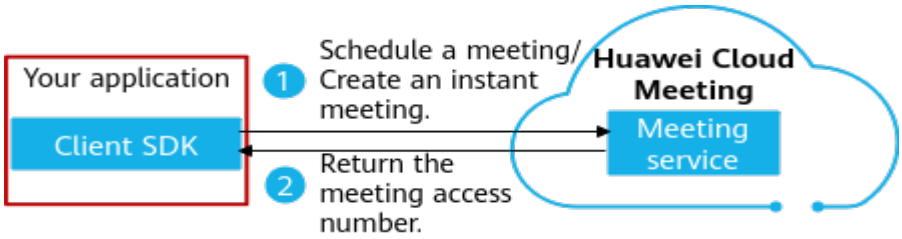


Figure 5-4 Integration solution for creating a meeting on the client

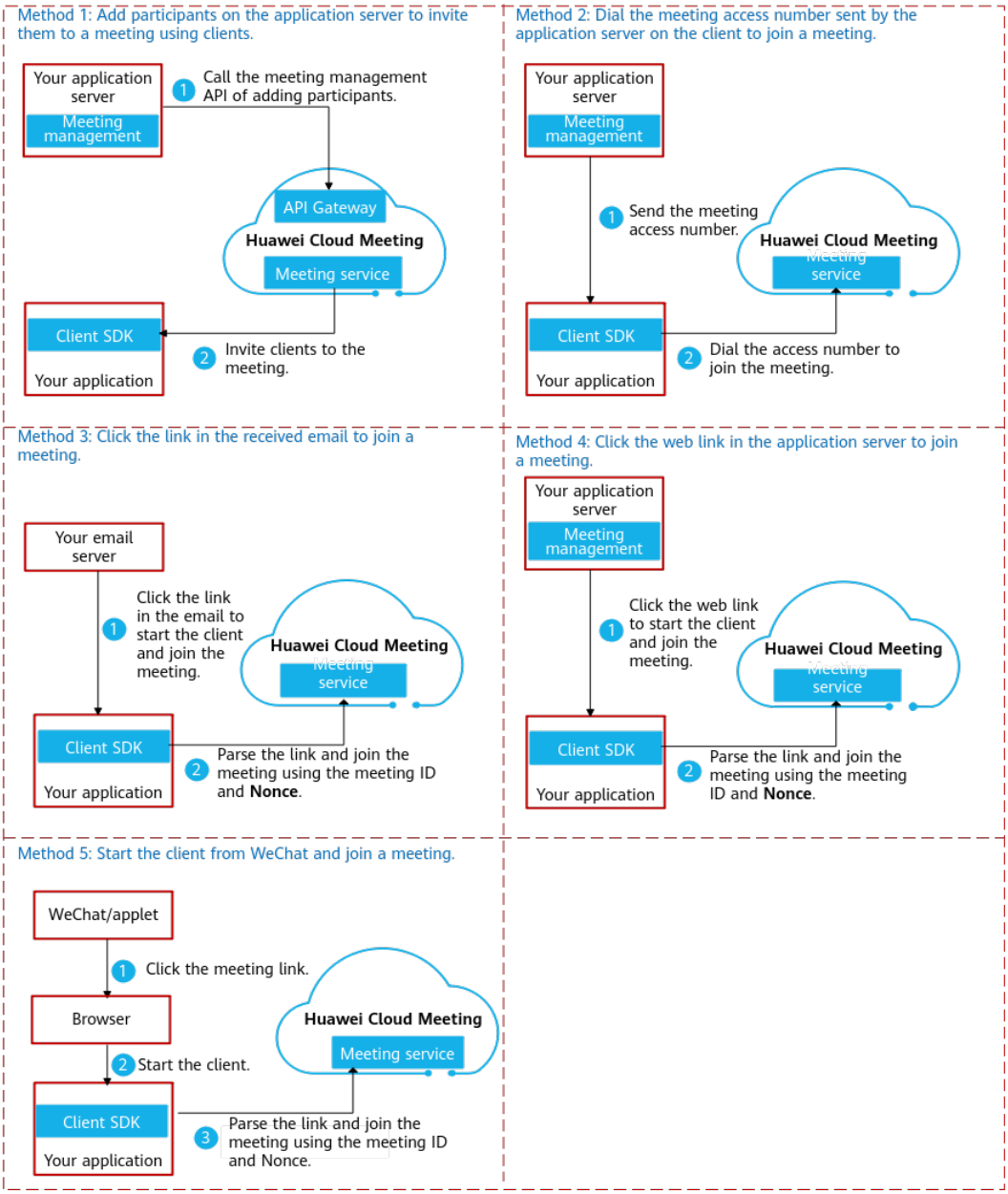


Best Practice 3: Joining a Meeting

Participants can join a meeting in multiple ways, which is achieved by integrating the corresponding API of Huawei Cloud Meeting into your application or application server.

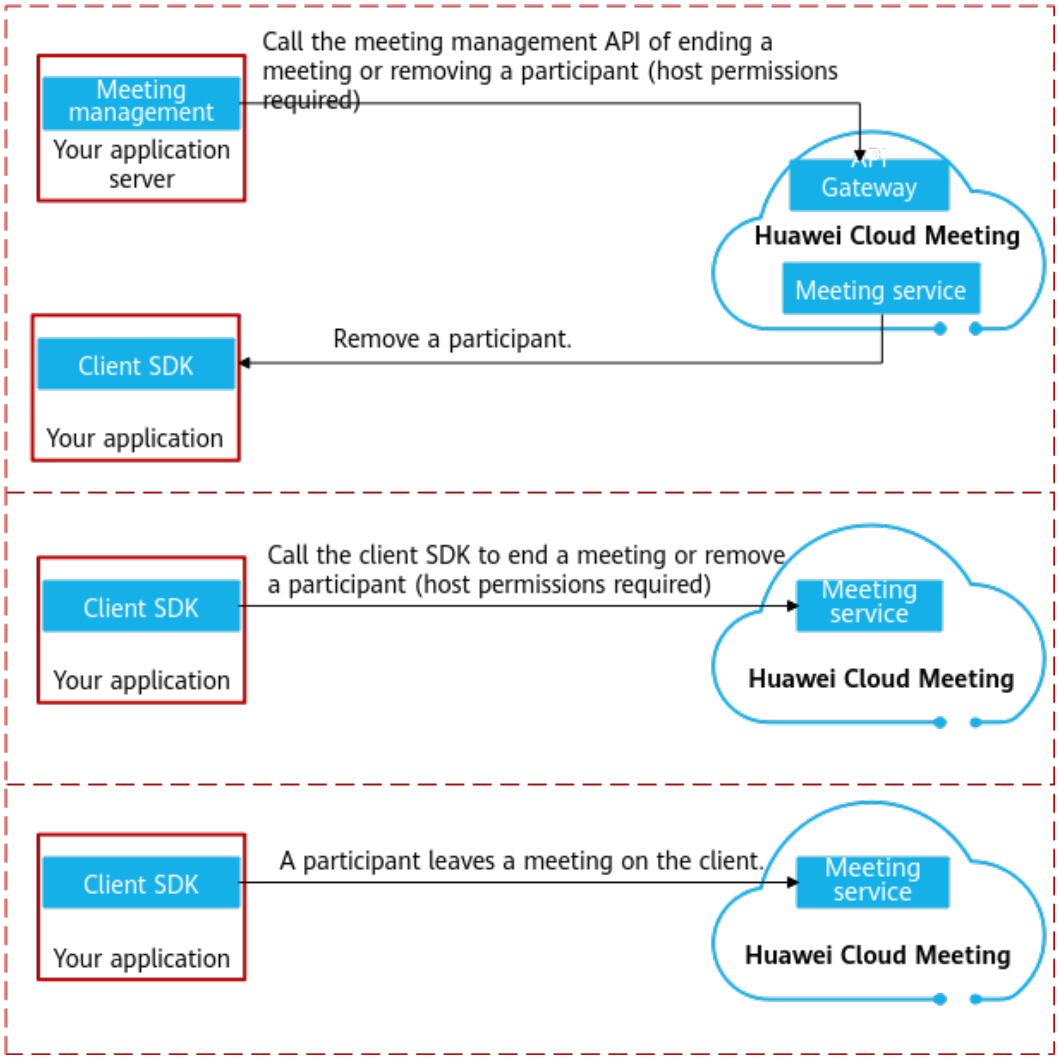
- Method 1: answering a call
- Method 2: entering the meeting ID and password
- Method 3: clicking the link in the notification email
- Method 4: clicking the link on the Management Platform
- Method 5: using the WeChat/applet

Figure 5-5 Integration solution for joining a meeting



Best Practice 4: Ending/Leaving a Meeting

Figure 5-6 Integration solution for leaving a meeting



6 Precautions

- Transmission security
  For secure transmission, REST APIs should use HTTPS instead of HTTP.
- HTTP encoding
  Requests and responses should be encoded using the UTF-8 character set.

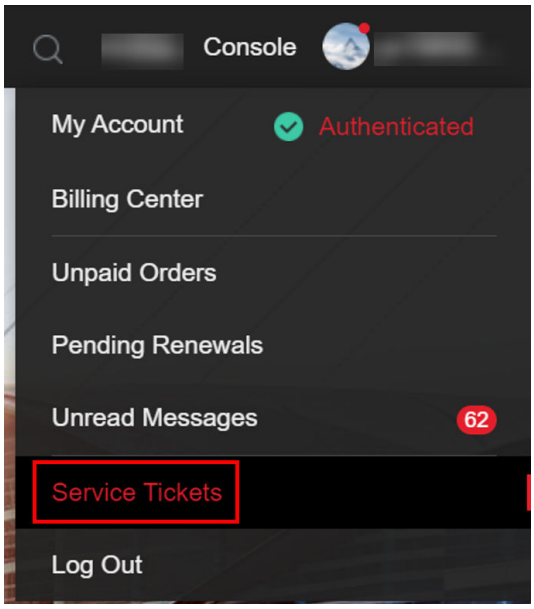
7 Submitting a Service Ticket

If you have any issues in integrating Huawei Cloud Meeting, submit a service ticket. Huawei openness technical support engineers will handle your issues as soon as possible.

Step 1 Log in to the **Huawei Cloud website**.

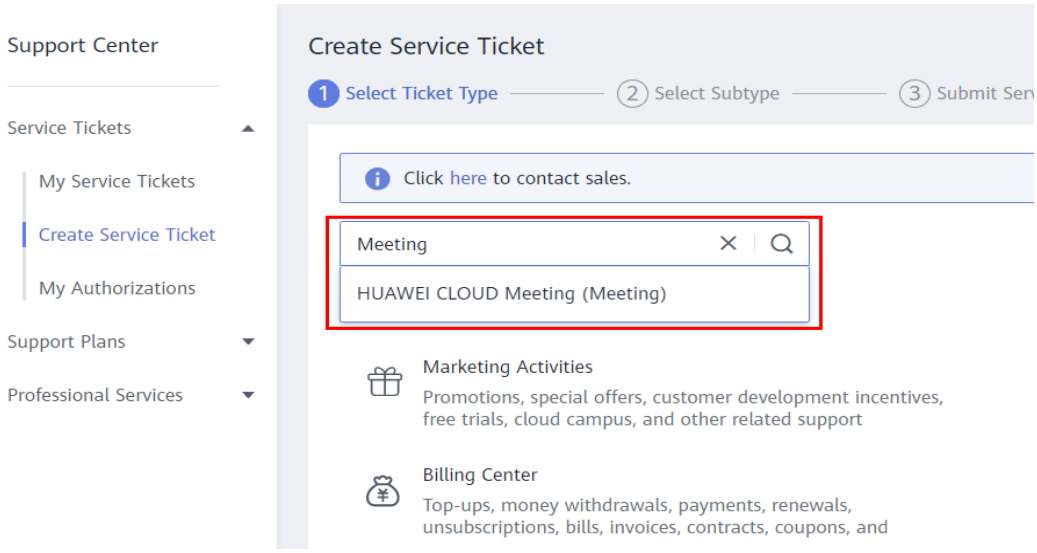
Step 2 Hover the cursor over your account name, and select **Service Tickets**.

Figure 7-1 Selecting Service Tickets



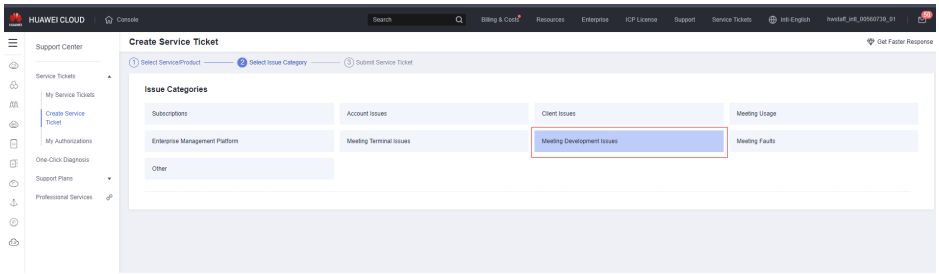
Step 3 Enter **Meeting** in the search box and click **HUAWEI CLOUD Meeting (Meeting)** from the drop-down list.

Figure 7-2 Searching for Meeting



Step 4 On the page displayed, select **Meeting Development Issues** and click **Create Service Ticket**.

Figure 7-3 Selecting subtype



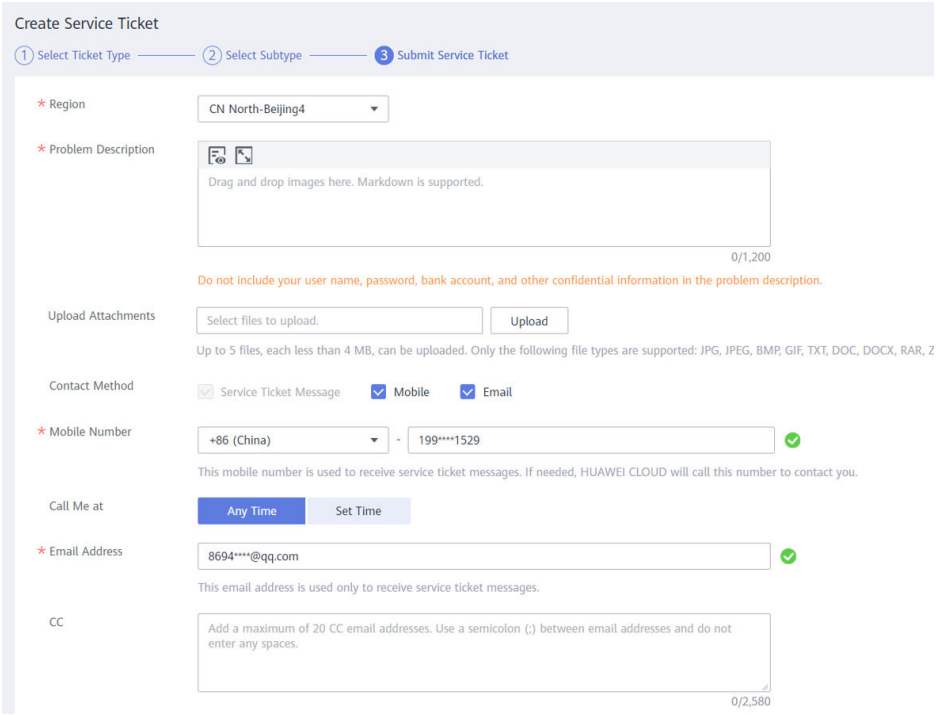
Step 5 Enter the information as prompted and submit the service ticket.

Describe the issue in detail and specify whether the issue occurs in server integration or client integration. Upload logs as attachments to help support engineer to quickly locate fault.

NOTE

Retain only the logs generated when the fault occurs and delete the logs generated before the fault occurs. It is advised to obtain logs as follows: Clear the previous log files > Reproduce the problem > Obtain the log files.

Figure 7-4 Filling service ticket information



----End

