Speech Interaction Service

API Reference

Issue	01
Date	2025-09-12

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property 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 Cloud 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 Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 Before You Start

1.1 Overview

Welcome to *Speech Interaction Service API Reference*.

Speech Interaction Service (SIS) lets you get instant responses by accessing and calling APIs in real time.

The APIs provided by SIS are proprietary APIs.

Table 1-1 Real-Time Automatic Speech Recognition (RASR) API

API	Description
RASR	This is a WebSocket API provided by Huawei Cloud, and is used for real-time speech recognition. An audio file is transmitted by fragments. The server can return the intermediate results during the recognition process and the final recognition result after the process is complete.

Table 1-2 Short Sentence Recognition API

API	Description
Short Sentence Recognition	This API is used for real-time recognition of short sentences. The entire audio is uploaded at a time, and the recognition result is returned immediately.

Table 1-3 Real-Time TTS

API	Description
Real-Time TTS	Real-Time TTS transforms text into natural-sounding speech using cutting-edge voice technology and deep learning algorithms. You can access and call APIs in real time to generate audio from your input text. By selecting timbres and customizing volume, speed, and pitch, you can tailor the audio format, offering personalized pronunciation services for both businesses and individuals.

Table 1-4 Hot word management API

API	Description
Hot Word Management	If there are specific terms in your business domain with subpar default recognition performance, consider utilizing hot word management to incorporate these terms into the lexicon, thereby enhancing recognition accuracy.

1.2 API Calling

SIS supports Representational State Transfer (REST) APIs, allowing you to call APIs using HTTPS. For details about API calling, see **Calling REST APIs**.

To obtain required audio examples, go to **Audio Examples**.

NOTE

You can directly cal the API without enabling any service.

Only pay-per-use billing is supported.

1.3 Endpoints

An endpoint is the request address for calling an API. Endpoints vary depending on services and regions.

Table 1-5 Short Sentence Recognition

Region	Endpoint Region	Endpoint	Protocol
AP-Singapore	ap-southeast-3	sis-ext.ap-southeast-3.myhuaweicloud.asia sis-ext.ap-southeast-3.myhuaweicloud.com	HTTPS
ME-Riyadh	me-east-1	sis-ext.me-east-1.myhuaweicloud.asia sis-ext.me-east-1.myhuaweicloud.com	HTTPS

Table 1-6 RASR

Region	Endpoint Region	Endpoint	Protocol
AP-Singapore	ap-southeast-3	sis-ext.ap-southeast-3.myhuaweicloud.asia sis-ext.ap-southeast-3.myhuaweicloud.com	Websocket
ME-Riyadh	me-east-1	sis-ext.me-east-1.myhuaweicloud.asia sis-ext.me-east-1.myhuaweicloud.com	Websocket

Table 1-7 Real-Time TTS

Region	Endpoint Region	Endpoint	Protocol
ME-Riyadh	me-east-1	sis-ext.me-east-1.myhuaweicloud.asia sis-ext.me-east-1.myhuaweicloud.com	Websocket

Table 1-8 Hot Word Management

Region	Endpoint Region	Endpoint	Protocol
AP-Singapore	ap-southeast-3	sis-ext.ap-southeast-3.myhuaweicloud.asia sis-ext.ap-southeast-3.myhuaweicloud.com	HTTPS
ME-Riyadh	me-east-1	sis-ext.me-east-1.myhuaweicloud.asia sis-ext.me-east-1.myhuaweicloud.com	HTTPS

1.4 Basic Concepts

- Account

  An account is created upon successful registration with Huawei Cloud. The account has full access permissions for all of its cloud services and resources. It can be used to reset user passwords and grant user permissions. The account is a payment entity. You should not directly use an account to perform routine management. For security purposes, create users and grant them permissions for routine management.

- User

  A user is created in IAM using an account, for access to cloud services. Each user has their own identity credentials (password and access keys).

  You can view the account ID and user ID on the **My Credentials** page of the console. The account name, username, and password will be required for API authentication.

- Region

  Regions are divided based on geographical location and network latency. Public services, such as Elastic Cloud Server (ECS), Elastic Volume Service (EVS), Object Storage Service (OBS), Virtual Private Cloud (VPC), Elastic IP (EIP), and Image Management Service (IMS), are accessible within the same region. Regions are classified as universal regions and dedicated regions. A universal region provides universal cloud services for common tenants. A dedicated region provides specific services for specific tenants.

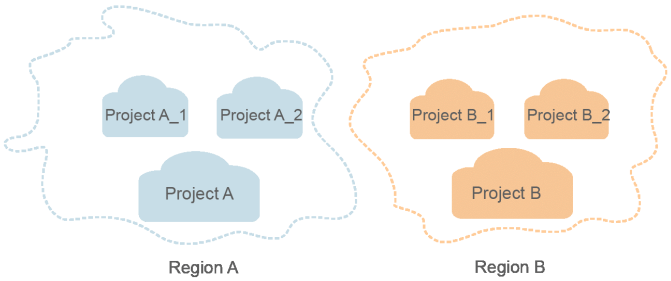
- Availability Zone (AZ)

  An availability zone (AZ) contains one or more physical data centers. Each AZ has independent cooling, fire extinguishing, moisture-proofing, and electricity facilities. Within an AZ, computing, network, storage, and other resources are logically divided into multiple clusters. AZs within a region are interconnected using high-speed optical fibers to support cross-AZ high-availability systems.

- Project

Projects isolate resources (including compute, storage, and network resources) across physical regions. A default project is provided for each Huawei Cloud region, and subprojects can be created under each default project. Users can be granted permissions to access all resources in a specific project. For more refined access control, create subprojects under a project and purchase resources in the subprojects. Users can then be assigned permissions to access only specific resources in the subprojects.

Figure 1-1 Project isolation model



2 API Overview

SIS supports Representational State Transfer (REST) APIs, allowing you to call APIs using HTTPS. For details, see **Table 2-1**.

Table 2-1 RESTful API functions

API	Function	API URI
Short Sentence Recognition	**Short Sentence Recognition**	POST /v1/{project_id}/asr/short-audio
Hot Word Management	**Hot Word Management APIs**	Create a hot word table: **POST /v1/{project_id}/asr/vocabularies** Update a hot word table: **PUT /v1/{project_id}/asr/vocabularies/{vocabulary_id}** Query a hot word table: **GET /v1/{project_id}/asr/vocabularies/{vocabulary_id}** Delete a hot word table: **DELETE /v1/{project_id}/asr/vocabularies/{vocabulary_id}** Query a hot word table list: **GET /v1/{project_id}/asr/vocabularies**

Table 2-2 WebSocket API functions

API	Function	API URI
RASR (Request)	Starting Recognition	The following three request modes are supported: • **Streaming** WSS /v1/{project_id}/rasr/short-stream • **Continuous** WSS /v1/{project_id}/rasr/continue-stream • **Single-Sentence** WSS /v1/{project_id}/rasr/sentence-stream
	Sending Audio Data	
	Ending Recognition	
Real-Time TTS (Request)	WebSocket Handshake Requests	WSS /v1/{project_id}/rtts

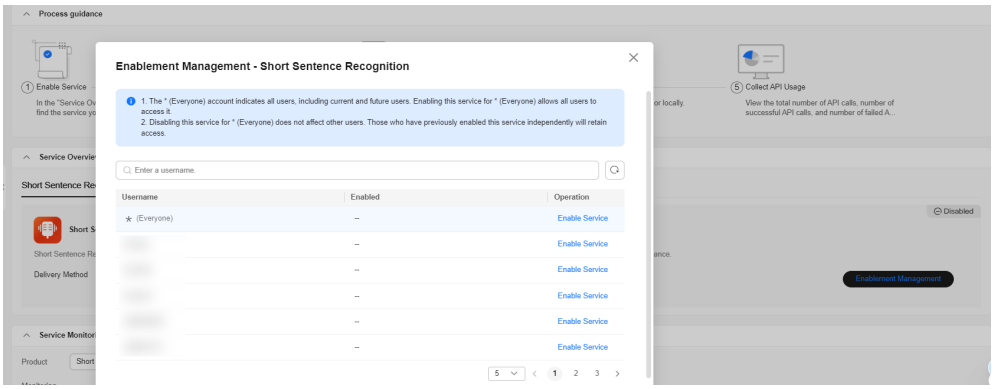
3 Applying for SIS

Enabling SIS

You can enable SIS as follows:

1. Log in to the **SIS console**.
2. On the **Service Overview** section on the **Overview** page, find the service you need, click **Enablement Management**, and enable it. It uses a pay-per-use billing mode by default. No charges will apply until you begin using the service.

   Figure 3-1 Enabling a service

   You have successfully enabled the service when **Enabled** is displayed.
3. Then, you can continue with subsequent operations according to the official documents.

CAUTION

- For users who have never used SIS (new users), they must first enable the service on the SIS console before use. The default billing mode is pay-per-use, meaning **no fees are incurred until the service is called**. Accounts with overdue payments will be restricted from making API calls, and the service remains inaccessible until enabled.
- SIS has been available for some time. Prior to supporting enablement management, certain users (legacy users) had already accessed SIS APIs. To maintain user experience continuity, all users under such legacy accounts were automatically granted access to SIS (**limited to projects where SIS APIs were previously utilized**). If you want to use this enablement management function, contact Huawei Cloud customer service. Processing typically requires approximately one business day, requiring submission of your account ID, project ID, and the user IDs of individuals requiring access. Once enabled, you can manage service enablement for individual IAM users, enabling or disabling their access as needed. IAM users without service access will be unable to call the relevant sub-service APIs.
- Legacy users can also enable SIS enablement management by creating new projects. For details about how to switch between projects, see **Switching Projects**.
- SIS enablement is **project-specific**. For example, if SIS is enabled for project A but not for project B, the service will remain unavailable in project B.
- Post-July 2024, users who have not called any SIS APIs fall into the category of new users. If you encounter the "not subscribe this service" error, enable the required service through a main account.
- With the introduction of enablement management, users cannot use a project ID associated with region A to call SIS services in region B. For cross-region calling needs, enable the SIS service under the project ID corresponding to the target region and use that project ID to call SIS APIs.

Disabling SIS

If you no longer need SIS after enablement, you can disable it through the console (currently supported only in the **AP-Singapore** region).

If you are the owner of a main account, you can enable or disable SIS for IAM users under your account.

Figure 3-2 Disabling SIS

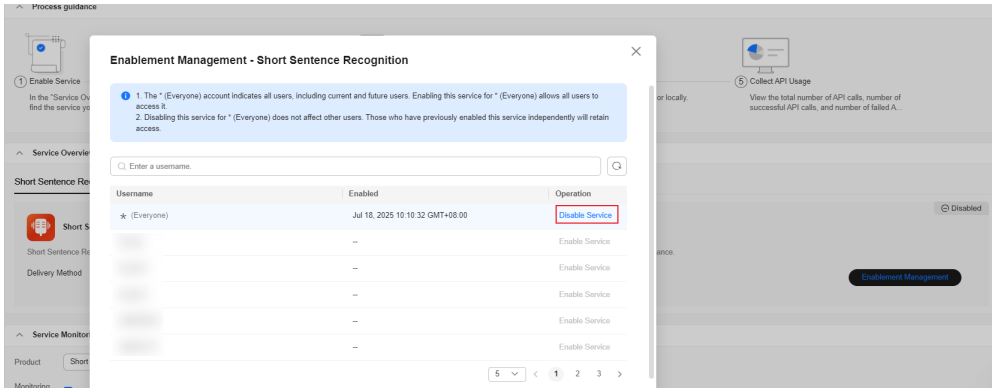
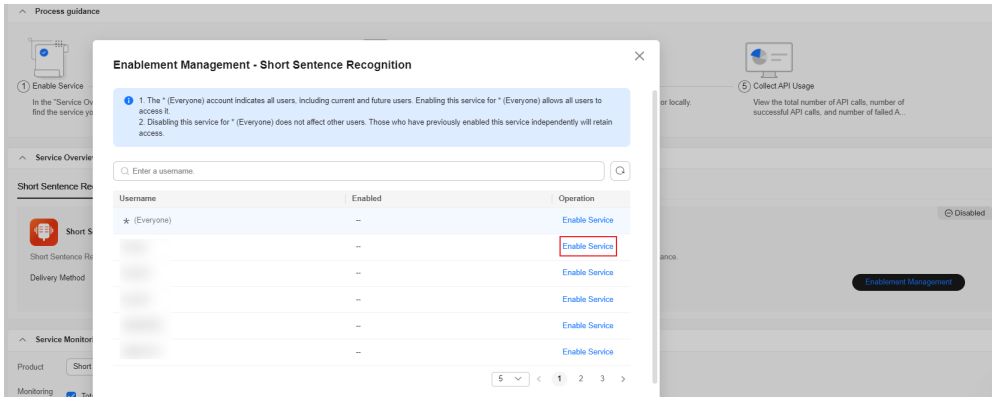


Figure 3-3 Enabling or disabling an SIS service for the IAM users under your account



4 Calling REST APIs

4.1 Making an API Request

This section describes the structure of a REST API request, and uses Short Sentence Recognition as an example.

Request URI

A request URI is in the following format:

{URI-scheme} :// {Endpoint} / {resource-path} ? {query-string}

Table 4-1 Request URI

Parameter	Description
URI-scheme	Protocol used to transmit requests. All APIs use HTTPS.
Endpoint	Domain name or IP address of the server bearing the REST service. The endpoint varies between services in different regions. It can be obtained from **Endpoints**. For example, the endpoint of Sentence Transcription in the **AP-Singapore** region is **sis-ext.ap-southeast-3.myhuaweicloud.com**.
resource-path	Access path of an API for performing a specified operation. Obtain the path from the URI of an API. For example, the **resource-path** of the Sentence Transcription API is **/v1/{project_id}/asr/short-audio**. Replace **project_id** with the user's actual project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.
query-string	Query parameter, which is optional. Ensure that a question mark (?) is included before each query parameter that is in the format of *"Parameter name=Parameter value"*.

For example, to call the **Short Sentence Recognition** API in the **AP-Singapore** region, use the endpoint (sis-ext.ap-southeast-3.myhuaweicloud.com). The combination is as follows:

```
https://sis-ext.ap-southeast-3.myhuaweicloud.com/v1/{project_id}/asr/short-audio
```

Figure 4-1 Example URI

NOTE

To simplify the URI display in this document, each API is provided only with a **resource-path** and a request method. The **URI-scheme** of all APIs is **HTTPS**, and the endpoints of all APIs in the same region are identical.

Request Methods

The HTTP protocol defines the following request methods that can be used to send a request to the server:

- **GET**: requests the server to return specified resources.
- **PUT**: requests the server to update specified resources.
- **POST**: requests the server to add resources or perform special operations.
- **DELETE**: requests the server to delete specified resources, for example, an object.
- **HEAD**: same as GET except that the server must return only the response header.
- **PATCH**: requests the server to update partial content of a specified resource. If the resource does not exist, a new resource will be created.

For example, in the case of the **Short Sentence Recognition** API, the request method is **POST**. The request is as follows:

```
POST https://sis-ext.ap-southeast-3.myhuaweicloud.com/v1/{project_id}/asr/short-audio
```

Request Header

You can also add additional header fields to a request, such as the fields required by a specified URI or HTTP method. For example, to request for the authentication information, add **Content-Type**, which specifies the request body type.

Common request headers are as follows:

- **Content-Type**: specifies the request body type or format. This field is mandatory and its default value is **application/json**.
- **X-Auth-Token**: specifies a user token only for token-based API authentication. For details about user tokens, see **Token-based Authentication** in **Authentication**.

The following provides an example request with a header included.


```
POST https://sis-ext.ap-southeast-3.myhuaweicloud.com/v1/{project_id}/asr/short-audio
Content-Type: application/json
X-Auth-Token: MIINRwYJKoZIhvcNAQcCoIINOD...
```

Request Body

The body of a request is often sent in a structured format as specified in the **Content-Type** header field. The request body transfers content except the request header. If the request body contains Chinese characters, these characters must be coded in UTF-8.

The request body varies between APIs. Some APIs do not require the request body, such as the APIs requested using the GET and DELETE methods.

In the case of the **Short Sentence Recognition** API, the request parameters and parameter descriptions can be obtained from the API request. The following is an example request with the body added. The **data** parameter indicates the Base64-encoded character string converted from the audio.

```
POST https://sis-ext.ap-southeast-3.myhuaweicloud.com/v1/{project_id}/asr/short-audio
Content-Type: application/json
X-Auth-Token: MIINRwYJKoZIhvcNAQcCoIINOD...

{
   "data": "encode audio by Base64",
   "config": {
      "audio_format": "wav",

      "property": "english_16k_common"
   }
}
```

If all data required for the API request is available, you can send the request to call the API through **curl**, **Postman**, or coding. For the **Short Sentence Recognition** API, you can obtain the request parameters and parameter descriptions from the response message.

4.2 Authentication

Requests for calling an API can be authenticated using either of the following methods:

- Token-based authentication: Requests are authenticated using a token.
- AK/SK-based authentication: Requests are authenticated by encrypting the request body using an AK/SK pair.

Token-based Authentication

NOTE

The validity period of a token is 24 hours. When using a token for authentication, cache it to prevent frequently calling the IAM API used to obtain a user token.

A token specifies temporary permissions in a computer system. During API authentication using a token, the token is added to requests to get permissions for calling the API.

Replace *username*, *domainname*, and *project name* with the actual values. You can log in to the console and choose **My Credential** to obtain the values. ***password*** indicates the user password.

- Pseudo-code

```
POST https://iam.ap-southeast-3.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json

{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",
                    "password": "********",
                    "domain": {
                        "name": "domainname"
                    }
                }
            }
        },
        "scope": {
            "project": {
                "name": "projectname"
            }
        }
    }
}
```

- Python

```
import requests
import json

url = "https://iam.ap-southeast-3.myhuaweicloud.com/v3/auth/tokens"
payload = json.dumps({
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",
          "password": "********",
          "domain": {
            "name": "domainname"
          }
        }
      }
    },
    "scope": {
      "project": {
        "name": "projectname"
      }
    }
  }
})
headers = {
  'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.headers["X-Subject-Token"])
```


The **X-Auth-Token** header field must be included to carry the token when calling other APIs. For example, if the token is **ABCDEFJ....**, **X-Auth-Token: ABCDEFJ....** can be added to a request as follows:

```
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

AK/SK-based Authentication

NOTE

AK/SK-based authentication supports API requests with a body not larger than 12 MB. For API requests with a larger body, token-based authentication is recommended.

In AK/SK-based authentication, AK/SK is used to sign requests and the signature is then added to the requests for authentication.

- AK: access key ID, which is a unique identifier used in conjunction with a secret access key to sign requests cryptographically.
- SK: secret access key used in conjunction with an AK to sign requests cryptographically. It identifies a request sender and prevents the request from being modified.

In AK/SK-based authentication, you can use an AK/SK to sign requests based on the signature algorithm or use the signing SDK to sign requests. For details about how to sign requests and use the signature SDK, see **API Request Signing Guide**.

NOTICE

The signing SDK is only used for signing requests and is different from the SDKs provided by services.

For details about how to obtain the AK/SK, see **Obtaining an AK/SK Pair**.

4.3 Response

Status Code

After sending a request, you will receive a response, including a status code, response header, and response body.

A status code is a group of digits, ranging from 1xx to 5xx. It indicates the status of a request. For more information, see **Status Codes**.

If status code **200** is returned for the calling of SIS APIs, the request is successful.

Response Header

Similar to a request, a response also has a header, for example, **Content-type**. The response headers of SIS can be used for fault locating.

Response Body

The body of a response is often returned in structured format as specified in the **Content-Type** header field. The response body transfers content except the response header.

The following shows part of the response body for the **Short Sentence Recognition** API. For details about the format, see the **Short Sentence Recognition** responses.

```
{
  "trace_id": "567e8537-a89c-13c3-a882-826321939651",
  "result":{
    "text": "Welcome to the voice cloud service.",
    "score": 0.9
  }
}
```

If an error occurs during API calling, an error code and a message will be displayed. The following shows an error response body.

```
{
    "error_msg": "***",
    "error_code": "SIS.0001"
}
```

In the response body, **error_code** is an error code, and **error_msg** provides information about the error.

5 Real-Time ASR

5.1 API Description

The Real-Time ASR API complies with the WebSocket protocol, and provides three modes: **Streaming**, **Continuous**, and **Single-sentence**.

The handshake request wss-URI of each mode differs, but the formats of requests and responses are the same.

Developers can use a WebSocket software package or library interface in the Java, Python, or C++ language to handshake with the Real-Time ASR engine, send the speech data, and obtain the recognition result. After the recognition work is done, close the WebSocket connection.

- For details about how to obtain the WebSocket handshake request wss-URI, see **WebSocket Handshake Requests**.
- For details about the format of a Real-Time ASR request, see **Real-Time ASR Requests**.
- For details about the format of a Real-Time ASR response, see **Real-Time ASR Responses**.

When a client uses the WebSocket protocol to access the RASR APIs, the connection can last 5 hours at most. If the WebSocket connection lasts more than 5 hours, the server automatically closes it.

5.2 WebSocket Handshake Requests

5.2.1 Streaming

Function

The speech used for streaming recognition must be shorter than one minute. This mode is applicable to dialog recognition.

This API allows you to input speech segments in streaming mode and obtain the final result soon after the recognition work is complete. After a speech segment is

input, the Real-Time ASR engine can immediately decode it and extract its features rather than wait until all segments are input. Therefore, after the last speech segment is input, you can get the final result after only a short period of time (the time for processing the last speech segment + the time for generating the final result). The streaming mode shortens the overall time for obtaining the final result and greatly improves user experience.

wss-URI

- wss-URI format

  wss /v1/{project_id}/rasr/short-stream

- Parameter descriptions

  Table 5-1 Parameter descriptions

Parameter	Mandatory	Description
project_id	Yes	Project ID. For details about how to obtain a project ID, see **Obtaining a Project ID**.

  Table 5-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

- Example request (pseudocode)

```
wss://{endpoint}/v1/{project_id}/rasr/short-stream

Request header:
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
```

  NOTE

  The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

- Example Python 3 request

```
# -*- coding: utf-8 -*-
# This demo is used only for tests. You are advised to use the SDK. The websocket-client must be installed in advance by running the pip install websocket-client command.
import websocket
import threading
import time
import json
```



```
def rasr_demo():
    url = 'wss://{{endpoint}}/v1/{{project_id}}/rasr/short-stream'  # Replace endpoint and project_id
with the actual values.
    audio_path = 'Audio path'
    token ='Token of the region to which the user belongs'
    header = {
        'X-Auth-Token': token
    }
    with open(audio_path, 'rb') as f:
        data = f.read()
    body = {
        'command': 'START',
        'config': {
            'audio_format': 'pcm8k16bit',
            'property': 'chinese_8k_general'
        }
    }
    def _on_message(ws, message):
        print(message)
    def _on_error(ws, error):
        print(error)
    ws = websocket.WebSocketApp(url, header, on_message=_on_message, on_error=_on_error)
    _thread = threading.Thread(target=ws.run_forever, args=(None, None, 30, 20))
    _thread.start()
    time.sleep(1)
    ws.send(json.dumps(body), opcode=websocket.ABNF.OPCODE_TEXT)
    now_index = 0
    byte_len = 4000
    while now_index < len(data):
        next_index = now_index + byte_len
        if next_index > len(data):
            next_index = len(data)
        send_array = data[now_index: next_index]
        ws.send(send_array, opcode=websocket.ABNF.OPCODE_BINARY)
        now_index += byte_len
        time.sleep(0.05)
    ws.send("{\"command\": \"END\", \"cancel\": \"false\"}", opcode=websocket.ABNF.OPCODE_TEXT)
    time.sleep(10)
    ws.close()
if __name__ == '__main__':
    rasr_demo()
```

- Example Java request

```
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import okhttp3.WebSocket;
import okhttp3.WebSocketListener;
import okio.ByteString;
import java.net.URL;
/**
* This demo is used only for tests. You are advised to use the SDK.
 * The okhttp.jar and okio.jar files have been configured. You can download the JAR files from the
SDK.
 */
public class RasrDemo {
  public void rasrDemo() {
    try {
      // Replace endpoint and projectId with the actual values.
      String url = "wss://{{endpoint}}/v1/{{project_id}}/rasr/short-stream";
      String token = "Token of the corresponding region";
      byte[] data = null; //Byte array that stores the audio to be sent
      OkHttpClient okHttpClient = new OkHttpClient();
      Request request = new Request.Builder().url(url).header("X-Auth-Token", token).build();
      WebSocket webSocket = okHttpClient.newWebSocket(request, new MyListener());
      webSocket.send("{\"command\": \"START\", \"config\": {\"audio_format\": \"pcm8k16bit\",
\"property\": \"chinese_8k_general\"}}");
      webSocket.send(ByteString.of(data));
      webSocket.send("{  \"command\": \"END\",  \"cancel\": false}");
```



```
      Thread.sleep(10000);
      webSocket.close(1000, null);

    } catch (Exception e) {
      e.printStackTrace();
    }


  }

  class MyListener extends WebSocketListener {
    @Override

    public void onOpen(WebSocket webSocket, Response response) {
      System.out.println("conneected");
    }
    @Override
    public void onClosed(WebSocket webSocket, int code, String reason) {
      System.out.println("closed");
    }
    @Override
    public void onFailure(WebSocket webSocket, Throwable t, Response response) {
      t.printStackTrace();
    }
    @Override
    public void onMessage(WebSocket webSocket, String text) {
      System.out.println(text);
    }
  }
  public static void main(String[] args) {
    RasrDemo rasrDemo = new RasrDemo();
    rasrDemo.rasrDemo();
  }
}
```

- JavaScript (Node.js v18.20.2 (LTS) is recommended.)

```
// Import the ws library of Node.js.
const WebSocket = require('ws');
function shortStreamDemo(endpoint,audioPath, projectID, token) {
    const url = `wss://${endpoint}/v1/${projectID}/rasr/short-stream`;  // Replace endpoint and
projectID with the actual values.
    // Read the content of the audio file.
    const fs = require('fs');
    let data = fs.readFileSync(audioPath);
    //Add the token to the HTTP header.
    const headers = {
        'X-Auth-Token': token,
        // Optionally add the enterprise ID.
        // 'Enterprise-Project-Id': Enterprise ID
    };
    // Create a WebSocket instance.
    const ws = new WebSocket(url, {
        headers // Add a custom HTTP header.
    });
    ws.on('open', async () => {
        const body = {
            command: 'START',
            config: {
                audio_format: 'pcm16k16bit',
                property: 'chinese_16k_general'
            }
        };
        ws.send(JSON.stringify(body));
        let nowIndex = 0;
        const byteLen = 3200; // Empty values are not allowed. Recommended range: 2000 to 10000.
        while (nowIndex < data.length) {
            const nextIndex = nowIndex + byteLen;
            const sendArray = data.slice(nowIndex, nextIndex > data.length ? data.length : nextIndex);
            ws.send(sendArray, { binary: true });
            nowIndex += byteLen;
```



```
            await new Promise(resolve => setTimeout(resolve, 100)); // Simulate a delay (unit: ms).
        }
        const endCommand = JSON.stringify({ command: 'END', cancel: 'false' });
        ws.send(endCommand);
    });
    ws.on('message', (data) => {
        if (data instanceof Buffer) {
            // Convert the Buffer to a UTF-8 encoded string.
            const messageString = data.toString('utf8');
            console.log('Received (converted from Buffer):', messageString);
            const type = JSON.parse(messageString).resp_type;
            if (type ==='END'|| type ==='ERROR') {
                ws.close();
            }
        }
    });
    ws.on('error', (error) => {
        console.error('WebSocket Error:', error);
    });
};
shortStreamDemo(endpoint,audioPath, projectID, token);
```

5.2.2 Continuous

Function

The speech used for continuous recognition must be shorter than five hours. This mode is applicable to meeting, lecture, and live video recognition.

It combines streaming recognition with endpoint detection. In this mode, the speech data is also input by segment, but endpoint detection is performed before the engine processes the data. If speech is detected, decoding work starts. If nothing (muting) is detected, the mute segment will be discarded. If the end point of a speech segment is detected, the segment's recognition result is directly returned, and then the engine continues to recognize subsequent speech data. Therefore, in the continuous recognition mode, the recognition results may be returned multiple times. If a long speech segment is sent, multiple recognition results may be returned at a time.

With the muting detection function, continuous recognition generally has a higher efficiency than streaming recognition, because the feature extraction and decoding operations are not performed on the mute segments, resulting in higher CPU efficiency. The streaming mode is usually combined with the endpoint detection function of the client, so only the detected valid speech segments are uploaded to the server for recognition.

wss-URI

- wss-URI format

  wss /v1/{project_id}/rasr/continue-stream
- Parameter descriptions

Table 5-3 Parameter descriptions

Parameter	Mandatory	Description
project_id	Yes	Project ID. For details about how to obtain a project ID, see **Obtaining a Project ID**.

Table 5-4 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

- Example request (pseudocode)

```
wss://{endpoint}/v1/{project_id}/rasr/continue-stream

Request header:
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
```

NOTE

The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

- Example Python 3 request

```
# -*- coding: utf-8 -*-
# This demo is used only for tests. You are advised to use the SDK. The websocket-client must be installed in advance by running the pip install websocket-client command.
import websocket
import threading
import time
import json

def rasr_demo():
    url = 'wss://{{endpoint}}/v1/{{project_id}}/rasr/continue-stream'  # Replace endpoint and project_id with the actual values.
    audio_path = 'Audio path'
    token ='Token of the region to which the user belongs'
    header = {
        'X-Auth-Token': token
    }
    with open(audio_path, 'rb') as f:
        data = f.read()
    body = {
        'command': 'START',
        'config': {
            'audio_format': 'pcm8k16bit',
            'property': 'chinese_8k_general'
        }
    }
```



```
    def _on_message(ws, message):
        print(message)
    def _on_error(ws, error):
        print(error)
    ws = websocket.WebSocketApp(url, header, on_message=_on_message, on_error=_on_error)
    _thread = threading.Thread(target=ws.run_forever, args=(None, None, 30, 20))
    _thread.start()
    time.sleep(1)
    ws.send(json.dumps(body), opcode=websocket.ABNF.OPCODE_TEXT)
    now_index = 0
    byte_len = 4000
    while now_index < len(data):
        next_index = now_index + byte_len
        if next_index > len(data):
            next_index = len(data)
        send_array = data[now_index: next_index]
        ws.send(send_array, opcode=websocket.ABNF.OPCODE_BINARY)
        now_index += byte_len
        time.sleep(0.05)
    ws.send("{\"command\": \"END\", \"cancel\": \"false\"}", opcode=websocket.ABNF.OPCODE_TEXT)
    time.sleep(10)
    ws.close()
if __name__ == '__main__':
    rasr_demo()
```

- Example Java request

```
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import okhttp3.WebSocket;
import okhttp3.WebSocketListener;
import okio.ByteString;
import java.net.URL;

/**
 * This demo is for testing purposes only. You are advised to use the SDK.
 * You need to configure the okhttp and okio JAR files first by downloading from the SDK.
 */
public class RasrDemo {
  public void rasrDemo() {
    try {
      // Replace endpoint and projectId with the actual values.
      String url = "wss://{{endpoint}}/v1/{{project_id}}/rasr/continue-stream";
      String token = "Token of the corresponding region";
      byte[] data = null; //Byte array that stores the audio to be sent
      OkHttpClient okHttpClient = new OkHttpClient();
      Request request = new Request.Builder().url(url).header("X-Auth-Token", token).build();
      WebSocket webSocket = okHttpClient.newWebSocket(request, new MyListener());
      webSocket.send("{\"command\": \"START\", \"config\": {\"audio_format\": \"pcm8k16bit\",
\"property\": \"chinese_8k_general\"}}");
      webSocket.send(ByteString.of(data));
      webSocket.send("{  \"command\": \"END\",  \"cancel\": false}");
      Thread.sleep(10000);
      webSocket.close(1000, null);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  class MyListener extends WebSocketListener {
    @Override
    public void onOpen(WebSocket webSocket, Response response) {
      System.out.println("conneected");
    }
    @Override
    public void onClosed(WebSocket webSocket, int code, String reason) {
      System.out.println("closed");
    }
    @Override
    public void onFailure(WebSocket webSocket, Throwable t, Response response) {
      t.printStackTrace();
```



```
    }
    @Override
    public void onMessage(WebSocket webSocket, String text) {
      System.out.println(text);
    }

  }
  public static void main(String[] args) {
    RasrDemo rasrDemo = new RasrDemo();
    rasrDemo.rasrDemo();
  }
}
```

- JavaScript (Node.js v18.20.2 (LTS) is recommended.)

```
// Import the ws library of Node.js.
const WebSocket = require('ws');
function continueStremDemo(endpoint,audioPath, projectID, token) {
    const url = `wss://${endpoint}/v1/${projectID}/rasr/continue-stream`;  // Replace endpoint and
projectID with the actual values.
    // Read the content of the audio file.
    const fs = require('fs');
    let data = fs.readFileSync(audioPath);
    //Add the token to the HTTP header.
    const headers = {
        'X-Auth-Token': token,
        // Optionally add the enterprise ID.
        // 'Enterprise-Project-Id': Enterprise ID
    };
    // Create a WebSocket instance.
    const ws = new WebSocket(url, {
        headers // Add a custom HTTP header.
    });
    ws.on('open', async () => {
        const body = {
            command: 'START',
            config: {
                audio_format: 'pcm16k16bit',
                property: 'chinese_16k_general'
            }
        };
        ws.send(JSON.stringify(body));
        let nowIndex = 0;
        const byteLen = 3200; // Empty values are not allowed. Recommended range: 2000 to 10000.
        while (nowIndex < data.length) {
            const nextIndex = nowIndex + byteLen;
            const sendArray = data.slice(nowIndex, nextIndex > data.length ? data.length : nextIndex);
            ws.send(sendArray, { binary: true });
            nowIndex += byteLen;
            await new Promise(resolve => setTimeout(resolve, 100)); // Simulate a delay (unit: ms).
        }
        const endCommand = JSON.stringify({ command: 'END', cancel: 'false' });
        ws.send(endCommand);
    });
    ws.on('message', (data) => {
        if (data instanceof Buffer) {
            // Convert the Buffer to a UTF-8 encoded string.
            const messageString = data.toString('utf8');
            console.log('Received (converted from Buffer):', messageString);
            const type = JSON.parse(messageString).resp_type;
            if (type ==='END' || type ==='ERROR') {
                ws.close();
            }
        }
    });
    ws.on('error', (error) => {
        console.error('WebSocket Error:', error);
    });
};
continueStremDemo(endpoint,audioPath, projectID, token);
```


5.2.3 Single-Sentence

Function

In the single-sentence mode, the end of a sentence is automatically detected. Therefore, it is applicable to scenarios that require interaction with your systems, such as outgoing call and password control.

Similar to the continuous mode, the single-sentence mode also involves endpoint detection. Mute segments are discarded directly and only speech segments can be decoded. The engine returns the recognition result after detecting a segment's end point. However, in single-sentence mode, after the recognition result of the first segment is returned, the subsequent segments are no longer recognized. This applies to scenarios involving speech interaction with users. After a user says a sentence, the system recognizes it and waits for subsequent interaction, such as listening to the feedback of your question, so it is unnecessary to recognize subsequent segments.

wss-URI

- wss-URI format

  wss /v1/{project_id}/rasr/sentence-stream
- Parameter descriptions

  Table 5-5 Parameter descriptions

Parameter	Mandatory	Description
project_id	Yes	Project ID. For details about how to obtain a project ID, see **Obtaining a Project ID**.

  Table 5-6 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

- Example request (pseudocode)

```
wss://{endpoint}/v1/{project_id}/rasr/sentence-stream

Request header:
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
```


> **NOTE**
>
> The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

- Example Python3 request

```
# -*- coding: utf-8 -*-
# This demo is for testing purposes only. You are advised to use the SDK. You need to install websocket-client first by running pip install websocket-client.
import websocket
import threading
import time
import json

def rasr_demo():
    url = 'wss://{{endpoint}}/v1/{{project_id}}/rasr/sentence-stream'  # Replace endpoint and project_id with the actual values.
    audio_path = 'Audio path'
    token ='Token of the region to which the user belongs'
    header = {
        'X-Auth-Token': token
    }
    with open(audio_path, 'rb') as f:
        data = f.read()
    body = {
        'command': 'START',
        'config': {
            'audio_format': 'pcm8k16bit',
            'property': 'chinese_8k_general'
        }
    }
    def _on_message(ws, message):
        print(message)
    def _on_error(ws, error):
        print(error)
    ws = websocket.WebSocketApp(url, header, on_message=_on_message, on_error=_on_error)
    _thread = threading.Thread(target=ws.run_forever, args=(None, None, 30, 20))
    _thread.start()
    time.sleep(1)
    ws.send(json.dumps(body), opcode=websocket.ABNF.OPCODE_TEXT)
    now_index = 0
    byte_len = 4000
    while now_index < len(data):
        next_index = now_index + byte_len
        if next_index > len(data):
            next_index = len(data)
        send_array = data[now_index: next_index]
        ws.send(send_array, opcode=websocket.ABNF.OPCODE_BINARY)
        now_index += byte_len
        time.sleep(0.05)
    ws.send("{\"command\": \"END\", \"cancel\": \"false\"}", opcode=websocket.ABNF.OPCODE_TEXT)
    time.sleep(10)
    ws.close()
if __name__ == '__main__':
    rasr_demo()
```

- Example Java request

```
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import okhttp3.WebSocket;
import okhttp3.WebSocketListener;
import okio.ByteString;
import java.net.URL;

/**
 * This demo is for testing purposes only. You are advised to use the SDK.
 * You need to configure the okhttp and okio JAR files first by downloading from the SDK.
 */
```



```
public class RasrDemo {
  public void rasrDemo() {
    try {
      // Replace endpoint and projectId with the actual values.
      String url = "wss://{{endpoint}}/v1/{{project_id}}/rasr/sentence-stream";
      String token = "Token of the corresponding region";
      byte[] data = null; //Byte array that stores the audio to be sent
      OkHttpClient okHttpClient = new OkHttpClient();
      Request request = new Request.Builder().url(url).header("X-Auth-Token", token).build();
      WebSocket webSocket = okHttpClient.newWebSocket(request, new MyListener());
      webSocket.send("{\"command\": \"START\", \"config\": {\"audio_format\": \"pcm8k16bit\",
\"property\": \"chinese_8k_general\"}}");
      webSocket.send(ByteString.of(data));
      webSocket.send("{  \"command\": \"END\",  \"cancel\": false}");
      Thread.sleep(10000);
      webSocket.close(1000, null);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

  class MyListener extends WebSocketListener {
    @Override
    public void onOpen(WebSocket webSocket, Response response) {
      System.out.println("conneected");
    }
    @Override
    public void onClosed(WebSocket webSocket, int code, String reason) {
      System.out.println("closed");
    }
    @Override
    public void onFailure(WebSocket webSocket, Throwable t, Response response) {
      t.printStackTrace();
    }
    @Override
    public void onMessage(WebSocket webSocket, String text) {
      System.out.println(text);
    }
  }
  public static void main(String[] args) {
    RasrDemo rasrDemo = new RasrDemo();
    rasrDemo.rasrDemo();
  }
}
```

- JavaScript (Node.js v18.20.2 (LTS) is recommended.)

```
// Import the ws library of Node.js.
const WebSocket = require('ws');
function sentenceStreamDemo(endpoint,audioPath, projectID, token) {
    const url = `wss://${endpoint}/v1/${projectID}/rasr/sentence-stream`;  // Replace endpoint and
projectID with the actual values.
    // Read the content of the audio file.
    const fs = require('fs');
    let data = fs.readFileSync(audioPath);
    //Add the token to the HTTP header.
    const headers = {
        'X-Auth-Token': token,
        // Optionally add the enterprise ID.
        // 'Enterprise-Project-Id': Enterprise ID
    };
    // Create a WebSocket instance.
    const ws = new WebSocket(url, {
        headers // Add a custom HTTP header.
    });
    ws.on('open', async () => {
        const body = {
            command: 'START',
            config: {
                audio_format: 'pcm16k16bit',
                property: 'chinese_16k_general'
```



```
            }
        };
        ws.send(JSON.stringify(body));
        let nowIndex = 0;
        const byteLen = 3200; // Empty values are not allowed. Recommended range: 2000 to 10000.
        while (nowIndex < data.length) {
            const nextIndex = nowIndex + byteLen;
            const sendArray = data.slice(nowIndex, nextIndex > data.length ? data.length : nextIndex);
            ws.send(sendArray, { binary: true });
            nowIndex += byteLen;
            await new Promise(resolve => setTimeout(resolve, 100)); // Simulate a delay (unit: ms).
        }
        const endCommand = JSON.stringify({ command: 'END', cancel: 'false' });
        ws.send(endCommand);
    });
    ws.on('message', (data) => {
        if (data instanceof Buffer) {
            // Convert the Buffer to a UTF-8 encoded string.
            const messageString = data.toString('utf8');
            console.log('Received (converted from Buffer):', messageString);
            const type = JSON.parse(messageString).resp_type;
            if (type ==='END'|| type ==='ERROR') {
                ws.close();
            }
        }
    });
    ws.on('error', (error) => {
        console.error('WebSocket Error:', error);
    });
};
sentenceStreamDemo(endpoint,audioPath, projectID, token);
```

5.3 Real-Time ASR Requests

5.3.1 Real-Time ASR Working Process

The working process of Real-Time ASR consists of four phases: starting recognition, sending audio data, ending recognition, and closing the connection.

- During the phase of starting recognition, a start command needs to be sent, containing configurations such as the sampling rate, audio format, and whether to return the intermediate results. The server returns a start response.
- During the phase of sending audio data, the client sends audio data in segments. The server returns the recognition results or other events, such as audio timeout and excess long mute duration.
- After the audio is sent, the client sends a request for ending audio sending, and the server returns an end response.
- In Real-Time ASR, the client must disconnect from the server proactively. If the server does not receive any data from the client after 20s, it returns an error and disconnects from the client.

5.3.2 Starting Recognition

Function

After the wss handshake request receives a successful response, the communication protocol between the client and the server is upgraded to the

WebSocket protocol. Through the WebSocket protocol, the client sends a recognition starting request for configuring related parameters.

Request Parameters

Table 5-7 Parameter descriptions

Parameter	Mandatory	Type	Description
command	Yes	String	The client sends a recognition start request. Set it to **START**.
config	Yes	Object	Configuration information. Structure information. For details, see **Table 5-8**.

Table 5-8 config data structure

Parameter	Mandatory	Type	Description
audio_format	Yes	String	Supported audio format. For details, see **Table 5-10**.
property	Yes	String	Model feature string in use. Generally, the value is in the *language_sampling rate_domain* format, for example, **chinese_8k_common**. For details, see **Table 5-9**.
add_punc	No	String	Whether to add punctuation marks to the recognition result. Possible values are **yes** and **no**. The default value is **no**.

Parameter	Mandatory	Type	Description
vad_head	No	Integer	In single-sentence mode, when real-time ASR encounters an audio segment with initial silence duration longer than or equal to this parameter value, it returns an **EXCEEDED_SILENCE** event, ending the recognition process. Conversely, in continuous mode, the system will segment the audio and proceed to recognize the subsequent sentence. This parameter does not take effect in streaming mode. If set to **0**, it is equivalent to setting it to **60000**. Range: An integer within [0, 60000], measured in milliseconds. The default value is **10000** ms, which is 10 seconds.
vad_tail	No	Integer	Silence duration at the end of the audio. In normal cases, it should not be set to a small value. If the silence duration at the end of the audio is greater than or equal to the value of this parameter, the **VOICE_END** (the recognition result is not empty) or **EXCEEDED_SILENCE** (the recognition result is empty) event is returned in single-sentence mode and the recognition ends. In continuous mode, a sentence is segmented and the recognition of the next sentence continues. This parameter does not take effect in streaming mode. Range: An integer within [0, 3000], measured in milliseconds. The default value is **500** ms. Note: If **vad_tail** is set to a small value (< 200 ms), sentence segmentation will be performed too frequently, affecting the recognition result.

Parameter	Mandatory	Type	Description
max_seconds	No	Integer	Maximum duration of a sentence. When the detected audio duration reaches or exceeds this value, in single-sentence mode, it will return either a **VOICE_END** event (the recognition result is not empty) or an **EXCEEDED_SILENCE** event (the recognition result is empty), thereby ending the recognition process. In continuous mode, it will segment the sentence and proceed with recognizing the next one. This parameter does not take effect in streaming mode. Range: An integer within [1, 60], measured in seconds. The default value is **30** seconds.
interim_results	No	String	Whether to output the intermediate result. The value can be **yes** or **no**. The default value is **no**, which indicates that the intermediate result is not output.
vocabulary_id	No	String	ID of a hot word table. If no hot word table is used, this field can be left blank. For details about how to create a hot word table, see **Creating a Hot Word Table**.

Table 5-9 Value range of **property**

Value	Description
chinese_8k_general	Supports Chinese Mandarin speech recognition at a sampling rate of 8 kHz and uses the next-gen end-to-end recognition algorithm to achieve higher recognition accuracy.
chinese_16k_general	Supports Chinese Mandarin speech recognition at a sampling rate of 16 kHz and uses the next-gen end-to-end recognition algorithm to achieve higher recognition accuracy.

Value	Description
english_16k_general	Supports English speech recognition at a sampling rate of 16 kHz and uses the next-gen end-to-end recognition algorithm to achieve higher recognition accuracy. Digit normalization (**digit_norm** parameter) is not available.
arabic_16k_general	Supports Arabic speech recognition at a sampling rate of 16 kHz and uses the next-gen end-to-end recognition algorithm to accommodate Standard Arabic, Egyptian dialect, Saudi dialect, and UAE dialect. It does not support punctuation prediction (**add_punc** parameter), digit normalization (**digit_norm** parameter), or hot words (**vocabulary_id** parameter).

Table 5-10 Value range of **audio_format**

Value	Description
pcm16k16bit	16 kHz, 16-bit mono-channel audio recording data
pcm8k16bit	8 kHz, 16-bit mono-channel audio recording data
ulaw16k8bit	16 kHz, 8-bit ulaw mono-channel audio recording data
ulaw8k8bit	8 kHz, 8-bit ulaw mono-channel audio recording data
alaw16k8bit	16 kHz, 8-bit alaw mono-channel audio recording data
alaw8k8bit	8 kHz, 8-bit alaw mono-channel audio recording data

NOTE

Currently, only RAW audio files in PCM-encoded wav format are supported. Any other formats (with a WAV header or in ARM format) are not supported.

Example

```
{
  "command": "START",
  "config":
  {
    "audio_format": "ulaw8k8bit",
```



```
    "property": "chinese_8k_general",
    "add_punc": "yes",
    "vad_tail": 400,
    "interim_results": "yes",

  }
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

5.3.3 Sending Audio Data

After receiving the "recognition starting" response, the client starts to send audio data. To save traffic, audio is sent in binary messages.

The audio data is sent by segment, so the client can send a binary message after a certain amount of audio data is obtained. It is recommended that each segment be 50 ms to 1000 ms long. When real-time feedback is required, a segment can be 100 ms long. Otherwise, a segment can be 500 ms long.

Currently, the SIS service limits the shard size of an 8 kHz audio file to [160, 32768] bytes and that of a 16 kHz audio file to [320, 65536] bytes. If the shard size exceeds the upper limit or is below the lower limit, an error is reported.

5.3.4 Ending Recognition

Function

To cancel or end the recognition of a speech being recognized, the client needs to send a "recognition ending" request over WebSocket. The "recognition ending" request is sent in a text message. The commands and parameters are in JSON character strings.

Request

Table 5-11 Parameters

Parameter	Mandatory	Type	Description
command	Yes	String	Indicates that the client ends the recognition request. Set this parameter to **END**.

Parameter	Mandatory	Type	Description
cancel	No	Boolen	Indicates whether to cancel the return of the recognition result. • **true**: Indicates that the recognition is canceled, that is, the audio data being recognized and to be recognized is discarded and the remaining results are not returned. • **false**: Continues to process the audio data being recognized and to be recognized until all input data is processed. Default value: **false**

Example

```
{
  "command": "END",
  "cancel": false
}
```

Status Code

For details about status codes, see **Status Codes**.

Error Code

For details about error codes, see **Error Codes**.

5.4 Real-Time ASR Responses

5.4.1 Responses to Recognition Start Requests

Since WebSocket operates in full-duplex mode, the response refers to messages sent from the server to the client. However, not every request has a corresponding response. Upon receiving the "recognition start" request, the server will provide the following response message, formatted as a JSON string within the text message.

Response Parameters

Table 5-12 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value of this parameter is **START**, indicating the response to a recognition starting request.
trace_id	String	Service internal token used to trace a specific process in logs

Example

```
{
   "resp_type": "START",
   "trace_id": "567e8537-a89c-13c3-a882-826321939651"
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

5.4.2 Event Responses

After detecting certain events, the server sends corresponding responses (JSON character strings) in text messages.

Response

Table 5-13 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value of this parameter is **EVENT**, indicating the response to a recognition starting request.
trace_id	String	Service internal token used to trace a specific process in logs
event	String	For detailed events, see **Value Range and Description of Parameter event**.

Parameter	Type	Description
timestamp	Integer	This is a reserved field. It will be used to indicate the time when an event occurs. The start time of a speech is 00:00. The unit is ms.

Value Range and Description of Parameter event

Table 5-14 Value range of **event**

Event	Description
VOICE_START	Detects the start of a sentence.
VOICE_END	Detects the end of a sentence.
EXCEEDED_SILENCE	The mute duration is too long, that is, no sound is detected during a certain time period.

- **In streaming mode:**
  - The **VOICE_START**, **VOICE_END**, and **EXCEEDED_SILENCE** events are not returned.
- **In single-sentence mode:**
  - When the **VOICE_START** event is returned, speech is detected and the IVR system can interrupt the recognition.
  - After the **VOICE_END** event is returned, the recognition of a sentence ends and subsequent audio segments will be ignored.
  - Only one set of **VOICE_START** and **VOICE_END** events will be returned.
  - If the **EXCEEDED_SILENCE** event is returned, no sound is detected during the period specified by **vad_head**. That is, the user did not speak. Then, subsequent audio segments will be ignored.
- **In continuous mode:**
  - The **VOICE_START**, **VOICE_END**, and **EXCEEDED_SILENCE** events are not returned.

Example

```
{
  "resp_type": "EVENT",
  "trace_id": "567e8537-a89c-13c3-a882-826321939651",
  "event": "VOICE_END",
  "timestamp": 1500
}
```

Status Code

For details about status codes, see **Status Codes**.

Error Code

For details about error codes, see **Error Codes**.

5.4.3 Responses to Recognition Results

After receiving the continuous audio data from the client, the server sends the recognition results back by sentence in real time. The result responses (JSON character strings) are put in text messages.

Response

Table 5-15 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value of this parameter is **RESULT**, indicating the recognition result response.
trace_id	String	Service internal token used to trace a specific process in logs
segments	Array of objects	Result of multiple segments. For details, see **Table 5-16**.

Table 5-16 segment data structure

Parameter	Type	Description
start_time	Integer	Relative timestamp, indicating the start of a sentence. The unit is millisecond (ms).
end_time	Integer	Relative timestamp, indicating the end of a sentence. The unit is millisecond (ms).
is_final	Boolen	Indicates whether the output is the final result. The value **true** indicates that the result is the final result, and the value **false** indicates that the result is a temporary intermediate result.
result	Object	If the calling is successful, this parameter indicates the recognition result. Otherwise, this parameter is invalid. For details, see **Table 5-17**.

Table 5-17 result data structure

Parameter	Type	Description
text	String	Recognition result
score	Float	Confidence level of the recognition result. The value ranges from 0 to 1. The value is continuously updated during real-time recognition until the final result is returned. **NOTE** The confidence level of the temporary result does not play a significant role. Do not overly depend on this value.

Example

```
{
  "resp_type": "RESULT",
  "trace_id": "b08a4603-3e1a-4f30-ae07-b2bbabe2e8d7",
  "segments": [
    {
      "start_time": 0,
      "end_time": 3159,
      "is_final": true,
      "result": {
        "text": "hello everybody",
        "score": 0.9178787469863892,
        "word_info": [
          {
            "start_time": 920,
            "end_time": 1200,
            "word": "_HE"
          },
          {
            "start_time": 1200,
            "end_time": 1380,
            "word": "LL"
          },
          {
            "start_time": 1380,
            "end_time": 1500,
            "word": "O"
          },
          {
            "start_time": 1580,
            "end_time": 2040,
            "word": "_EVERYBODY"
          }
        ]
      }
    }
  ]
}
```

Status Code

For details about status codes, see **Status Codes**.

Error Code

For details about error codes, see **Error Codes**.

5.4.4 Error Responses

Error responses report errors that do not affect the recognition process but interrupt the current session, including the following:

- Incorrect configuration strings, including those that cannot be recognized and those whose value range is invalid.
- Incorrect time sequence. For example, the "recognition starting" command is sent twice consecutively.
- Errors occurring during the recognition process. For example, an error occurs during audio decoding.

If an error response is sent during a session, a "recognition ending" response will also be sent to mark the end of the session. If the session does not start, the system does not perform other operations after sending the error response. The subsequent audio data is ignored until the next "recognition starting" request is received.

Response

Table 5-18 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value of this parameter is **ERROR**, indicating the error response.
trace_id	String	Service internal token used to trace a specific process in logs. In some cases, this field may not exist.
error_code	String	Error codes. For details, see **Error Codes**.
error_msg	String	Returned error message

Example

```
{
    "resp_type": "ERROR",
    "trace_id": "567e8537-a89c-13c3-a882-826321939651",
    "error_code": "SIS.0002",
    "error_msg": "***"
}
```

Status Code

For details about status codes, see **Status Codes**.

Error Code

For details about error codes, see **Error Codes**.

5.4.5 Fatal Error Responses

Fatal errors interrupt the whole recognition process. For example, the interval between two audio segments sent by the client times out (for example, exceed 20s).

When a fatal error occurs, the process stops and the server disconnects from the client.

Response

Table 5-19 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value of this parameter is **FATAL_ERROR**, indicating the response to a recognition starting request.
trace_id	String	Service internal token used to trace a specific process in logs
error_code	String	Error codes. For details, see **Error Codes**.
error_msg	String	Returned error message

Example

```
{
   "resp_type": "FATAL_ERROR",
   "trace_id": "567e8537-a89c-13c3-a882-826321939651",
   "error_code": "SIS.0002",
   "error_msg": "***"
}
```

Status Code

For details about status codes, see **Status Codes**.

Error Code

For details about error codes, see **Error Codes**.

5.4.6 Responses to Recognition End Requests

When the server receives a "recognition end" request or if an error occurs during the speech recognition process, the server will push the following response message to the client, formatted as a JSON string within the text message.

Response Parameters

Table 5-20 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value of this parameter is **END**, indicating the response to a recognition ending request.
trace_id	String	Service internal token used to trace a specific process in logs
reason	String	End cause. For details, see **Table 5-21**.

Table 5-21 End causes

Parameter	Description
NORMAL	The process ends normally.
CANCEL	The user cancels the process. When the client sends a "recognition ending" command, the value of parameter **cancel** is **true**.
ERROR	An error occurred during the recognition process.

Example

```
{
    "resp_type": "END",
    "trace_id": "567e8537-a89c-13c3-a882-826321939651",
    "reason": "NORMAL",
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

6 Short Sentence Recognition

6.1 HTTP Interface

Function

This API is used for real-time recognition of short sentences. Audio files within 1 minute can be uploaded at a time without segmentation, and the recognition result can be returned immediately.

URI

POST /v1/{project_id}/asr/short-audio

Table 6-1 Path parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 6-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

Table 6-3 Request body parameters

Parameter	Mandatory	Type	Description
config	Yes	**Config** object	Configuration information.
data	Yes	String	Base64-encoded voice data. The size of the Base64-encoded voice data cannot exceed 4 MB, and the audio duration cannot exceed 1 minute. For example, **/+MgxAAUeHpMAUkQAANhuRAC.....** If the prefix **data:audio/mp3;base64,** is carried, an error is reported.

Table 6-4 Config

Parameter	Mandatory	Type	Description
audio_format	Yes	String	Supported audio format. For details, see **Table 6-5**.
property	Yes	String	Model feature string in use, which is generally in the *Language_Sampling rate_Domain* format. The sampling rate must be the same as the audio sampling rate. For details about the value range, see **Table 6-6**.
add_punc	No	String	Whether to add punctuation marks to the recognition result. The value can be **yes** or **no**. The default value is **no**.

Parameter	Mandatory	Type	Description
digit_norm	No	String	Whether to convert digits in the speech into Arabic numerals. The value can be **yes** or **no**. The default value is **yes**.
vocabulary_id	No	String	Hot word table ID. If no hot word table is used, this field can be left blank. This parameter is currently not supported on the international website.
need_word_info	No	String	Whether to output the word segmentation result in the recognition result. The value can be **yes** or **no**. The default value is **no**.

Table 6-5 Value range of **audio_format**

Value	Description
pcm16k16bit	16 kHz, 16-bit mono-channel audio recording data
pcm8k16bit	8 kHz, 16-bit mono-channel audio recording data
ulaw16k8bit	16 kHz, 8-bit u-law mono-channel audio recording data
ulaw8k8bit	8 kHz, 8-bit u-law mono-channel audio recording data
alaw16k8bit	16 kHz, 8-bit A-law mono-channel audio recording data
alaw8k8bit	8 kHz, 8-bit A-law mono-channel audio recording data
mp3	MP3 audio recording data Currently, only mono-channel audio is supported.
aac	AAC audio recording data Currently, only mono-channel audio is supported.
wav	Format with the WAV encapsulation header. The format is automatically determined by the encapsulation header. Currently, only the 8 kHz/16 kHz sampling rate, mono channel, and pcm encoding format are supported.
amr	AMR narrowband (8 kHz) compressed audio recording data. Currently, only mono-channel audio is supported.
amrwb	AMR broadband (16 kHz) compressed audio recording data. Currently, only mono-channel audio is supported.

Value	Description
auto	The engine automatically determines and decodes the audio data in amr, flac, m4a, mp3, ogg, webm, wav, aac, ac3, mov, wma, and amrwb format.

Table 6-6 Value range of **property**

Value	Description
chinese_16k_general	Supports speech recognition of Chinese Mandarin with a sampling rate of 8 kHz or 16 kHz. The next-gen end-to-end recognition algorithm is used to achieve higher recognition accuracy.
arabic_16k_general	Supports Arabic speech recognition at a sampling rate of 16 kHz and supports Standard Arabic, Egyptian dialect, and Saudi dialect. The **add_punc**, **digit_norm**, and **vocabulary_id** parameters are currently not supported.
english_16k_common	Supports English speech recognition at a sampling rate of 8 kHz or 16 kHz. The **digit_norm** parameter is currently not supported.
english_8k_common	Supports English speech recognition at a sampling rate of 8 kHz. This model is an old version and will not be maintained later. You are advised to use **english_16k_common**.

Response Parameters

Status code: 200

Table 6-7 Response body parameters

Parameter	Mandatory	Type	Description
trace_id	Yes	String	Internal token used to trace a specific process in logs. This parameter is not included when the API fails to be called. In some error cases, this field may not exist.
result	Yes	**Result** object	If the calling is successful, this parameter indicates the recognition result. Otherwise, this parameter is invalid.

Table 6-8 Result

Parameter	Mandatory	Type	Description
text	Yes	String	Recognition result of a successful call
score	Yes	Float	Confidence of a successful call. The value ranges from 0 to 1.
word_info	No	Array of **WordInfo** objects	Word segmentation information list Word segmentation refers to splitting the recognized text into individual words.

Table 6-9 WordInfo

Parameter	Mandatory	Type	Description
start_time	No	Integer	Start time
end_time	No	Integer	End time
word	No	String	Word segmentation

Status code: 400

Table 6-10 Response body parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not returned for a successful call.
error_msg	String	Error message when the API call fails. This parameter is not returned for a successful call.

Example Requests

NOTE

The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

- Upload a short audio and quickly obtain the recognition result.

```
POST https://{endpoint}/v1/{project_id}/asr/short-audio

Request header:
Content-Type: application/json
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
```



```
Request body:
{
  "config":
  {
    "audio_format": "wav",
    "property": "arabic_16k_general",
    "add_punc": "yes",
    "need_word_info": "yes"
  },
  "data": "/+MgxAAUeHpMAUkQAANhuRAC..."
}
```

Example Responses

Status code: 200

Example response for a successful request

```
{
  "trace_id": "567e8537-a89c-13c3-a882-826321939651",
  "result":{
    "text": "Nice to meet you",
    "score": 0.9,
  }
}
```

Status code: 400

Example response for a failed request

```
{
    "error_code":"SIS.0001",
    "error_msg":"***"
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

7 Real-Time TTS

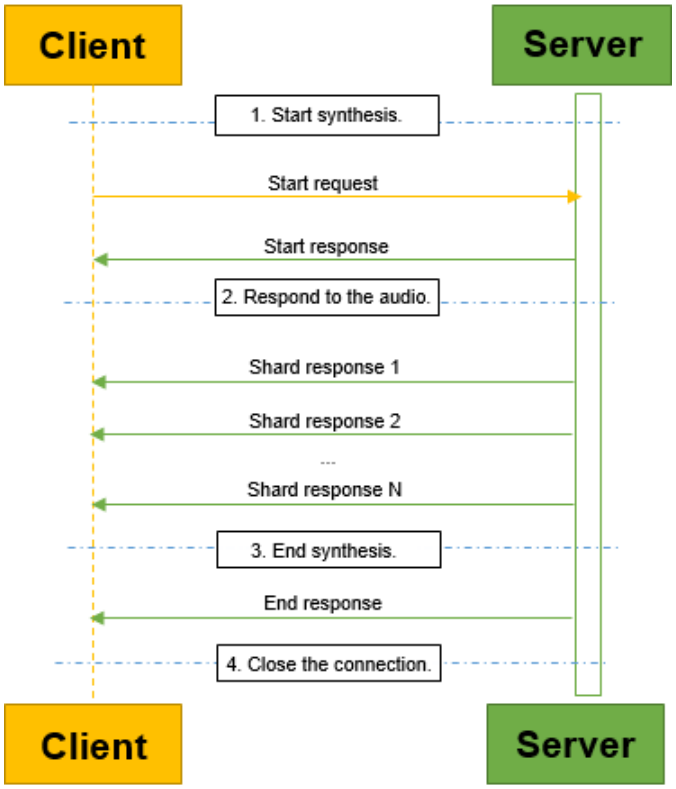
7.1 WebSocket Handshake Requests

Function

This API is used to synthesize real-time streaming voices. Users establish a connection each time they wish to synthesize text. Upon sending the text for synthesis, the server responds with the synthesized results. Only one piece of text can be sent per connection. Multiple texts require establishing separate connections for each.

Working Process

As shown in the flowchart below, only one start request needs to be sent for real-time TTS, and the start response, segment response, and end response are received in sequence.



wss-URI

- wss-URI format
  wss /v1/{project_id}/rtts
- Parameter descriptions

Table 7-1 Parameter descriptions

Parameter	Mandatory	Description
project_id	Yes	Project ID. For details about how to obtain a project ID, see **Obtaining a Project ID**.

Table 7-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

- Example request (pseudocode)

```
wss://{endpoint}/v1/{project_id}/rtts

Request header:
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
```

Example Python3 request

```
# -*- coding: utf-8 -*-
# This demo is for testing purposes only. You are advised to use the SDK. You need to install websocket-client first by running pip install websocket-client.
import websocket
import threading
import time
import json


def rtts_demo():
    url = 'wss://{{endpoint}}/v1/{{project_id}}/rtts'  # Replace endpoint and project_id with actual values.
    text = 'Text to be synthesized'
    token = 'Token of the region corresponding to the user'
    header = {
        'X-Auth-Token': token
    }

    body = {
        'command': 'START',
        'text': text,
        'config': {
            'audio_format': 'pcm',
            'property': 'chinese_xiaoyu_common',
            'sample_rate': '8000'
        }
    }

    def _on_message(ws, message):
        if isinstance(message, bytes):
            print('receive data length %d' % len(message))
        else:
            print(message)

    def _on_error(ws, error):
        print(error)

    ws = websocket.WebSocketApp(url, header, on_message=_on_message, on_error=_on_error)
    _thread = threading.Thread(target=ws.run_forever, args=(None, None, 30, 20))
    _thread.start()
    time.sleep(1)
    ws.send(json.dumps(body), opcode=websocket.ABNF.OPCODE_TEXT)
    time.sleep(10)
    ws.close()


if __name__ == '__main__':
    rtts_demo()
```

Example Java request

```
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import okhttp3.WebSocket;
import okhttp3.WebSocketListener;
import okio.ByteString;

/**
 * This demo is for testing purposes only. You are advised to use the SDK.
 * You need to configure the okhttp and okio JAR files first by downloading from the SDK.
```



```
*/
public class RttsDemo {
  public void rttsDemo() {
    try {
      // Replace endpoint and projectId with the actual values.
      String url = "wss://{{endpoint}}/v1/{{project_id}}/rtts";
      String token = "Token of the corresponding region";
      String text = "Text to be synthesized";
      OkHttpClient okHttpClient = new OkHttpClient();
      Request request = new Request.Builder().url(url).header("X-Auth-Token", token).build();
      WebSocket webSocket = okHttpClient.newWebSocket(request, new MyListener());
      webSocket.send("{\"command\": \"START\", \"text\":\"" + text
          + "\", \"config\": {\"audio_format\": \"pcm\", \"property\": \"chinese_xiaoyu_common\"}}");
      Thread.sleep(10000);
      webSocket.close(1000, null);

    } catch (Exception e) {
      e.printStackTrace();
    }

  }

  class MyListener extends WebSocketListener {
    @Override

    public void onOpen(WebSocket webSocket, Response response) {
      System.out.println("conneected");
    }

    @Override
    public void onClosed(WebSocket webSocket, int code, String reason) {
      System.out.println("closed");
    }

    @Override
    public void onFailure(WebSocket webSocket, Throwable t, Response response) {
      t.printStackTrace();
    }

    @Override
    public void onMessage(WebSocket webSocket, String text) {
      System.out.println(text);
    }

    public void onMessage(WebSocket webSocket, ByteString bytes) {
      byte[] data = bytes.toByteArray();
      System.out.println("receive data length is " + data.length);
    }

  }

  public static void main(String[] args) {
    RttsDemo rttsDemo = new RttsDemo();
    rttsDemo.rttsDemo();
  }
}
```

7.2 Real-Time TTS Requests

7.2.1 Request for Starting TTS

Function

After establishing a WebSocket connection with the TTS engine, the client can send a start TTS request to initiate TTS. If the client sends multiple synthesis requests over the same WebSocket connection, the WebSocket connection must be re-established for each request, as one connection can handle only a single synthesis request at a time.

Request Parameters

Table 7-3 Parameter descriptions

Parameter	Type	Mandatory	Description
command	String	Yes	Set it to **START** to initiate the recognition request.
text	String	Yes	Text to be synthesized, which can contain up to 10,000 characters.
config	Object	No	Configuration information. For details, see **Table 7-4**.

Table 7-4 config data structure

Parameter	Type	Mandatory	Description
audio_format	String	No	Audio format header. Option: **pcm**. Default value: **pcm**.
sample_rate	String	No	Sampling rate. Options: **16000** and **8000**. Default value: **8000**
property	String	Yes	For details, see **Table 7-5**. For speakers with high-quality pronunciation, the charge is made every 50 characters.
subtitle	String	No	Whether to generate timestamp information. Leave this parameter blank if not used. Range: **word_level**: text-level timestamp. **phoneme_level**: phoneme-level timestamp.

Table 7-5 Range of property for speakers with high-quality pronunciation

Parameter	Value	Type	Use Case	Sampling Rate (Hz)	Audio Format
Ahmed	arabic_dh_male	Virtual human	Arabic	8k/16k	pcm
Aisha	arabic_dh_female	Virtual human	Arabic	8k/16k	pcm
Ahmed	english_dh_male	Virtual human	English	8k/16k	pcm
Aisha	english_dh_female	Virtual human	English	8k/16k	pcm

Example

```
{
   "command": "START",
   "text": "Nice to meet you.",
   "config":
   {
      "audio_format": "pcm",
      "sample_rate": "16000",
      "property": "english_dh_female"
   }
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

7.3 Real-Time TTS Responses

7.3.1 Responses to Starting TTS

Function

When the speech synthesis engine receives a real-time speech synthesis request, it first sends a synthesis start response to the client, indicating that it has begun processing the speech synthesis request.

Response Parameters

Table 7-6 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value is **START**, indicating the start of speech synthesis.
trace_id	String	Internal token of the service, which can be used to trace a specific process in logs.

Example

```
{
    "resp_type": "START",
    "trace_id": "567e8537-a89c-13c3-a882-826321939651"
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

7.3.2 Responses to TTS Results

7.3.2.1 Audio Stream Data

Function

This API is used to return binary voice data streams in multiple segments. If the user does not specify a voice format, it defaults to returning voice in PCM format.

7.3.2.2 Timestamp Data

Function

While generating audio streams, real-time TTS can also generate timestamp information for each Chinese character/English word. This information can be used for video subtitles and driving virtual human lip-sync.

Request Parameters

Set **subtitle** to **word_level** or **phoneme_level** to enable the timestamp function.

Response Parameters

Table 7-7 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value is **RESULT**.
trace_id	String	Internal token of the service, which can be used to trace a specific process in logs.
result	List	Timestamp information.

Table 7-8 result data structure

Parameter	Type	Description
start_time	Integer	Start timestamp of the synthesized audio corresponding to the text, in milliseconds.
end_time	Integer	End timestamp of the synthesized audio corresponding to the text, in milliseconds.
text	String	Text information.
word_index	Integer	Position of the text in the entire sentence, starting from 0.
phonemes	List	Phoneme timestamp information, which is returned when **subtitle** is set to **phoneme_level**.

Table 7-9 phonemes data structure

Parameter	Type	Description
phoneme	String	Phoneme text information.
start_time	Integer	Start timestamp of the synthesized audio corresponding to the phoneme, in milliseconds.
end_time	Integer	End timestamp of the synthesized audio corresponding to the phoneme, in milliseconds.
phoneme_index	Integer	Phoneme position, starting from 0.

Example

word_level


```
{
 'resp_type': 'RESULT',
 'trace_id':  'd34e3ccb-0383-4c76-a107-ec6ced44614f',
 'result':
       [
          {'start_time': 43980, 'end_time': 44210, 'word_index': 10, 'text': 'there'},
          {'start_time': 44210, 'end_time': 45298, 'word_index': 11, 'text': 'by'}
       ]
}
```

```
{
 'resp_type': 'RESULT',
 'trace_id':  'd34e3ccb-0383-4c76-a107-ec6ced44614f',
 'result':
       [
          {'start_time': 0, 'end_time': 384, 'text': 'Nice', 'word_index': 0},
          {'start_time': 384, 'end_time': 512, 'text': 'to', 'word_index': 1},
          {'start_time': 512, 'end_time': 800, 'text': 'meet', 'word_index': 2},
          {'start_time': 800, 'end_time': 1184, 'text': 'you.', 'word_index': 3},
          {'start_time': 1184, 'end_time': 1284, 'text': '', 'word_index': 4}
       ]
}
```

phoneme_level

```
{
   'resp_type': 'RESULT',
   'trace_id': '39f02607-32d8-4c9f-8b20-11d4af28eecc',
   'result':
   [
      {
         'start_time': 0,
         'end_time': 384,
         'text': 'Nice',
         'word_index': 0,
         'phonemes': [
            {'phoneme_index': 0, 'start_time': 0, 'end_time': 181, 'phoneme': 'n'},
            {'phoneme_index': 1, 'start_time': 181, 'end_time': 288, 'phoneme': 'ay'},
            {'phoneme_index': 2, 'start_time': 288, 'end_time': 384, 'phoneme': 's'}
         ]
      },
      {
         'start_time': 384,
         'end_time': 512,
         'text': 'to',
         'word_index': 1,
         'phonemes': [
            {'phoneme_index': 0, 'start_time': 384, 'end_time': 426, 'phoneme': 't'},
            {'phoneme_index': 1, 'start_time': 426, 'end_time': 512, 'phoneme': 'ah0'}
         ]
      },
      {
         'start_time': 512,
         'end_time': 800,
         'text': 'meet',
         'word_index': 2,
         'phonemes': [
            {'phoneme_index': 0, 'start_time': 512, 'end_time': 608, 'phoneme': 'm'},
            {'phoneme_index': 1, 'start_time': 608, 'end_time': 693, 'phoneme': 'iy'},
            {'phoneme_index': 2, 'start_time': 693, 'end_time': 800, 'phoneme': 't'}
         ]
      },
      {
         'start_time': 800,
         'end_time': 1184,
         'text': 'you.',
         'word_index': 3,
         'phonemes': [
            {'phoneme_index': 0, 'start_time': 800, 'end_time': 864, 'phoneme': 'y'},
```



```
            {'phoneme_index': 1, 'start_time': 864, 'end_time': 1013, 'phoneme': 'uw'},
            {'phoneme_index': 2, 'start_time': 1013, 'end_time': 1184, 'phoneme': ''}
        ]
    },
    {
        'start_time': 1184,
        'end_time': 1284,
        'text': '',
        'word_index': 4,
        'phonemes': [
            {'phoneme_index': 0, 'start_time': 1184, 'end_time': 1284, 'phoneme': ''}
        ]
    }
  ]
}
```

7.3.3 Responses to Ending TTS

Function

After processing the synthesis request, the synthesis engine sends a synthesis end response. Upon receiving this response, the client closes the current WebSocket connection.

Response Parameters

Table 7-10 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value is **END**, indicating the end of voice synthesis.
trace_id	String	Internal token of the service, which can be used to trace a specific process in logs.
reason	String	End cause.

Table 7-11 End causes

Parameter	Description
NORMAL	The response ends properly.
ERROR	An error occurred during synthesis.

Example

```
{
   "resp_type": "END",
   "trace_id": "567e8537-a89c-13c3-a882-826321939651",
   "reason": "NORMAL"
}
```


Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

7.3.4 Responses to TTS Errors

Function

This response is returned if an error occurs when the synthesis engine processes a synthesis request.

Response Parameters

Table 7-12 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value is **ERROR**, indicating an error response.
trace_id	String	Internal token of the service, which can be used to trace a specific process in logs.
error_code	String	Refer to the error code list.
error_msg	String	Returned error message.

Example

```
{
    "resp_type": "ERROR",
    "trace_id": "567e8537-a89c-13c3-a882-826321939651",
    "error_code":"SIS.0032",
    "error_msg":"'audio_format' is invalid"
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

7.3.5 Responses to Fatal Errors

Function

Fatal errors interrupt the whole recognition process. When a fatal error occurs, the process stops, and the server actively disconnects from the client.

Response Parameters

Table 7-13 Response parameters

Parameter	Type	Description
resp_type	String	Response type. The value is **FATAL_ERROR**, indicating that an unrecoverable error occurred during the synthesis.
trace_id	String	Internal token of the service, which can be used to trace a specific process in logs.
error_code	String	Error codes. For details, see "Error Codes".
error_msg	String	Returned error message.

Example

```
{
   "resp_type": "FATAL_ERROR",
   "trace_id": "567e8537-a89c-13c3-a882-826321939651",
   "error_code": "SIS.0304",
   "error_msg": "wait voice timeout"
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

8 Hot Word Management APIs

8.1 Creating a Hot Word Table

In Short Sentence Recognition and RASR services, if there are unique terms in your business domain with suboptimal default recognition performance, consider using hot word management to add these terms to the vocabulary table to enhance recognition accuracy.

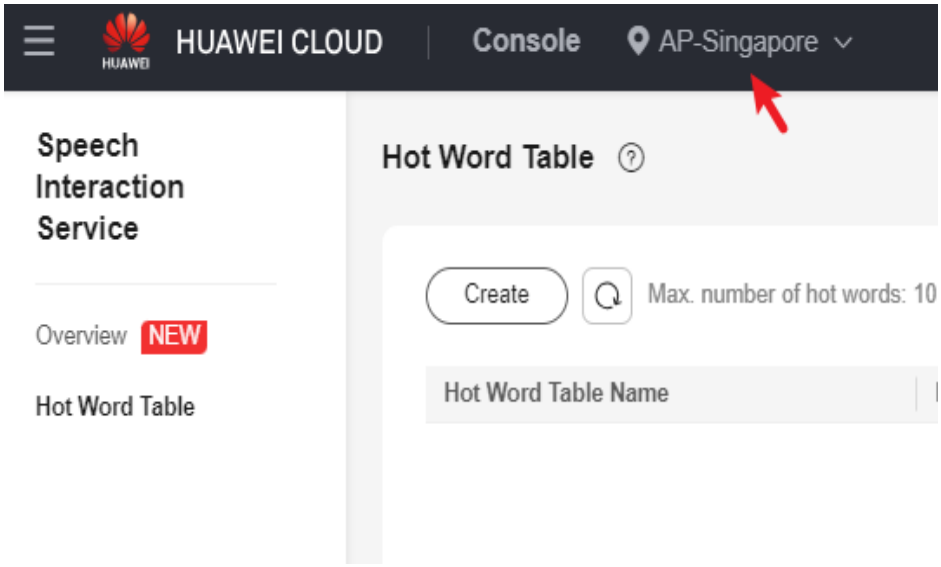
Hot words can include names, company names, or domain-specific proper nouns such as **HUAWEI CLOUD**. It is advisable that hot words should not be excessively lengthy nor contain punctuation marks or special symbols. For constraints, see **Creating a Hot Word Table**.

Function

This API is used to create a hot word table. Upon successful creation, an ID will be returned. Each user is limited to creating up to 100 hot word tables.

When calling hot words, ensure that the region from which they are called matches the region where they were created. As illustrated below, you can check the region associated with the creation of hot words:

Figure 8-1 Check the region where hot words are created



Notes and Constraints

- All hot words that contain English letters must be capitalized, for example: Eiffel Tower (correct example: EIFFEL TOWER).
- All numbers should be represented by their corresponding spoken words, avoiding the use of Arabic numerals, such as Huawei Mate 50 (correct example: HUAWEI MATE FIFTY).
- Hot word content must consist only of English and Chinese characters. Do not use punctuation marks, spaces, or the following special characters: ,?.*
- Do not use monosyllabic English words, for example, MAY and TEE, to prevent incorrect recalls.
- When using hot word tables, the same project ID can share a hot word table, while different project IDs cannot share a hot word table.

URI

POST /v1/{project_id}/asr/vocabularies

Table 8-1 Path parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain a project ID, see **Obtaining a Project ID**.

Request Parameters

Table 8-2 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

Table 8-3 Request body parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Hot word table name, which must be unique. The name can contain a maximum of 32 characters. Only letters, digits, underscores (_), hyphens (-), and pounds (#) are allowed.
description	No	String	Description of the hot word table. The value can contain a maximum of 255 characters.
language	Yes	String	Language type of the hot word table. Value: **chinese_mandarin**, indicating Mandarin Chinese.
contents	Yes	Array of strings	Chinese-English hybrid hot words are supported. A hot word can contain only English letters and Chinese characters encoded in Unicode mode. Other symbols, including spaces, are not allowed. Arabic numerals are written in English (such as one). A single hot word table supports up to 10,000 hot words. A single hot word can contain a maximum of 32 characters in Chinese and 64 characters in English.

Response Parameters

Status code: 200

Table 8-4 Response body parameters

Parameter	Mandatory	Type	Description
vocabulary_id	Yes	String	If the API is successfully called, the hot word table ID is returned. If it fails, this field is not included.

Status code: 400

Table 8-5 Response body parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not returned for a successful call.
error_msg	String	Error message when the API call fails. This parameter is not returned for a successful call.

Example Requests

NOTE

The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

Create a hot word table.

```
POST https://{endpoint}/v1/{project_id}/asr/vocabularies

Request header:
Content-Type: application/json
X-Auth-Token: MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...

Request body:
{
  "name": "telepower",
  "description": "telepower descriptiondescription",
  "language": "chinese_mandarin",
  "contents": ["example"]
}
```

Example Responses

Status code: 200

Example response for a successful request

```
{
  "vocabulary_id": "CFD08A32-6176-4ad7-92F9-11ED015C8109",
}
```


Status code: 400

Example response for a failed request

```
{
   "error_code":"SIS.0201",
   "error_msg":"***"
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

8.2 Updating a Hot Word Table

Function

This API is used to update a hot word table. Upon successful update, an ID will be returned.

URI

PUT /v1/{project_id}/asr/vocabularies/{vocabulary_id}

Table 8-6 Path parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain a project ID, see **Obtaining a Project ID**.
vocabulary_id	Yes	String	ID of the hot word table to be updated.

Request Parameters

Table 8-7 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

Table 8-8 Request body parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Hot word table name, which must be unique. The name can contain a maximum of 32 characters. Only letters, digits, underscores (_), hyphens (-), and pounds (#) are allowed.
description	No	String	Description of the hot word table. The value can contain a maximum of 255 characters.
language	Yes	String	Language type of the hot word table. Value: **chinese_mandarin**, indicating Mandarin Chinese.
contents	Yes	Array of strings	Chinese-English hybrid hot words are supported. A hot word can contain only English letters and Chinese characters encoded in Unicode mode. Other symbols, including spaces, are not allowed. A single hot word table supports up to 10,000 hot words. A hot word can contain 32 bytes at most.

Response Parameters

Status code: 200

Table 8-9 Response body parameters

Parameter	Mandatory	Type	Description
vocabulary_id	Yes	String	If the calling is successful, ID of the hot word table is returned. Otherwise, this parameter is invalid.

Status code: 400

Table 8-10 Response body parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not returned for a successful call.
error_msg	String	Error message when the API call fails. This parameter is not returned for a successful call.

Example Requests

NOTE

The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

- Update a hot word table.

```
PUT https://{endpoint}/v1/{project_id}/asr/vocabularies/{vocabulary_id}

Request header:
Content-Type: application/json
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...

Request body:
{
  "name": "telepower",
  "description": "telepower "description,
  "language": "chinese_mandarin",
  "contents": ["example"]
}
```

Example Responses

Status code: 200

Example response for a successful request

```
{
  "vocabulary_id": "CFD08A32-6176-4ad7-92F9-11ED015C8109",
}
```

Status code: 400

Example response for a failed request


```
{
    "error_code":"SIS.0201",
    "error_msg":"***"
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

8.3 Querying Hot Word Table Information

Function

This API is used to query the information and content of a hot word table based on its ID.

URI

GET /v1/{project_id}/asr/vocabularies/{vocabulary_id}

Table 8-11 Path parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain a project ID, see **Obtaining a Project ID**.
vocabulary_id	Yes	String	ID of the hot word table to be queried.

Request Parameters

Table 8-12 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

Response Parameters

Status code: 200

Table 8-13 Response body parameters

Parameter	Mandatory	Type	Description
vocabulary_id	Yes	String	If the calling is successful, ID of the hot word table is returned. Otherwise, this parameter is invalid.
name	Yes	String	If the calling is successful, name of the hot word table is returned. Otherwise, this parameter is invalid.
language	Yes	String	If the calling is successful, language type of the hot word table is returned. Otherwise, this parameter is invalid.
description	Yes	String	If the calling is successful, description of the hot word table is returned. Otherwise, this parameter is invalid.
contents	Yes	Array of strings	If the calling is successful, the hot word table is returned. Otherwise, this parameter is invalid.

Status code: 400

Table 8-14 Response body parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not returned for a successful call.
error_msg	String	Error message when the API call fails. This parameter is not returned for a successful call.

Example Requests

NOTE

The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

- Query hot word table information.

```
GET https://{endpoint}/v1/{project_id}/asr/vocabularies/{vocabulary_id}

Request header:
Content-Type: application/json
```



```
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
```

Example Responses

Status code: 200

Example response for a successful request

```
{
  "vocabulary_id": "CFD08A32-6176-4ad7-92F9-11ED015C8109",
  "name": "telepower",
  "description": "telepower description",
  "language": "chinese_mandarin",
  "contents": ["example"]
}
```

Status code: 400

Example response for a failed request

```
{
    "error_code":"SIS.0201",
    "error_msg":"***"
}
```

Status Codes

See **Status Codes**.

Error Codes

See **Error Codes**.

8.4 Deleting a Hot Word Table

Function

This API is used to delete a hot word table based on its ID.

URI

DELETE /v1/{project_id}/asr/vocabularies/{vocabulary_id}

Table 8-15 Path parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.
vocabulary_id	Yes	String	ID of the hot word table to be deleted.

Request Parameters

Table 8-16 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

Response Parameters

No response message will be returned. If HTTP status code **200** is returned, the table is deleted successfully.

Status code: 400

Table 8-17 Response body parameters

Parameter	Type	Description
error_code	String	Error code returned when the API fails to be called. This parameter is not included when the API is successfully called.
error_msg	String	Error message returned when the API fails to be called. This parameter is not included when the API is successfully called.

Example Requests

NOTE

The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

- Delete a hot word table.

```
DELETE https://{endpoint}/v1/{project_id}/asr/vocabularies/{vocabulary_id}

Request header:
Content-Type: application/json
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
```

Example Responses

Status code: 400

Failure response example

```
{
  "error_code" : "SIS.0201",
  "error_msg" : "***"
}
```

Status Code

For details about status codes, see **Status Codes**.

Error Code

For details about error codes, see **Error Codes**.

8.5 Querying Hot Word Tables

Function

This API is used to query all hot word tables of a user.

URI

GET /v1/{project_id}/asr/vocabularies

Table 8-18 Path parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request Parameters

Table 8-19 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It is used to obtain the permission to call APIs. For details about how to obtain a user token, see **Authentication**. The token is the value of **X-Subject-Token** in the response header.

Table 8-20 Request body parameters

Parameter	Mandatory	Type	Description
name	No	String	Name of a hot word table, used for filtering hot word tables.

Response Parameters

Status code: 200

Table 8-21 Response body parameters

Parameter	Mandatory	Type	Description
count	Yes	Integer	Number of hot word tables.
result	Yes	Array of **VocabInfo** objects	If the calling is successful, hot word tables are returned. Otherwise, this parameter is invalid.

Table 8-22 VocabInfo

Parameter	Mandatory	Type	Description
vocabulary_id	Yes	String	Hot word ID
name	Yes	String	Name of a hot word table
language	Yes	String	Language type of the hot word table
description	Yes	String	Description of a hot word table

Status code: 400

Table 8-23 Response body parameters

Parameter	Type	Description
error_code	String	Error code returned when the API fails to be called. This parameter is not included when the API is successfully called.

Parameter	Type	Description
error_msg	String	Error message returned when the API fails to be called. This parameter is not included when the API is successfully called.

Example Requests

NOTE

The endpoint is the request URL for calling an API. Endpoints vary according to services and regions. For details, see **Endpoints**.

- Obtain the hot word table list.

```
GET https://{endpoint}/v1/{project_id}/asr/vocabularies

Request header:
Content-Type: application/json
X-Auth-Token:
MIINRwYJKoZIhvcNAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
```

Example Responses

Status code: 200

Example response (successful request)

```
{
  "result": [
    {
      "vocabulary_id": "5F85A74C-BED9-4a15-B66E-039251D877D6",
      "language": "chinese_mandarin",
      "name": "weather",
      "description": "no desc"
    },
    {
      "vocabulary_id": "50875954-7328-42ab-B236-B3EC6E22207A",
      "language": "chinese_mandarin",
      "name": "war",
      "description": "no desc"
    }
  ]
}
```

Status code: 400

Failure response example

```
{
    "error_code":"SIS.0201",
    "error_msg":"***"
}
```

Status Code

For details about status codes, see **Status Codes**.

Error Code

For details about error codes, see **Error Codes**.

9 Appendix

9.1 Audio Examples

Table 9-1 lists the test audio recordings. The title of each audio recording file contains the sampling rate and bit width. For example, **8k16bit.pcm** indicates that the audio sampling rate is 8 kHz and the bit width is 16 bits.

Table 9-1 Audio examples

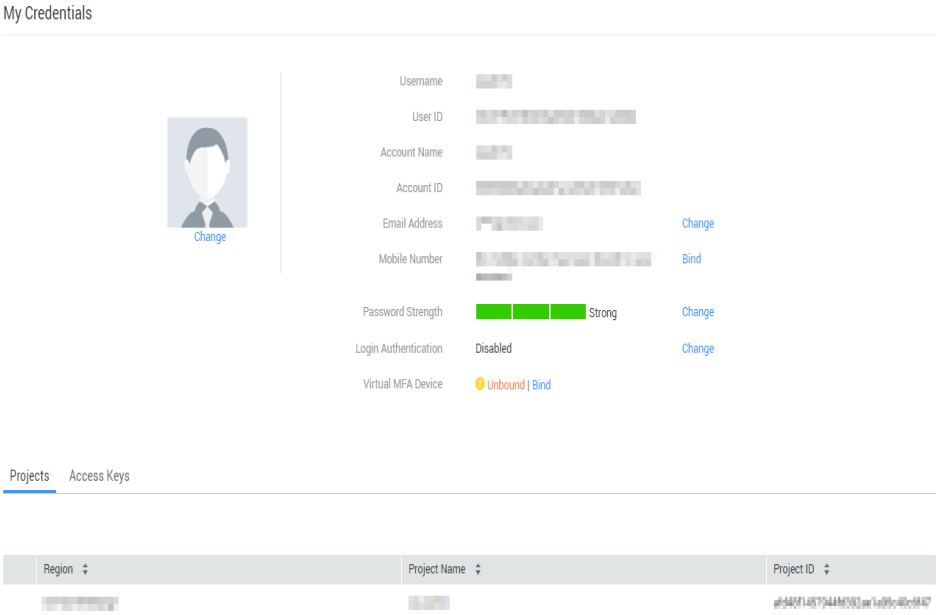
Audio Format	Download Link
8k pcm	https://sis-open-data.obs.ap-southeast-3.myhuaweicloud.com/audio/8k.pcm
8k wav	https://sis-open-data.obs.ap-southeast-3.myhuaweicloud.com/audio/8k.wav
16k pcm	https://sis-open-data.obs.ap-southeast-3.myhuaweicloud.com/audio/16k.pcm
16k wav	https://sis-open-data.obs.ap-southeast-3.myhuaweicloud.com/audio/16k.wav

9.2 Obtaining a Project ID

Obtaining a Project ID from the Console

1. Log in to the **management console**.
2. Move the cursor over your username in the upper right corner and click **My Credentials** from the drop-down list.
3. On the **My Credentials** page, view the username and account name and view projects in the project list.

Figure 9-1 Viewing the project ID



If there are multiple projects, unfold the target region and obtain the project ID from the **Project ID** column.

Obtaining a Project ID by Calling an API

The API for obtaining a project ID is **GET https://{Endpoint}/v3/projects**. **{Endpoint}** indicates the endpoint of IAM. For details about API authentication, see **Authentication**.

The following is an example response. If SIS is deployed in the **ap-southeast-1** region, the value of **name** in the request body is **southeast-1**, and the value of **id** in **projects** is the project ID.

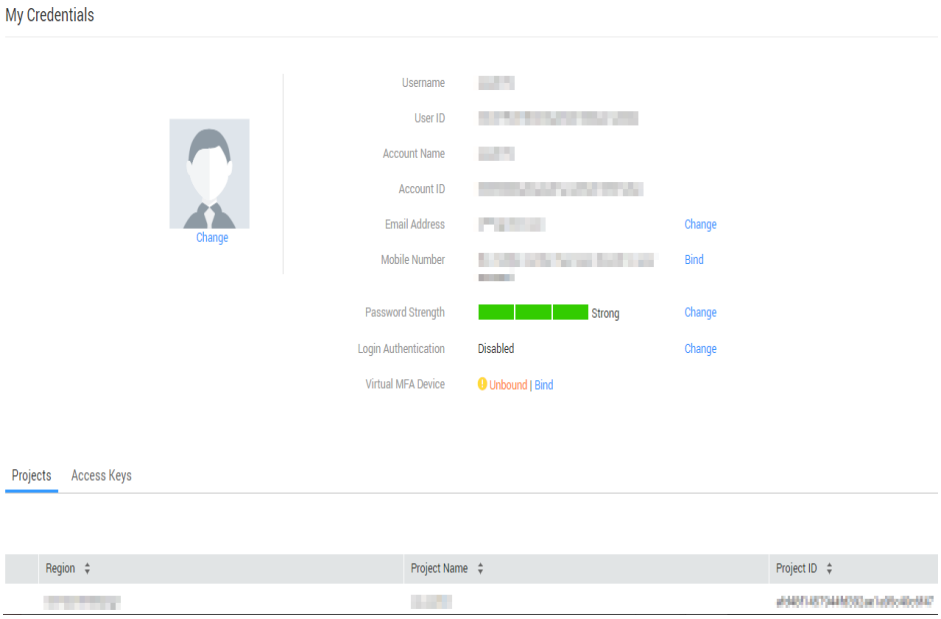
```
{
    "projects": [
        {
            "domain_id": "65382450e8f64ac0870cd180d14e684b",
            "is_domain": false,
            "parent_id": "65382450e8f64ac0870cd180d14e684b",
            "name": "project_name",
            "description": "",
            "links": {
                "next": null,
                "previous": null,
                "self": "https://www.example.com/v3/projects/a4a5d4098fb4474fa22"
            },
            "id": "a4a5d4098fb4474fa22cd05f897d6b99",
            "enabled": true
        }
    ],
    "links": {
        "next": null,
        "previous": null,
        "self": "https://www.example.com/v3/projects"
    }
}
```


9.3 Obtaining an Account ID

An account ID (**domain-id**) is required for some URLs when an API is called. To obtain an account ID, perform the following steps:

1. Log in to the management console after registration.
2. Move the cursor over your username in the upper right corner and click **My Credentials** from the drop-down list.
3. On the **My Credentials** page, view the **Account ID**.

Figure 9-2 Viewing the account ID



9.4 Obtaining an AK/SK Pair

If an AK/SK has already been generated, skip this step. Find the downloaded AK/SK file, which is usually named **credentials.csv**.

As shown in the following figure, the file contains the username, AK, and SK

Figure 9-3 Content of the **credential.csv** file

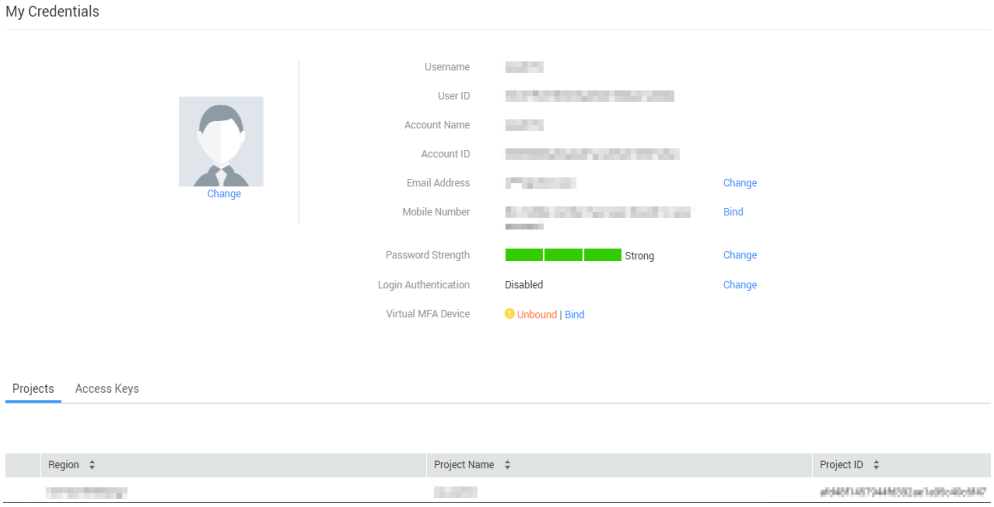
	A	B	C
1	User Name	Access Key Id	Secret Access Key
2	hu[redacted]dg	QTWA[redacted]UT2QVKYUC	MFyfvK41ba2[redacted]npdUKGpownRZlmVmHc

Perform the following operations to generate an AK/SK pair:

1. Log in to the console.
2. Click the username and choose **My Credentials** from the drop-down list.
3. On the **My Credentials** page, click the **Access Keys** tab.

4. On the page that is displayed, click **Add Access Key**.
5. Obtain and download the key as prompted. Keep the key secure.

Figure 9-4 Obtaining an AK/SK



9.5 Common Request Parameters

Table 9-2 describes the request parameters.

Table 9-2 Common request parameters

Parameter	Mandatory	Description	Example
Content-type	Yes	MIME type of the response body	application/json
Content-Length	This parameter is mandatory for POST and PUT requests, but must be left blank for GET requests.	Length of the request body. The unit is byte.	3495
X-Auth-Token	Yes	User token	MIINRwYJKoZIhvc-NAQcCoIINODCCDTQCAQExDTALBglghkgBZQMEAgEwgguVBgkqhkiG...
X-Language	No	Request language type. The default value is **zh-cn**.	en-us

NOTE

- For details about other parameters in the message header, see the HTTPS protocol documentation.
- When calling a service API, add the message body of **Content-Type** to the request message header.

9.6 Common Response Parameters

Table 9-3 describes the request parameters.

Table 9-3 Common response parameters

Parameter	Description
Content-Length	Length of the response message body. The unit is byte.
Date	Time when a response is returned
Content-type	MIME type of the request body. The value is **application/json**.

9.7 Status Codes

Table 9-4 Status codes

Status Code	Description
100	Continue
101	Switching Protocols
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
204	NO Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Found

Status Code	Description
303	See Other
304	Not Modified
305	Use Proxy
306	Unused
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
422	Unprocessable Entity
429	Too Many Requests
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported

9.8 Error Codes

If an error occurs during API calling, no result is returned. You can locate the cause of an error using the error codes of each API. When an API call fails, HTTPS status code 4*xx* or 5*xx* is returned. The returned message body contains a specific error code and error message. If you fail to locate the cause of the error, contact the Huawei Cloudcustomer service and provide the error code for troubleshooting.

Format of an Error Response Body

If an error occurs during API calling, an error code and a message will be displayed. The following shows an error response body.

```
{

   "error_code": "SIS.0032",
   "error_msg": "'audio_format' is invalid"
}
```

In the response body, **error_code** is an error code, and **error_msg** provides information about the error.

Error Code Description

If an error code starting with APIGW is returned after you call an API, rectify the fault by referring to the instructions provided in **API Gateway Error Codes**.

Error Code	Description	Solution
SIS.0001	Connection error. For example, the connection to OBS or Redis is incorrect.	Contact technical support engineers.
SIS.0003	The token does not contain user information, the agency has been created, or an internal error occurs.	Contact technical support engineers.
SIS.0100	Internal service error.	Contact technical support engineers.

Error Code	Description	Solution
SIS.0101	The authentication token is abnormal.	• Check whether the value of X-Auth-Token in the request header is correct. • Check whether the token request parameters are correct. • Check whether the value of **projectId** in the URL is correct. • When obtaining a token, add the scope parameter to the request to obtain a project-level token. Do not use a global token when calling SIS. 
SIS.0102	Authentication information is missing.	Check whether the **X-Auth-Token** field in the request header exists.
SIS.0103	Real-name authentication is missing.	Perform real-name authentication. No authentication is required for Huawei Cloud International Website.
SIS.0012	Fields required in the request body are missing.	Check whether mandatory fields of request parameters are missing.
SIS.0031	The request parameter is not supported.	Check whether the request parameters are correct.

Error Code	Description	Solution
SIS.0032	The JSON format of the request body is incorrect.	Check whether the JSON format of the request body is correct. ● Ensure that the value of **audio_format** is valid and is same as the format of the audio to be identified. If you are not sure about the value of **audio_format** when calling the Long Audio Transcription API, set the value to auto for debugging. ● If the error message **data base64 encode invalid** is displayed, check whether the character string is correct after Base64 encoding, for example, whether there are extra spaces or incorrect characters. ● If the error message "xx cannot be empty" is displayed, the field must not be left blank and requires assignment. For example, if the error "language cannot be empty" occurs when creating a hot word, the **language** field in the request body cannot be empty. ● If the error message "callback_url is invalid or too long" is displayed, check whether the URL format is correct.
SIS.0022	The product cannot be subscribed to.	The product cannot be purchased. Contact technical support.
SIS.0023	Failed to subscribe to the product.	Failed to purchase the product. Contact technical support.
SIS.0024	Updating restrictions is not allowed.	The restrictions cannot be updated. Contact technical support.
SIS.0033	The engine response timed out.	Contact technical support engineers.
SIS.0201	Failed to find the hot word table.	Check whether the request parameters are correct or contact technical support.

Error Code	Description	Solution
SIS.0203	Some hot words are too long or invalid.	Correct the invalid hot words based on the error information.
SIS.0204	The hot word table name already exists.	Modify the hot word table name.
SIS.0205	The language is not supported.	Modify the hot word table.
SIS.0206	Failed to save the hot word.	Contact technical support engineers.
SIS.0207	The hot word content is damaged.	Contact technical support engineers.
SIS.0208	There are too many hot word tables.	Delete unnecessary hot word tables. If you need to configure more hot word tables, contact customer service.
SIS.0301	The input **audio_format** parameter does not match the model.	Check whether the request parameters are correct.
SIS.0302	The internal service is abnormal.	Contact technical support engineers.
SIS.0303	The engine failed to be connected to.	Try again. If the problem persists, contact technical support.
SIS.0304	Audio waiting timed out.	This error is reported if the client does not send a voice message for a long time and the server does not receive the voice message within 20 seconds. • Reduce the data sending interval. • Check the code to view whether there is too much idle time after an audio segment is sent. • Check whether an ending request is sent and the client is closed.
SIS.0305	An exception occurred during speech recognition.	Try again or contact technical support engineers.
SIS.0306	An exception occurred during speech recognition.	Try again or contact technical support engineers.
SIS.0307	An exception occurred during speech recognition.	Try again or contact technical support engineers.

Error Code	Description	Solution
SIS.0309	The maximum duration of Real-Time ASR is reached.	Note the duration limits for audio recognition: the maximum length for both streaming and single-sentence modes is 60 seconds, while for continuous mode, it extends up to 5 hours.
SIS.0312	The maximum number of concurrent resources is exceeded.	If the application sends requests at an excessively high rate, reduce the request frequency or contact technical support engineers.
SIS.0401	An exception occurred during audio synthesis.	Try again or contact technical support engineers.
SIS.0402	The input **sample_rate** parameter for TTS is invalid.	Check whether the input **sample_rate** for TTS matches the sampling ratio supported by the value range of **property**.
SIS.0410	The input **audio_format** parameter for TTS is invalid.	Check whether the request parameters are correct.
SIS.0411	The input **property** parameter for TTS is invalid.	Check whether the request parameters are correct.
SIS.0412	Connection to the TTS engine timed out.	Try again or contact technical support engineers.
SIS.0413	Internal TTS error occurred.	Try again or contact technical support engineers.
SIS.0414	TTS waiting timed out.	Try again or contact technical support engineers.
SIS.0415	The TTS request body is incorrect.	Check whether the request body is correct.
SIS.0416	Waiting for the user to send the text timed out.	Resend the text you want to convert to speech.
SIS.0417	TTS engine error.	Try again or contact technical support engineers.
SIS.0418	TA TTS task is being processed.	Do not repeatedly send the command for starting a TTS task.
SIS.0419	Invalid SSML.	The input text contains invalid characters.
SIS.0504	The audio to be recognized is invalid for Express Audio File Transcription.	Check whether the audio file size meets the requirements.

Error Code	Description	Solution
SIS.0506	Failed to find a proxy.	• Check whether the network has a proxy. • Check whether OBS access is authorized. 
SIS.0507	The input parameters are invalid for Express Audio File Transcription.	Check whether the **obs_bucket_name** and **obs_object_key** parameters are empty.
SIS.0511	The audio to be recognized is invalid for Express Audio File Transcription.	Check whether the audio duration meets the requirements.
SIS.0512	The input parameters are invalid for Express Audio File Transcription.	Check whether the bucket name and file name specified by **obs_bucket_name** and **obs_object_key** exist.
SIS.0513	The input parameters are invalid for Express Audio File Transcription.	Check whether the **obs_object_key** parameter is invalid.
SIS.0533	The request parameters of Long Audio Transcription are incorrect.	Check whether the request parameters are correct.
SIS.0534	The request body of Long Audio Transcription is incorrect.	Check whether the request body is correct. • If **data syntax error!** is displayed, check whether the data format or encoding format meets the requirements. • If **asr data url param is invalid** is displayed, use the path of the audio file in an OBS bucket and ensure that the region of the OBS service is the same as that of the requested service.

Error Code	Description	Solution
SIS.0535	The requested file type of Long Audio Transcription is not supported.	• Ensure that the value of **audio_format** is valid. • Ensure that the format of the audio to be identified is the same as the value of **audio_format**.
SIS.0536	The number of submitted jobs of Long Audio Transcription reached the upper limit.	Try again later.
SIS.0537	The size of an audio recording is too large.	Reduce the file size. For example, divide it into multiple files.
SIS.0538	The Long Audio Transcription job timed out.	Try again later.
SIS.0540	Size of a recording file.	Increase the size of the recording file.
SIS.0541	The **property** and **format** parameters of Audio File Transcription do not match.	Check whether the formats of **property** and **format** are correct.
SIS.0601	The input parameter of ASR or TTS is invalid.	Check whether the request parameters are correct and complete.
SIS.0602	The speech format for ASR is not supported.	Check whether the encoding format of the submitted speech is supported.
SIS.0604	The size of the ASR file does not meet the requirements or the number of words to be synthesized in TTS exceeds the upper limit.	Reduce the size of the speech file or the number of words to be synthesized. Note that when the OBS link is used for submission, the file size increases after Base64 encoding.
SIS.0605	An ASR or TTS internal error occurred.	Try again or contact technical support engineers.
SIS.0608	Invalid URL.	Check the URL and ensure that the OBS link of the corresponding region is used.
SIS.0609	Failed to download the audio file from the specified URL.	Check whether the OBS is in the public-read status or whether OBS authentication is enabled.

Error Code	Description	Solution
SIS.0701	The size of the input audio, video, or exam paper exceeds the upper limit.	Refer to the error information and API document to reduce the size of the audio, video, or exam paper.
SIS.0702	The input parameter is invalid. For example, the assessment language or mode is not supported. The audio or video format is not supported. The exam paper is invalid.	Refer to the error information and API document to enter correct parameters.
SIS.30003	Failed to download the audio from OBS.	Check whether the audio on OBS can be accessed.
SIS.30004	Failed to download the hot word file.	Check whether the hot word file exists.
SIS.30006	Failed to decode the audio.	Check whether the format of the audio file is correct and whether the audio file is empty.
SIS.30007	The transcription engine failed to load hot words.	Check whether the hot word file exists.
SIS.30008	The transcription engine failed to transcript the audio.	Check whether the parameters are correct or contact technical support.

