

Alibaba Cloud

Intelligent Speech Interaction
Real-time Speech Recognition

Document Version: 20220414

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).
5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
 Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: If the weight is set to 0, the server no longer receives new requests.
 Note	A note indicates supplemental instructions, best practices, tips, and other content.	 Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings> Network> Set network type .
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click OK .
<code>Courier font</code>	Courier font is used for commands	Run the <code>cd /d C:/window</code> command to enter the Windows system folder.
<i>Italic</i>	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>
{ } or {a b}	This format is used for a required value, where only one item can be selected.	<code>switch {active stand}</code>

Table of Contents

1.NUI SDK for mobile clients	05
1.1. NUI SDK for Android	05
1.2. Error codes	10
2.SDK for C++	21
3.WebSocket	37

1.NUI SDK for mobile clients

1.1. NUI SDK for Android

The real-time speech recognition service provides a Natural User Interaction (NUI) SDK for Android. This topic describes how to download the NUI SDK for Android, lists the key methods in the SDK, and provides sample code for you to use the SDK.

Prerequisites

- You understand how the SDK works. For more information, see [Overview](#).
- The appkey of your project is obtained. For more information, see [create a project](#).
- A token used to access the service is obtained. For more information, see [Obtain a token](#).

Download and install the SDK

1. [Download the NUI SDK for Android and sample code](#).
2. Decompress the downloaded package to obtain the demo project and find the SDK package in the app/libs directory, which is an AAR package.
3. Open the demo project in Android Studio.

The sample code for the real-time speech recognition service is stored in the `SpeechTranscriberActivity.java` file.

Key methods

- **initialize**: initializes the SDK.

```
/**
 * Initialize the SDK. The SDK uses a singleton pattern. To initialize the SDK again,
 * you must first release the SDK. Do not call the SDK on the user interface (UI) thread. Ot
 * herwise, the process may be blocked.
 * @param callback: the event listener callback. For more information, see the follow
 * ing callback methods.
 * @param parameters: the parameters used in the initialization. For more information
 * , see Overview.
 * @param level: the log level to use. The smaller the parameter value is, the more l
 * ogs are recorded.
 * @param save_log: specifies whether to store logs in files. The debug_path paramete
 * r specifies the directory where log files are stored.
 * @return: the returned error code. For more information, see Error codes.
 */
public synchronized int initialize(final INativeNuiCallback callback,
                                   String parameters,
                                   final Constants.LogLevel level,
                                   final boolean save_log)
```

INativeNuiCallback supports the following callback methods.

- **onNuiAudioStateChanged**: determines whether to enable recording based on the value of **AudioState**.

```
/**
 * When the start, stop, or cancel method is called, the SDK uses this callback method to instruct the client to enable or disable recording.
 * @param state: specifies whether to enable recording.
 */
void onNuiAudioStateChanged(AudioState state);
```

- **onNuiNeedAudioData**: provides audio data.

```
/**
 * When the server starts a recognition task, this method is continuously called to read audio data from the client.
 * @param buffer: the storage space of the server for storing audio data.
 * @param len: the required number of bytes of the audio data to be read from the client.
 * @return: the actual number of bytes of the audio data that is read from the client.
 */
int onNuiNeedAudioData(byte[] buffer, int len);
```

- **onNuiEventCallback**: reports the occurred event to the server.

```
/**
 * Report the occurred event to the server.
 * @param event: the event to be reported by the client. You can view possible events in the following table.
 * @param resultCode: the returned error code. This parameter is valid for the EVENT_ASR_ERROR event.
 * @param arg2: Reserved.
 * @param kwsResult: the wake-up word recognition feature.
 * @param asrResult: the recognition result of the audio stream.
 */
void onNuiEventCallback(NuiEvent event, final int resultCode, final int arg2, KwsResult kwsResult, AsrResult asrResult);
```

The following table lists the possible events in the SDK.

Name	Description
EVENT_VAD_START	Detects the beginning of a speech.
EVENT_VAD_END	Detects the end of a speech.
EVENT_ASR_PARTIAL_RESULT	Generates the intermediate recognition result.
EVENT_ASR_RESULT	Generates the final recognition result.

Name	Description
EVENT_ASR_ERROR	Determines the error cause based on the returned error code.
EVENT_MIC_ERROR	Returns a recording error.
EVENT_SENTENCE_START	Detects the beginning of a sentence. This event is valid for the real-time speech recognition service.
EVENT_SENTENCE_END	Detects the end of a sentence. This event is valid for the real-time speech recognition service.
EVENT_SENTENCE_SEMANTICS	Reserved.
EVENT_TRANSCRIBER_COMPLETE	Indicates that the recognition task is completed.

- **set_params**: sets SDK parameters in the JSON format.

```
/**
 * Set parameters in the JSON format.
 * @param params: the request parameters. For more information, see Overview.
 * @return: the returned error code. For more information, see Error codes.
 */
public synchronized int setParams(String params)
```

- **startDialog**: starts the recognition task.

```
/**
 * Start the recognition task.
 * @param vad_mode: the voice activity detection (VAD) mode of the task. Use the Production-to-Test (P2T) mode for a recognition task.
 * @return: the returned error code. For more information, see Error codes.
 */
public synchronized int startDialog(VadMode vad_mode, String dialog_params)
```

- **stopDialog**: completes the recognition task.

```
/**
 * When this method is called, the server returns the final recognition result to the client and completes the recognition task.
 * @return: the returned error code. For more information, see Error codes.
 */
public synchronized int stopDialog()
```

- **release**: releases the SDK.

```
/**
 * Release the SDK.
 * @return: the returned error code. For more information, see Error codes.
 */
public synchronized int release()
```

Procedure

1. Initialize the SDK and the recorder instance.
2. Set request parameters based on your business requirements.
3. Call the startDialog method to start the recognition task.
4. Call the onNuiAudioStateChanged method based on the value of AudioState and then enable recording accordingly.
5. Call the onNuiNeedAudioData method to send audio data to the server.
6. Obtain the recognition result in the EVENT_ASR_PARTIAL_RESULT and EVENT_SENTENCE_END callback events.
7. Call the stopDialog method to complete the recognition task.
8. Call the release method to release the SDK.

ProGuard configuration

If you use the obfuscating code, configure the following command in the proguard-rules.pro file:

```
-keep class com.alibaba.idst.nui. *{*};
```

Sample code

Initialize the NUI SDK

```
CommonUtils.copyAssetsData(this);
int ret = NativeNui.GetInstance().initialize(this, genInitParams(path,path2), Constants.Log
Level.LOG_LEVEL_VERBOSE, true);
```

The genInitParams method generates a JSON string that contains the information about the resource directory and user. The user information contains the following parameters:


```
private String genInitParams(String workpath, String debugpath) {
    String str = "";
    try{
        JSONObject object;
        object.put("app_key","");
        object.put("token","");
        object.put("device_id",Utils.getDeviceId());
        object.put("url","wss://nls-gateway.ap-southeast-1.aliyuncs.com:443/ws/v1");
        object.put("workspace", workpath);
        object.put("debug_path",debugpath);
        str = object.toString();
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return str;
}
```

Set the request parameters

Set the request parameters in the format of a JSON string, as shown in the following code:

```
private String genParams() {
    String params = "";
    try {
        JSONObject nls_config = new JSONObject();
        nls_config.put("enable_intermediate_result", true);
        JSONObject parameters = new JSONObject();
        parameters.put("nls_config", nls_config);
        // Select the real-time speech recognition service.
        parameters.put("service_type", Constants.kServiceTypeSpeechTranscriber);
        params = parameters.toString();
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return params;
}

NativeNui.GetInstance().setParams(genParams());
```

Start the recognition task

Call the startDialog method to start the recognition task.

```
NativeNui.GetInstance().startDialog(Constants.VadMode.TYPE_P2T, genDialogParams());
```

Handle callbacks

- Call the onNuiAudioStateChanged method based on the value of AudioState. Then, the SDK determines whether to enable recording based on the obtained value.

```
public void onNuiAudioStateChanged(Constants.AudioState state) {
    Log.i(TAG, "onNuiAudioStateChanged");
    if (state == Constants.AudioState.STATE_OPEN) {
        Log.i(TAG, "audio recorder start");
        mAudioRecorder.startRecording();
    } else if (state == Constants.AudioState.STATE_CLOSE) {
        Log.i(TAG, "audio recorder close");
        mAudioRecorder.release();
    } else if (state == Constants.AudioState.STATE_PAUSE) {
        Log.i(TAG, "audio recorder pause");
        mAudioRecorder.stop();
    }
}
```

- Call the `onNuiNeedAudioData` method to send audio data to the server.

```
public int onNuiNeedAudioData(byte[] buffer, int len) {
    int ret = 0;
    if (mAudioRecorder.getState() != AudioRecord.STATE_INITIALIZED) {
        Log.e(TAG, "audio recorder not init");
        return -1;
    }
    ret = mAudioRecorder.read(buffer, 0, len);
    return ret;
}
```

- Call the `onNuiEventCallback` method to report the occurred event to the server. Do not call an SDK method in the callbacks. Otherwise, a deadlock may occur.

```
public void onNuiEventCallback(Constants.NuiEvent event, final int resultCode, final int
arg2, KwsResult kwsResult, AsrResult asrResult) {
    Log.i(TAG, "event=" + event);
    if (event == Constants.NuiEvent.EVENT_ASR_RESULT) {
        showText(asrView, asrResult.asrResult);
    } else if (event == Constants.NuiEvent.EVENT_ASR_PARTIAL_RESULT ||
        event == Constants.NuiEvent.EVENT_SENTENCE_END) {
        showText(asrView, asrResult.asrResult);
    } else if (event == Constants.NuiEvent.EVENT_ASR_ERROR) {
        ;
    } else if (event == Constants.NuiEvent.EVENT_TRANSCRIBER_COMPLETE) {
        ;
    }
}
```

Complete the recognition task

```
NativeNui.GetInstance().stopDialog();
```

1.2. Error codes

This topic describes the error codes and error messages that may be returned during the use of Intelligent Speech Interaction to facilitate troubleshooting.

Description

An error code may be returned by the SDK or the server.

- Error codes returned by the SDK:

These error codes may be generated during the use of the Intelligent Speech Interaction SDK. The SDK returns an error code in a callback.

- Error codes returned by the server:

If a `DEFAULT-NLS_ERROR` or `HTTP_SERVER_ERROR` event is returned, an error occurred during the service use. The status parameter in the header object of the event displays the error code.

Note

The error codes and error messages described in this topic apply to the short sentence recognition and real-time speech recognition services.

Successful requests

Error code	Error message	Description
0	SUCCESS	The task is successful.

Configuration or Parameter errors

Error code	Error message	Description
240999	DEFAULT_ERROR	The error message returned because a default internal error occurred.
240001	NUI_CONFIG_INVALID	The error message returned because the configuration file is invalid.
240002	ILLEGAL_PARAM	The error message returned because a specified parameter is invalid.
240003	ILLEGAL_INIT_PARAM	The error message returned because an initialization parameter is invalid.
240004	NECESSARY_PARAM_LACK	The error message returned because a required parameter is missing.

Error code	Error message	Description
240005	NULL_PARAM_ERROR	The error message returned because a parameter is left empty.
240006	NULL_LISTENER_ERROR	The error message returned because the listener callback is not specified.
240007	NULL_DIALOG_ERROR	The error message returned because no valid dialog instance is specified. This is generally an internal status error.
240008	NULL_ENGINE_ERROR	The error message returned because no valid engine instance is specified. Check whether the engine initialization is successful.
240009	ILLEGAL_DATA	The error message returned because the URL of the audio data is invalid or the size of the audio data exceeds the upper limit.

Status errors related to the SDK

Error code	Error message	Description
240010	ILLEGAL_REENTRANT	The error message returned because you used the SDK after you exited the client.
240011	SDK_NOT_INIT	The error message returned because the SDK is not properly initialized.
240012	SDK_ALREADY_INIT	The error message returned because the SDK is repeatedly initialized.
240013	DIALOG_INVALID_STATE	The error message returned because the internal dialog instance is in an abnormal state.
240014	STATE_INVALID	The error message returned because the SDK is in an abnormal state.

Error code	Error message	Description
240015	ILLEGAL_FUNC_CALL	The error message returned because the SDK is not used in a valid mode.

System errors

Error codes	Error message	Description
240020	MEM_ALLOC_ERROR	The error message returned because memory resources failed to be allocated.
240021	FILE_ACCESS_FAIL	The error message returned because the file failed to be accessed.
240022	CREATE_DIR_ERROR	The error message returned because the storage directory failed to be created.

Internal SDK call errors

Error code	Error message	Description
240030	CREATE_NUI_ERROR	The error message returned because the engine failed to be created.
240031	TEXT_DIALOG_START_FAIL	The error message returned because the text comprehension task failed to be started.
240032	TEXT_CANCEL_START_FAIL	The error message returned because the text comprehension task failed to be canceled.
240033	WUW_DUPLICATE	The error message returned because you specified repeated dynamic wake-up words.

Client engine errors

Error code	Error message	Description
240040	CEI_INIT_FAIL	The error message returned because the client engine failed to be initialized.
240041	CEI_SET_PARAM_FAIL	The error message returned because an engine parameter failed to be set.
240042	CEI_COMPILE_GRAMMER_FAIL	The error message returned because the code syntax failed to be compiled.
240043	CEI_STOP_FAIL	The error message returned because the client engine failed to stop the recognition task.
240044	CEI_CANCEL_FAIL	The error message returned because the client engine failed to cancel the recognition task.
240045	CEI_UNLOAD_KWS_FAIL	The error message returned because the client engine failed to cancel the specified wake-up words.
240046	GET_WUW_ERROR	The error message returned because the client engine failed to obtain the specified wake-up words.

Errors related to audio data

Error code	Error message	Description
240050	SELECT_RECORDER_ERROR	The error message returned because the recording device is not properly selected.
240051	UPDATE_AUDIO_ERROR	The error message returned because the audio data failed to be pushed to the server. The general cause is that the size of the audio data exceeds the upper limit.

Error code	Error message	Description
240052	MIC_ERROR	The error message returned because the microphone has not captured any audio data for 2 consecutive seconds.

Errors related to request timeout

Error code	Error message	Description
240080	ENGINE_INIT_TIMEOUT	The error message returned because the request to initialize the engine timed out.
240081	SET_PARAM_TIMEOUT	The error message returned because the request to set parameters timed out.
240082	SET_WUW_TIMEOUT	The error message returned because the request to set wake-up words timed out.
240083	SELECT_RECORDER_TIMEOUT	The error message returned because the request to select the recording device timed out.
240084	STOP_TIMEOUT	The error message returned because the request to terminate the dialog timed out.
240085	ASR_ENGINE_STOP_TIMEOUT	The error message returned because the request to disable the engine timed out.
240086	UNLOAD_DYNAMIC_WUW_TIMEOUT	The error message returned because the request to cancel the dynamic wake-up words timed out.
240087	ADD_DYNAMIC_WUW_TIMEOUT	The error message returned because the request to add the dynamic wake-up words timed out.
240100	WAIT_TIMEOUT	The error message returned because the engine request timed out.

Error code	Error message	Description
240101	HANDLE_API_TIMEOUT	The error message returned because the API request timed out.

Network errors

Error code	Error message	Description
240060	CREATE_DA_REQUEST_ERROR	The error message returned because the dialog assistant failed to be created.
240061	START_DA_REQUEST_ERROR	The error message returned because the dialog assistant failed to be started.
240062	DEFAULT-NLS_ERROR	The error message returned because an error occurred on the server. Note: This error also generates an error code that is returned by the server. For more information, see the Error codes returned by the server table.
240063	SSL_ERROR	The error message returned because the Secure Sockets Layer (SSL) certificate failed to be created.
240064	SSL_CONNECT_FAILED	The error message returned because the SSL connection failed.
240065	HTTP_CONNECT_FAILED	The error message returned because the HTTP connection failed.
240066	DNS_FAILED	The error message returned because the Domain Name System (DNS) resolution failed.
240067	CONNECT_FAILED	The error message returned because the socket connection failed.
240068	SERVER_NOT_ACCESS	The error message returned because the server cannot be accessed.

Error code	Error message	Description
240069	SOCKET_CLOSED	The error message returned because the socket is closed.
240070	AUTH_FAILED	The error message returned because the authentication failed.
240071	HTTPDNS_FAILED	The error message returned because the connection between the server and the client by using the specified IP address failed.

Errors related to network timeout

Error code	Error message	Description
240090	UPDATE_CONTEXT_TIMEOUT	The error message returned because the request to update the client timed out.
240091	CONNECTION_TIMEOUT	The error message returned because the network connection timed out.
240092	PARTIAL_ASR_TIMEOUT	The error message returned because the request to obtain the intermediate recognition result timed out.
240093	ASR_TIMEOUT	The error message returned because the request to obtain the final recognition result timed out.
240094	DIALOG_TIMEOUT	The error message returned because the request to obtain the dialog processing result timed out.
240095	WWV_TIMEOUT	The error message returned because the request to obtain the wake-up result of the server timed out.

Error codes returned by the server

If the client receives an `EVENT_ASR_ERROR` event, and the error code and error message returned by the SDK are 240062 and `DEFAULT-NLS_ERROR` respectively, the status parameter in the header object of the event displays the error code that is returned by the server.

Error code	Cause	Solution
40000001	The authentication failed.	Check whether the token expires or is invalid.
40000002	The message is invalid.	Check whether the message that is sent meets the requirement.
403	The token expires or is invalid.	1. Check whether the token expires. 2. Check whether the token is valid.
40000004	The idle connection timed out.	Check whether no data has been sent to the server for 10 consecutive seconds.
40000005	The number of requests exceeds the upper limit.	Check whether the number of concurrent connections or the queries per second (QPS) exceeds the upper limit. If the number of concurrent connections exceeds the upper limit, we recommend that you upgrade Intelligent Speech Interaction from the trial edition to Commercial Edition. If you have upgraded the service to Commercial Edition, we recommend that you purchase more resources for higher concurrency.
40000000	A client error occurred. This is the default client error code.	Check the error message or submit a ticket.
41010120	The client timed out.	The client has not sent audio data for 10 or more consecutive seconds.
50000000	A server error occurred. This is the default server error code.	If the error code is occasionally returned, ignore it. If the error code is returned multiple times, submit a ticket.
50000001	An internal call error occurred.	If the error code is occasionally returned, ignore it. If the error code is returned multiple times, submit a ticket.
52010001	An internal call error occurred.	If the error code is occasionally returned, ignore it. If the error code is returned multiple times, submit a ticket.

Error code	Cause	Solution
40010001	The method is not supported.	If you use the SDK, submit a ticket.
40010002	The instruction is not supported.	If you use the SDK, submit a ticket.
40010003	The instruction is invalid.	If you use the SDK, submit a ticket.
40010004	The client is disconnected.	Check whether the client is disconnected before the server completes the requested task.
40010005	The task is in an abnormal state.	Check whether the instruction is supported in the current task status.
40020105	The specified appkey is invalid.	Resolve the route to check whether the application exists.
40020106	The specified appkey and token do not match.	Check whether the appkey of the application is valid and belongs to the same Alibaba Cloud account as the token.
40020503	Resource Access Management (RAM) user authentication fails.	Use your Alibaba Cloud account to authorize the RAM user to access the POP API.
41040201	The client has not sent data for 10 consecutive seconds.	Check the network connection or whether no business data needs to be sent.
41040202	The client sends data at a high transmission rate and consumes all resources of the server.	Check whether the client sends data at an appropriate transmission rate, for example, at the real-time factor of 1:1.
41040203	The client sends audio data in an invalid audio coding format.	Convert the audio coding format of audio data to a format supported by the SDK.
41040204	The client calls methods in an invalid order.	Check whether the client calls the relevant method to send a request before it calls other methods.
41040205	The specified MAXSILENCE_PARAM parameter is invalid.	Check whether the value of the MAXSILENCE_PARAM parameter is in the range of 200 to 2000.

Error code	Cause	Solution
41050008	The specified sampling rate does not match that of the selected model.	Check whether the audio sampling rate specified for the service call matches the audio sampling rate of the automatic speech recognition (ASR) model that is bound to the application in the console.
51040101	An internal error occurred on the server.	Resolve the error based on the error message.
51040102	Reserved.	N/A
51040103	The real-time speech recognition service is unavailable.	Check whether the number of real-time speech recognition tasks exceeds the upper limit.
51040104	The request for real-time speech recognition timed out.	Check the logs of the real-time speech recognition service.
51040105	The real-time speech recognition service failed to be called.	Check whether the real-time speech recognition service is enabled and whether the port works properly.
51040106	The load balancing of the real-time speech recognition service failed and the client failed to obtain the IP address of the real-time speech recognition service.	Check whether the real-time speech recognition server in the configured virtual private cloud (VPC) works properly.

2.SDK for C++

This topic describes how to use the C++ SDK provided by Alibaba Cloud Intelligent speech interaction, including the SDK installation method and SDK sample code.

Note

- The latest version of the SDK for C++ is 3.0.8, which was released on January 09, 2020. This version applies only to the Linux operating system. The Windows operating system is not supported.
- Before you use this SDK, make sure that you understand how this SDK works. For more information, see [Overview](#).
- The methods of this SDK version are different from those of the earlier version. If you are familiar with the earlier version, pay attention to the updated methods described in this topic.

Download and installation

Download the SDK:

To download the SDK for C++, click [here](#). The compressed package contains the following files or folders:

- CMakeLists.txt: the CMakeList file of the demo project.
- readme.txt: the SDK description.
- release.log: the release notes.
- version: the version number.
- build.sh: the demo compilation script.
- lib: the SDK libraries.
- build: the compilation directory.
- demo: the folder that contains demo.cpp files, which are the configuration files of various Intelligent Speech Interaction services. The following table describes the files contained in the folder.

File name	Description
speechRecognizerDemo.cpp	The demo of short sentence recognition.
speechSynthesizerDemo.cpp	The demo of speech synthesis.
speechTranscriberDemo.cpp	The demo of real-time speech recognition.
speechLongSynthesizerDemo.cpp	The demo of long-text-to-speech synthesis.

File name	Description
test0.wav/test1.wav	The 16-bit audio files with a sampling rate of 16,000 Hz for testing.

- include: the folder that contains SDK header files. The following table describes the files contained in the folder.

File name	Description
nlsClient.h	The header file of the NlsClient object.
nlsEvent.h	The header file of callback events.
speechRecognizerRequest.h	The header file of short sentence recognition.
speechSynthesizerRequest.h	The header file of speech synthesis and long-text-to-speech synthesis.
speechTranscriberRequest.h	The header file of real-time speech recognition.

Compile and run the demo project :

1. Check the local operating system to ensure that required tools are installed based on the following minimum requirements:
 - i. Cmake 3.1
 - ii. Glibc 2.5
 - iii. Gcc 4.1.2
2. Run the following script on the Linux terminal.

```
mkdir build
cd build && cmake .. && make
cd... /demo# The following executable demo programs are generated: srDemo for short sentence recognition, stDemo for real-time speech recognition, syDemo for speech synthesis, and syLongDemo for long-text-to-speech synthesis.
./stDemo appkey <yourAccessKey Id> <yourAccessKey Secret> # The data is used for testing.
```

Key objects

- Basic objects
 - NlsClient: the speech processing client, which is equivalent to a factory for all speech processing classes. You can globally create an NlsClient object.

- `NlsEvent`: the event object. You can use this object to obtain the request status code, response from the server, and error message.
- Recognition object
`SpeechTranscriberRequest`: The request object of real-time speech recognition. It is used for real-time speech recognition.

Error codes of the SDK for C++

Error code	Error message	Description and solution
10000001	SSL: couldn't create a	The error message returned because an internal error has occurred. Try again later.
10000002	An official OpenSSL error message.	The error message returned because an internal error has occurred. Resolve the error based on the error message and try again later.
10000003	A system error message.	The error message returned because a system error has occurred. Resolve the error based on the error message.
10000004	URL: The url is empty.	The error message returned because no endpoint is specified. Check whether an endpoint is specified.
10000005	URL: Could not parse WebSocket url.	The error message returned because the specified endpoint is invalid. Check whether the specified endpoint is correct.

Error code	Error message	Description and solution
10000006	MODE: unsupport mode.	The error message returned because the specified Intelligent Speech Interaction service is not supported. Check whether the Intelligent Speech Interaction service is correctly configured.
10000007	JSON: Json parse failed.	The error message returned because the server returns an invalid response. Submit a ticket and provide the task ID to Alibaba Cloud.
10000008	WEBSOCKET: unkown head type.	The error message returned because the server returns an invalid WebSocket type. Submit a ticket and provide the task ID to Alibaba Cloud.
10000009	HTTP: connect failed.	The error message returned because the client fails to connect to the server. Check the network and try again later.
Official HTTP status code	HTTP: Got bad status.	The error message returned because an internal error has occurred. Resolve the error based on the error message.
System error code	IP: ip address is not valid.	The error message returned because the IP address is invalid. Resolve the error based on the error message.
System error code	ENCODE: convert to utf8 error.	The error message returned because the file fails to be converted to the UTF-8 format. Resolve the error based on the error message.

Error code	Error message	Description and solution
10000010	<code>please check if the memory is enough.</code>	The error message returned because the memory is insufficient. Check the memory of the local device.
10000011	<code>Please check the order of execution.</code>	The error message returned because the client calls methods in an invalid order. For example, if the client receives a failed or complete message, the SDK disconnects the client from the server. If the client calls the relevant method to send data, this error message is returned.
10000012	<code>StartCommand/StopCommand Send failed.</code>	The error message returned because the request contains invalid parameters. Check the settings of request parameters.
10000013	<code>The sent data is null or dataSize <= 0.</code>	The error message returned because the client sends invalid data. Check the settings of request parameters.
10000014	<code>Start invoke failed.</code>	The error message returned because the start method times out. Call the stop method to release resources, and then start the recognition process again.
10000015	<code>connect failed.</code>	The error message returned because the connection between the client and the server fails. Release resources and start the recognition process again.

Service status codes

For more information about the service status codes, see the "Service status codes" section of the [API reference](#).

Sample code

Note

- The demo uses an audio file with the sampling rate of 16,000 Hz. To obtain correct recognition results, set the model to universal model for the project to which the appkey is bound in the Intelligent Speech Interaction console. You must select a model that matches the audio sampling rate based on your business scenario. For more information about model setting, see [Manage projects](#).
- You can obtain the complete sample code from the speechTranscriberDemo.cpp file in the demo folder of the SDK package.

```
#include <pthread.h>
#include <unistd.h>
#include <ctime>
#include <stdlib.h>
#include <string.h>
#include <string>
#include <vector>
#include <fstream>
#include "nlsClient.h"
#include "nlsEvent.h"
#include "speechTranscriberRequest.h"
#include "nlsCommonSdk/Token.h"

#define FRAME_SIZE 3200
#define SAMPLE_RATE 16000
using namespace AlibabaNlsCommon;
using AlibabaNls::NlsClient;
using AlibabaNls::NlsEvent;
using AlibabaNls::LogDebug;
using AlibabaNls::LogInfo;
using AlibabaNls::SpeechTranscriberRequest;

// Customize the thread parameters.
struct ParamStruct {
    std::string fileName;
    std::string token;
    std::string appkey;
};

// Customize the callback parameters.
struct ParamCallBack {
    int userId;
    char userInfo[10];
};

// Specify a token for service authentication and the timestamp that indicates the validity
period of the token. The token and timestamp can be used throughout the project.
// Each time before you call the service, you must check whether the specified token expires.
// If the token expires, you can use the AccessKey ID and AccessKey secret of your Alibaba
```

```

Cloud account to obtain a new token. Then, reset the g_token and g_expireTime parameters.
// Note: Do not obtain a new token each time you call the real-time speech recognition service. A token can be used for service authentication when it is valid. In addition, you can use the same token for all Intelligent Speech Interaction services.
std::string g_akId = "";
std::string g_akSecret = "";
std::string g_token = "";
long g_expireTime = -1;

// Obtain a new token by using the AccessKey ID and AccessKey secret and obtain a timestamp for the validity period of the token.
// A token can be used when it is valid. You can use the same token for multiple processes, multiple threads, or multiple applications. We recommend that you apply for a new token when the current token is about to expire.
int generateToken(std::string akId, std::string akSecret, std::string* token, long* expireTime) {
    NlsToken nlsTokenRequest;
    nlsTokenRequest.setAccessKeyId(akId);
    nlsTokenRequest.setKeySecret(akSecret);

    if (-1 == nlsTokenRequest.applyNlsToken()) {
        // Receive the error message.
        printf("generateToken Failed: %s\n", nlsTokenRequest.getErrorMsg());
        return -1;
    }

    *token = nlsTokenRequest.getToken();
    *expireTime = nlsTokenRequest.getExpireTime();
    return 0;
}

// @brief: Call the sendAudio method to obtain the sleep duration of audio data sending.
// @param dataSize: the size of the audio data to be sent.
// @param sampleRate: the audio sampling rate. Supported sampling rates include 8,000 Hz and 16,000 Hz.
// @param compressRate: the data compression rate. Set this parameter to 10 for Opus-encoded audio data with a sampling rate of 16,000 Hz and a data compression rate of 10:1. Set this parameter to 1 if the data is not compressed.
// @return: the sleep duration after the audio data is sent.
// @note: For 16-bit pulse-code modulation (PCM)-encoded audio data with a sampling rate of 8,000 Hz, we recommend that you set the sleep duration to 100 ms for every 1,600 bytes sent.
// For 16-bit PCM-encoded audio data with a sampling rate of 16,000 Hz, we recommend that you set the sleep duration to 100 ms for every 3,200 bytes sent.
// For audio data in other formats, calculate the sleep duration based on the compression rate. For example, if the compression rate is 10:1 for Opus-encoded audio data with a sampling rate of 16,000 Hz, the sleep duration is calculated in the following way: 3200/10 = 320 ms.
unsigned int getSendAudioSleepTime(int dataSize, int sampleRate, int compressRate) {
    // Only 16-bit audio data is supported.
    const int sampleBytes = 16;
    // Only mono audio data is supported.
    const int soundChannel = 1;
    // Recommend sampling rate which indicates the size of data in the specified audio format.

```

```

    // The current sampling rate, which indicates the size of data in the specified audio bit
    // depth sampled per second.
    int bytes = (sampleRate * sampleBytes * soundChannel) / 8;
    // The current sampling rate, which indicates the size of data in the specified audio bit
    // depth sampled per millisecond.
    int bytesMs = bytes / 1000;
    // The sleep duration is the size of the audio data to be sent divided by the sampling
    // rate per millisecond.
    int sleepMs = (dataSize * compressRate) / bytesMs;
    return sleepMs;
}

// @brief: Call the start method to connect the client to the server. The SDK reports a started
// event in an internal thread.
// @param cbEvent: the syntax of the event in a callback. For more information, see the Nls
// Event.h file.
// @param cbParam: the custom parameter in a callback. The default value is null. You can set
// this parameter based on your business requirements.
void onTranscriptionStarted(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // The following code demonstrates how to obtain details of the started event and customize
    // callback parameters.
    printf("onTranscriptionStarted: %d\n", tmpParam->userId);
    // The ID of the current recognition task. The task ID is the unique identifier that indicates
    // the interaction between the caller and the server. You must record the task ID. If an error
    // occurs, you can submit a ticket and provide the task ID to Alibaba Cloud to facilitate
    // troubleshooting.
    printf("onTranscriptionStarted: status code=%d, task id=%s\n", cbEvent->getStatusCode(),
        cbEvent->getTaskId());
    // Obtain the complete information returned by the server.
    // printf("onTranscriptionStarted: all response=%s\n", cbEvent->getAllResponse());
}

// @brief: The server detects the beginning of a sentence. Then, the SDK reports a Sentence
// Begin event in an internal thread.
// @param cbEvent: the syntax of the event in a callback. For more information, see the Nls
// Event.h file.
// @param cbParam: the custom parameter in a callback. The default value is null. You can set
// this parameter based on your business requirements.
void onSentenceBegin(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // The following code demonstrates how to obtain details of the SentenceBegin event and
    // customize callback parameters.
    printf("onSentenceBegin: %d\n", tmpParam->userId);
    // The ID of the current recognition task. The task ID is the unique identifier that indicates
    // the interaction between the caller and the server. You must record the task ID. If an error
    // occurs, you can submit a ticket and provide the task ID to Alibaba Cloud to facilitate
    // troubleshooting.
    printf("onSentenceBegin: status code=%d, task id=%s, index=%d, time=%d\n", cbEvent->getStatusCode(),
        cbEvent->getTaskId(),
        cbEvent->getSentenceIndex(), // The sequence number of the sentence, which
        // starts from 1.
        cbEvent->getSentenceTime() // The duration of the audio stream that has been
        // processed, in milliseconds.
    );
}

```

```

    // Obtain the complete information returned by the server.
    //printf("onTranscriptionStarted: all response=%s\n", cbEvent->getAllResponse());
}

// @brief: The server detects the end of a sentence. Then, the SDK reports a SentenceEnd event in an internal thread.
// @param cbEvent: the syntax of the event in a callback. For more information, see the NlsEvent.h file.
// @param cbParam: the custom parameter in a callback. The default value is null. You can set this parameter based on your business requirements.
void onSentenceEnd(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // The following code demonstrates how to obtain details of the SentenceEnd event and customize callback parameters.
    printf("onSentenceEnd: %d\n", tmpParam->userId);
    // The ID of the current recognition task. The task ID is the unique identifier that indicates the interaction between the caller and the server. You must record the task ID. If an error occurs, you can submit a ticket and provide the task ID to Alibaba Cloud to facilitate troubleshooting.
    printf("onSentenceEnd: status code=%d, task id=%s, index=%d, time=%d, begin_time=%d, result=%s\n", cbEvent->getStatusCode(), cbEvent->getTaskId(),
        cbEvent->getSentenceIndex(), // The sequence number of the sentence, which starts from 1.
        cbEvent->getSentenceTime() // The duration of the audio stream that has been processed, in milliseconds.
        cbEvent->getSentenceBeginTime(), // The time when the SentenceBegin event occurred.
        cbEvent->getResult() // The recognition result of the current sentence.
    );
    // << ", confidence: " << cbEvent->getSentenceConfidence() // The confidence level of the recognition result. Valid values: 0.0 to 1.0. A larger value indicates a higher confidence level.
    // << ", stashResult begin_time: " << cbEvent->getStashResultBeginTime() // The time when the next sentence begins.
    // << ", stashResult current_time: " << cbEvent->getStashResultCurrentTime() // The current time when the next sentence is being processed.
    // << ", stashResult Sentence_id: " << cbEvent->getStashResultSentenceId() //The ID of the sentence.
    // << ", stashResult Text: " << cbEvent->getStashResultText() // The beginning words of the next sentence.
    // Obtain the complete information returned by the server.
    //printf("onTranscriptionStarted: all response=%s\n", cbEvent->getAllResponse());
}

// @brief: The recognition result is updated. When the SDK receives the updated result, the SDK reports a ResultChanged event in an internal thread.
// @param cbEvent: the syntax of the event in a callback. For more information, see the NlsEvent.h file.
// @param cbParam: the custom parameter in a callback. The default value is null. You can set this parameter based on your business requirements.
void onTranscriptionResultChanged(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // The following code demonstrates how to obtain details of the ResultChanged event and customize callback parameters.

```

```

    printf("onTranscriptionResultChanged: %d\n", tmpParam->userId);
    // The ID of the current recognition task. The task ID is the unique identifier that indicates the interaction between the caller and the server. You must record the task ID. If an error occurs, you can submit a ticket and provide the task ID to Alibaba Cloud to facilitate troubleshooting.
    printf("onTranscriptionResultChanged: status code=%d, task id=%s, index=%d, time=%d, result=%s\n", cbEvent->getStatusCode(), cbEvent->getTaskId(),
        cbEvent->getSentenceIndex(), // The sequence number of the sentence, which starts from 1.
        cbEvent->getSentenceTime() // The duration of the audio stream that has been processed, in milliseconds.
        cbEvent->getResult() // The recognition result of the current sentence.
    );
    // Obtain the complete information returned by the server.
    //printf("onTranscriptionStarted: all response=%s\n", cbEvent->getAllResponse());
}

// @brief: When the server stops the real-time recognition of the audio stream, the SDK reports a Completed event in an internal thread.
// @note: After a Completed event is reported, the SDK disconnects the client from the server in an internal thread. At this time, if you call the sendAudio method, -1 is returned. Stop sending audio data in this case.
// @param cbEvent: the syntax of the event in a callback. For more information, see the NlsEvent.h file.
// @param cbParam: the custom parameter in a callback. The default value is null. You can set this parameter based on your business requirements.
void onTranscriptionCompleted(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // The following code demonstrates how to obtain details of the Completed event and customize callback parameters.
    printf("onTranscriptionCompleted: %d\n", tmpParam->userId);
    printf("onTranscriptionCompleted: status code=%d, task id=%s\n", cbEvent->getStatusCode(), cbEvent->getTaskId());
}

// @brief: When an error occurs during the recognition process that covers calls of the start, send, and stop methods, the SDK reports a TaskFailed event in an internal thread.
// @note: After a TaskFailed event is reported, the SDK disconnects the client from the server in an internal thread. At this time, if you call the sendAudio method, -1 is returned. Stop sending audio data in this case.
// @param cbEvent: the syntax of the event in a callback. For more information, see the NlsEvent.h file.
// @param cbParam: the custom parameter in a callback. The default value is null. You can set this parameter based on your business requirements.
void onTaskFailed(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // The following code demonstrates how to obtain details of the TaskFailed event and customize callback parameters.
    printf("onTaskFailed: %d\n", tmpParam->userId);
    printf("onTaskFailed: status code=%d, task id=%s, error message=%s\n", cbEvent->getStatusCode(), cbEvent->getTaskId(), cbEvent->getErrorMessage());
    // Obtain the complete information returned by the server.
    //printf("onTaskFailed: all response=%s\n", cbEvent->getAllResponse());
}

```

```

// @brief: The SDK reports the final recognition result in an internal thread.
// @param cbEvent: the syntax of the event in a callback. For more information, see the nls
Event.h file.
// @param cbParam: the custom parameter in a callback. The default value is null. You can s
et this parameter based on your business requirements.
void onSentenceSemantics(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // The following code demonstrates how to obtain details of the SentenceSemantics event
and customize callback parameters.
    printf("onSentenceSemantics: %d\n", tmpParam->userId);
    // Obtain the complete information returned by the server.
    printf("onSentenceSemantics: all response=%s\n", cbEvent->getAllResponse());
}

// @brief: When the recognition ends or an error occurs during the recognition process, the
SDK disconnects the client from the server and reports a ChannelClosed event in an internal
thread.
// @param cbEvent: the syntax of the event in a callback. For more information, see the nls
Event.h file.
// @param cbParam: the custom parameter in a callback. The default value is null. You can s
et this parameter based on your business requirements.
void onChannelClosed(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    delete tmpParam; // The recognition process ends and the callback parameter is released
    .
}

// The worker thread.
void* pthreadFunc(void* arg) {
    int sleepMs = 0;
    ParamCallBack *cbParam = NULL;
    // Initialize custom callback parameters. The following settings are used as an example
to demonstrate how to pass parameters. The settings have no effect on the demo.
    // The settings of callback parameters are stored in a heap. When the SDK clears the re
quest objects, it clears the parameter settings as well. You do not need to manually releas
e the parameters.
    cbParam = new ParamCallBack;
    cbParam->userId = 1234;
    strcpy(cbParam->userInfo, "User.");

    // 1. Obtain parameters such as the token and configuration files from custom thread pa
rameters.
    ParamStruct* tst = (ParamStruct*)arg;
    if (tst == NULL) {
        printf("arg is not valid\n");
        return NULL;
    }

    /* Open the audio file and obtain audio data.*/
    std::ifstream fs;
    fs.open(tst->fileName.c_str(), std::ios::binary | std::ios::in);
    if (!fs) {
        printf("%s isn't exist..\n", tst->fileName.c_str());
    }
}

```

```

        return NULL;
    }

    // 2. Create the SpeechTranscriberRequest object of real-time speech recognition.
    SpeechTranscriberRequest* request = NlsClient::getInstance()->createTranscriberRequest(
    );
    if (request == NULL) {
        printf("createTranscriberRequest failed.\n");
        return NULL;
    }

    request->setOnTranscriptionStarted(onTranscriptionStarted, cbParam);           //
    Set a callback to be fired when the speech recognition starts.
    request->setOnTranscriptionResultChanged(onTranscriptionResultChanged, cbParam); //
    Set a callback to be fired when a recognition result is returned.
    request->setOnTranscriptionCompleted(onTranscriptionCompleted, cbParam);       //
    Set a callback to be fired when the speech recognition is completed.
    request->setOnSentenceBegin(onSentenceBegin, cbParam);                         //
    Set a callback to be fired when the beginning of a sentence is detected.
    request->setOnSentenceEnd(onSentenceEnd, cbParam);                             //
    Set a callback to be fired when the end of a sentence is detected.
    request->setOnTaskFailed(onTaskFailed, cbParam);                               //
    Set a callback to be fired when an error occurs.
    request->setOnChannelClosed(onChannelClosed, cbParam);                         //
    Set a callback to be fired when the TCP connection set up for the recognition task is closed.
    request->setOnSentenceSemantics(onSentenceSemantics, cbParam);                 //
    Set a callback to be fired when an updated recognition result is returned. The result is returned when the enable_nlp parameter is used.

    request->setAppKey(tst->appkey.c_str()); // Specify the appkey. This parameter is required. If you do not have an appkey, obtain it as instructed on the Alibaba Cloud international site (alibabacloud.com).
    request->setFormat("pcm"); // Specify the audio encoding format. Default value: pcm.
    request->setSampleRate(SAMPLE_RATE); // Specify the audio sampling rate. This parameter is optional. Valid values: 16000 and 8000. Default value: 16000.
    request->setIntermediateResult(true); // Specify whether to return intermediate recognition results. This parameter is optional. Default value: false.
    request->setPunctuationPrediction(true); // Specify whether to add punctuation marks during post-processing. This parameter is optional. Default value: false.
    request->setInverseTextNormalization(true); // Specify whether to convert Chinese numerals to Arabic numerals during post-processing. This parameter is optional. Default value: false.

    // Specify the threshold for detecting the end of a sentence. If the silence duration exceeds the specified threshold, the system determines the end of a sentence. Unit: milliseconds. Valid values: 200 to 2000. Default value: 800.
    //request->setMaxSentenceSilence(800);
    //request->setCustomizationId("TestId_123"); // Specify the ID of the custom model. This parameter is optional.
    //request->setVocabularyId("TestId_456"); // Specify the vocabulary ID of custom extensive hotwords. This parameter is optional.
    // Pass custom or advanced parameters in the JSON format of {"key": "value"}.
    //request->setAdvancedParameters("{\"key\": \"value\"}");

```



```

//request->setPayloadParam("{\"vad_model\": \"farfield\"}");
// Specify whether to return the recognition results of words.
request->setPayloadParam("{\"enable_words\": true}");

// Specify whether to enable voice activity detection (VAD). Default value: false. We recommend that you do not enable VAD unless otherwise required.
//request->setPayloadParam("{\"enable_semantic_sentence_detection\": false}");
// Specify whether to enable disfluency detection. Default value: false. We recommend that you do not enable disfluency detection unless otherwise required.
//request->setPayloadParam("{\"disfluency\": true}");

// Specify the ID of the VAD mode. By default, this parameter is left empty. We recommend that you do not set this parameter unless otherwise required.
//request->setPayloadParam("{\"vad_model\": \"farfield\"}");
// Specify whether to ignore the recognition timeout issue of a single sentence.
//request->setPayloadParam("{\"enable_ignore_sentence_timeout\": false}");
// Specify whether to enable post-processing for VAD. Default value: false. We recommend that you do not enable post-processing unless otherwise required.
//request->setPayloadParam("{\"enable_vad_unify_post\": true}");

request->setToken(tst->token.c_str());

// 3. Call the start method in asynchronous callback mode. If the method is called, a started event is returned. If the method fails, a TaskFailed event is returned.
if (request->start() < 0) {
    printf("start() failed. may be can not connect server. please check network or firewall\n");
    NlsClient::getInstance()->releaseTranscriberRequest(request); // The start method fails. The SpeechTranscriberRequest object is released.
    return NULL;
}

while (!fs.eof()) {
    uint8_t data[FRAME_SIZE] = {0};

    fs.read((char *)data, sizeof(uint8_t) * FRAME_SIZE);
    size_t nlen = fs.gcount();
    if (nlen <= 0) {
        continue;
    }

    // 4. Send audio data. If the sendAudio method returns -1, indicating that data fails to be sent, the client stops sending data.
    int ret = request->sendAudio(data, nlen);
    if (ret < 0) {
        // Indicate that data fails to be sent. The client stops sending data cyclically.
        printf("send data fail.\n");
        break;
    }

    // Set the transmission speed of data sending:
    // If you recognize a real-time recording, you do not need to specify the transmission speed by using the sleep method.
    // If you recognize an audio file, you must specify the transmission speed. Ensure

```

```

// If you recognize an audio file, you must specify the transmission speed. Ensure
that the data size sent per unit interval approaches to the data size of a unit interval in
the audio file.

    sleepMs = getSendAudioSleepTime(nlen, SAMPLE_RATE, 1); // Obtain the sleep duration
based on the size of sent data, audio sampling rate, and data compression rate.

    // 5. Set the latency for audio data sending.
    usleep(sleepMs * 1000);
}

// Close the audio file.
fs.close();

// 6: Notify the server that the audio data is sent.
// Call the stop method in asynchronous callback mode. If the method fails, a TaskFailed
event is returned.
request->stop();
// 7. Release the SpeechRecognizerRequest object after the recognition is completed.
NlsClient::getInstance()->releaseTranscriberRequest(request);
return NULL;
}

// Recognize a single audio file.
int speechTranscriberFile(const char* appkey) {
    // Obtain the timestamp of the current system time to check whether the token expires.
    std::time_t curTime = std::time(0);
    if (g_expireTime - curTime < 10) {
        printf("the token will be expired, please generate new token by AccessKey-ID
and AccessKey-Secret.\n");
        if (-1 == generateToken(g_akId, g_akSecret, &g_token, &g_expireTime)) {
            return -1;
        }
    }

    ParamStruct pa;
    pa.token = g_token;
    pa.appkey = appkey;
    pa.fileName = "test0.wav";

    pthread_t pthreadId;
    // Start a worker thread to perform speech recognition.
    pthread_create(&pthreadId, NULL, &pthreadFunc, (void *)&pa);
    pthread_join(pthreadId, NULL);
    return 0;
}

// Recognize multiple audio files.
// If the SDK uses multiple concurrent threads at a time, the SDK recognizes each audio fil
e in a thread. The SDK does not recognize the same audio file in different threads.
// In the sample code, two threads are used to recognize two audio files.
// If you are a free-trial user, you can make only a maximum of two concurrent calls.
#define AUDIO_FILE_NUMS 2
#define AUDIO_FILE_NAME_LENGTH 32
int speechTranscriberMultFile(const char* appkey) {
    // Obtain the timestamp of the current system time to check whether the token expires.

```

```

        std::time_t curTime = std::time(0);
        if (g_expireTime - curTime < 10) {
            printf("the token will be expired, please generate new token by AccessKey-ID and AccessKey-Secret.\n");
            if (-1 == generateToken(g_akId, g_akSecret, &g_token, &g_expireTime)) {
                return -1;
            }
        }

        char audioFileNames[AUDIO_FILE_NUMS][AUDIO_FILE_NAME_LENGTH] = {"test0.wav", "test1.wav"};
        ParamStruct pa[AUDIO_FILE_NUMS];
        for (int i = 0; i < AUDIO_FILE_NUMS; i++) {
            pa[i].token = g_token;
            pa[i].appkey = appkey;
            pa[i].fileName = audioFileNames[i];
        }

        std::vector<pthread_t> pthreadId(AUDIO_FILE_NUMS);
        // Start two worker threads and recognize two audio files at a time.
        for (int j = 0; j < AUDIO_FILE_NUMS; j++) {
            pthread_create(&pthreadId[j], NULL, &pthreadFunc, (void *)&(pa[j]));
        }
        for (int j = 0; j < AUDIO_FILE_NUMS; j++) {
            pthread_join(pthreadId[j], NULL);
        }

        return 0;
    }

int main(int arc, char* argv[]) {
    if (arc < 4) {
        printf("params is not valid. Usage: ./demo <your appkey> <your AccessKey ID> <your AccessKey Secret>\n");
        return -1;
    }

    std::string appkey = argv[1];
    g_akId = argv[2];
    g_akSecret = argv[3];

    // Configure output logs of the SDK. The configuration is optional. As configured in the following code, the SDK logs are generated in the log-Transcriber.txt file. LogDebug specifies that logs at all levels are generated.
    int ret = NlsClient::getInstance()->setLogConfig("log-transcriber", LogDebug);
    if (-1 == ret) {
        printf("set log failed\n");
        return -1;
    }

    // Start the worker thread.
    NlsClient::getInstance()->startWorkThread(4);

    // Recognize a single audio file.
    speechTranscriberFile(appkey.c_str());
}

```

```
// Recognize multiple audio files.  
// speechTranscriberMultFile(appkey.c_str());  
  
// All the tasks are completed. Release the NlsClient object before the process exits.  
Note that the releaseInstance method is not thread-safe.  
NlsClient::releaseInstance();  
return 0;  
}
```

3.WebSocket

If you do not want to use the SDKs for Intelligent Speech Interaction, or the SDKs for Java, C, or C++ cannot meet your business requirements, you can develop custom programs to access Intelligent Speech Interaction.

Overview

Intelligent Speech Interaction uses the WebSocket protocol to convert voice messages into text in real time. Long voice messages are supported. Commands and events are both data frames of the Text type for WebSocket. You must upload audio streams to the server in the binary frame format. The calling sequence must meet the requirements of WebSocket. Outbound audio data uses the binary frame format of WebSocket. For more information, see [Data Frames](#).

- Supported input format: uncompressed PCM or WAV files with 16-bit sampling and mono channel.
- Supported audio sampling rates: 8,000 Hz and 16,000 Hz.
- Allows you to specify whether to return intermediate results, add punctuation marks during post-processing, and convert Chinese numerals to Arabic numerals.
- Allows you to select linguistic models to recognize voice messages in different languages when you manage projects in the Intelligent Speech Interaction console. For more information, see [Manage projects](#).

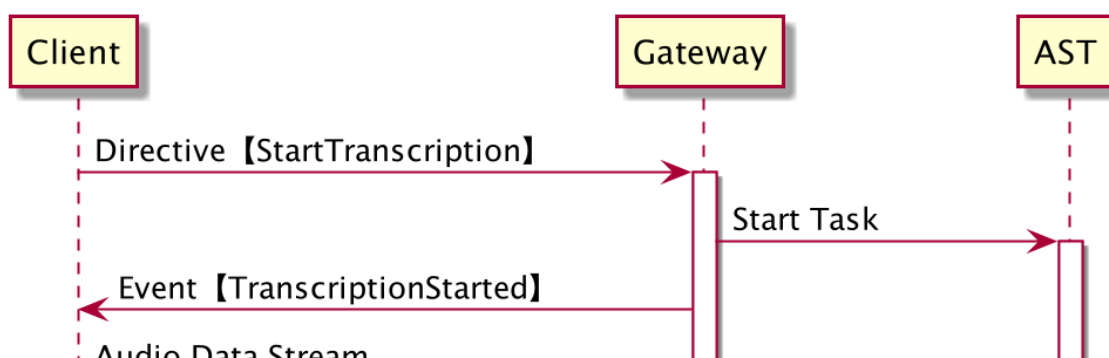
Authentication

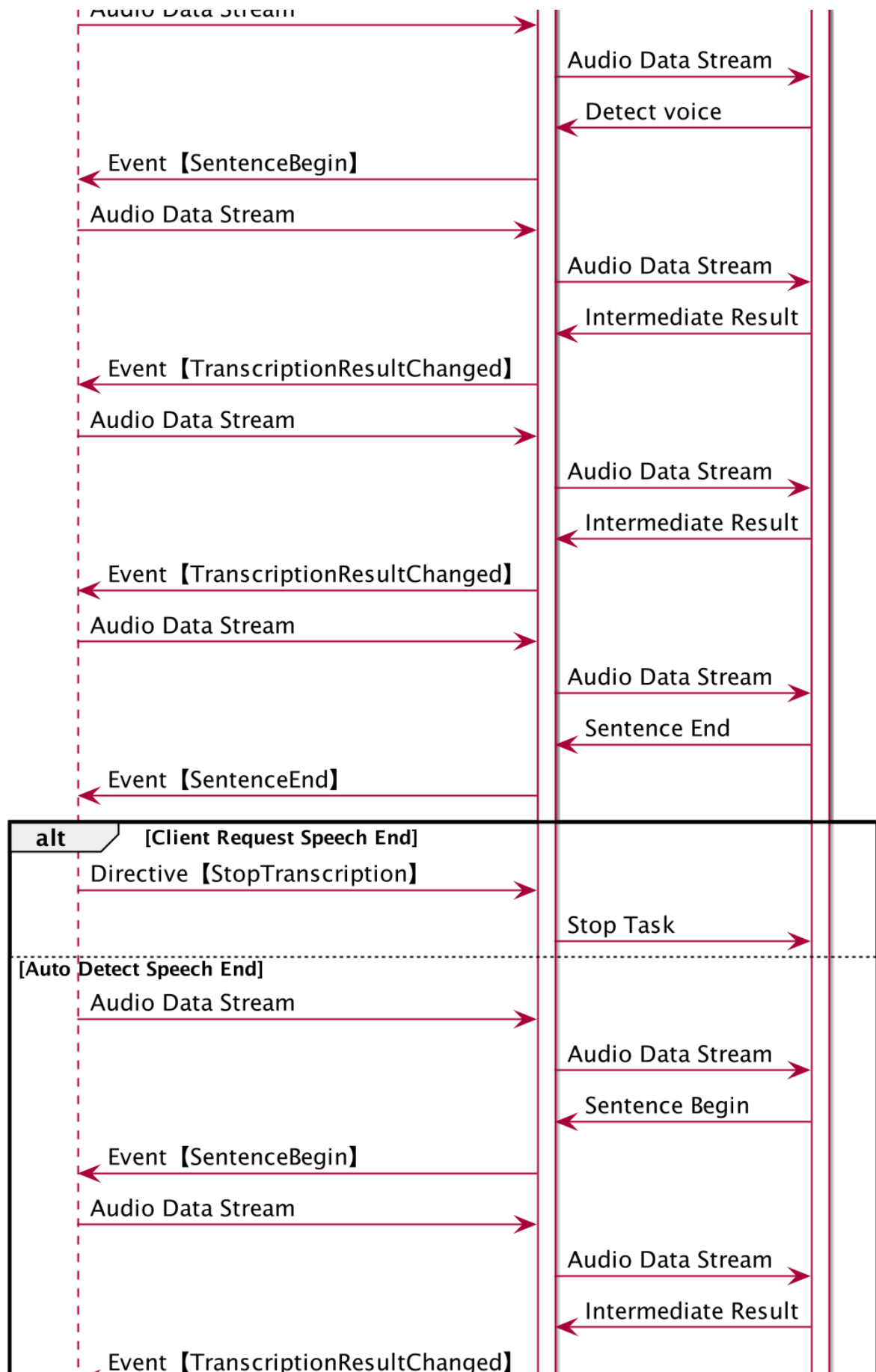
The server uses temporary access tokens for authentication. When you make a request, you must include the access token in the URL. For more information about how to obtain an access token, see [Obtain an access token](#). After you obtain an access token, you can access Intelligent Speech Interaction in one of the following ways.

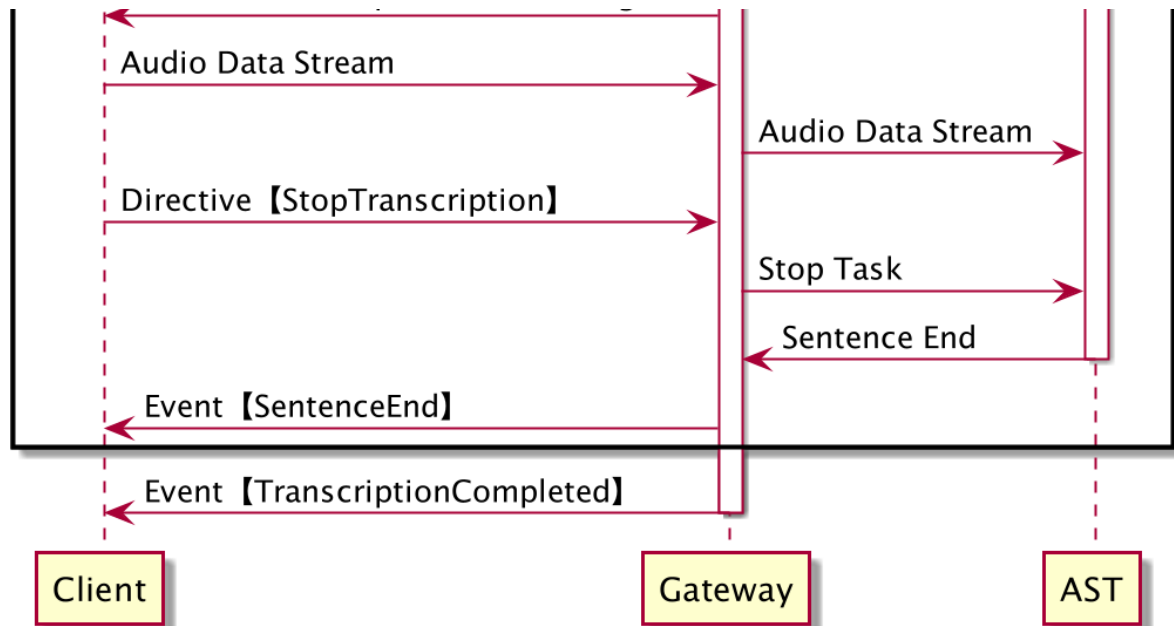
Access type	Description	URL
Access from external networks	You can use the URL to access Intelligent Speech Interaction from all servers.	wss://nls-gateway.ap-southeast-1.aliyuncs.com/ws/v1?token=<your token>

Interaction process

The commands and audio streams must be sent in order as shown in the following figure. Otherwise, interaction with the server fails.







Commands

The request command is used to start or stop a speech recognition task. You must send the request in the JSON format by using the text frame method. You must set basic information about the request in the Header section. A command consists of the Header and Payload sections. The Header section uses a unified format, whereas the Payload section uses different formats for different commands.

1. The Header section


The Header section consists of the following parameters.


Parameter	Type	Required	Description
appkey	String	Yes	The AppKey of your project that is created in the Intelligent Speech Interaction console.
message_id	String	Yes	The 32-bit ID of the request. The ID is randomly generated and unique.
task_id	String	Yes	The 32-bit ID of the speech recognition session. The ID is unique and must remain unchanged throughout the request.
namespace	String	Yes	The name of the service to be accessed. Set the value to SpeechTranscriber.

Parameter	Type	Required	Description
name	String	Yes	The names of the StartTranscription and StopTranscription commands. For more information, see The StartTranscription command and The StopTranscription command .

2. The StartTranscription command

The following table describes the parameters in the Payload section.

Parameter	Type	Required	Description
format	String	No	The audio coding format. Supported format: uncompressed PCM or WAV files with 16-bit sampling and mono channel.
sample_rate	Integer	No	The audio sampling rate. The default rate is 16,000 Hz. After you set this parameter, you must specify a model that is applicable to the scenario and audio sampling rate for your project in the Intelligent Speech Interaction console.
enable_intermediate_result	Boolean	No	Specifies whether to return intermediate results. Default value: False.
enable_punctuation_prediction	Boolean	No	Specifies whether to add punctuation marks during post-processing. Default value: False.
enable_inverse_text_normalization	Boolean	No	Specifies whether to enable inverse text normalization (ITN) during post-processing. Default value: False. If you set this parameter to true, Chinese numerals are converted to Arabic numerals. <div> Note ITN does not apply to words.</div>

Parameter	Type	Required	Description
customization_id	String	No	The ID of the custom linguistic model.
vocabulary_id	String	No	The vocabulary ID of custom popular words.
max_sentence_silence	Integer	No	The threshold for determining the end of a sentence. If the silence duration exceeds the specified threshold, the system determines the end of a sentence. Unit: milliseconds. Valid values: 200 to 2000. Default value: 800.
enable_words	Boolean	No	Specifies whether to return information about words. Default value: False.
enable_ignore_sentence_timeout	Boolean	No	Specifies whether to ignore the recognition timeout of a single sentence in real-time speech recognition. Default value: False.
disfluency	Boolean	No	Specifies whether to enable disfluency detection to remove modal particles and repetitive speech. Default value: False.
speech_noise_threshold	Float	No	<p>The threshold for recognizing audio streams as noise. Valid values: -1 to 1. The following information describes the values:</p> <ul style="list-style-type: none"> A value close to -1 indicates that the audio stream is likely to be recognized as a normal speech. A value close to 1 indicates that the audio stream is likely to be recognized as noise. <div>  Note This parameter is an advanced parameter. Proceed with caution. We recommend that you run tests to find a proper value. </div>

Parameter	Type	Required	Description
enable_semantic_sentence_detection	Boolean	No	Specifies whether to enable semantic sentence segmentation. Default value: False.

Sample code:

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechTranscriber",
    "name": "StartTranscription",
    "appkey": "17d4c634****"
  },
  "payload": {
    "format": "opus",
    "sample_rate": 16000,
    "enable_intermediate_result": true,
    "enable_punctuation_prediction": true,
    "enable_inverse_text_normalization": true
  }
}
```

3. The StopTranscription command

You can run the StopTranscription command to stop a speech recognition task. Therefore, leave the Payload section empty. Sample code:

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechTranscriber",
    "name": "StopTranscription",
    "appkey": "17d4c634****"
  }
}
```

Events

1. The TranscriptionStarted event

The TranscriptionStarted event indicates that the server is ready to recognize speeches and you can send audio streams from the client.

Parameter	Type	Description
session_id	String	If the session_id is set when the client sends the request, the same value is returned. Otherwise, a unique 32-bit ID that is randomly generated is returned.

Sample code:

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechTranscriber",
    "name": "TranscriptionStarted",
    "status": 20000000,
    "status_message": "GATEWAY|SUCCESS|Success."
  },
  "payload": {
    "session_id": "1231231dfdf****"
  }
}
```

2. The SentenceBegin event

The SentenceBegin event indicates that the server detects the start of a sentence.

Parameter	Type	Description
index	Integer	The sequence number of the sentence, which starts from 1.
time	Integer	The start time of a sentence to the start time of the audio stream. Unit: milliseconds.

Sample code:

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechTranscriber",
    "name": "SentenceBegin",
    "status": 20000000,
    "status_message": "GATEWAY|SUCCESS|Success."
  },
  "payload": {
    "index": 1,
    "time": 320
  }
}
```

3. The TranscriptionResultChanged event

The TranscriptionResultChanged event indicates that the recognition result has changed.

Parameter	Type	Description
index	Integer	The sequence number of the sentence, which starts from 1.
time	Integer	The duration of the processed audio stream. Unit: milliseconds.
result	String	The recognition result.
words	Word	The information about words.
status	Integer	The status code.

Word structure:

Parameter	Type	Description
text	String	The text content.
startTime	Integer	The start time of the word.
endTime	Integer	The end time of the word.

Sample code:

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechTranscriber",
    "name": "TranscriptionResultChanged",
    "status": 20000000,
    "status_message": "GATEWAY|SUCCESS|Success."
  },
  "payload": {
    "index": 1,
    "time": 1800,
    "result": "Double Eleven this year",
    "words": [
      {
        "text": "this year",
        "startTime": 1,
        "endTime": 2
      },
      {
        "text": "Double Eleven",
        "startTime": 2,
        "endTime": 3
      }
    ]
  }
}
```

4. The SentenceEnd event

The SentenceEnd event indicates that the server detects the end of a sentence.

Parameter	Type	Description
index	Integer	The sequence number of the sentence, which starts from 1.
time	Integer	The duration of the processed audio stream. Unit: milliseconds.
begin_time	Integer	The time of the SentenceBegin event that corresponds to the sentence. Unit: milliseconds.
result	String	The recognition result.

Parameter	Type	Description
confidence	Double	The accuracy level of the result. Valid values: 0.0 to 1.0. A larger value indicates a higher accuracy level.
words	Word	The information about words.
status	Integer	The status code. Default value: 20000000. After <code>enable_ignore_sentence_timeout</code> is enabled, the error code 51040104 is returned and the current connection is maintained if a timeout occurs.
stash_result	StashResult	The temporarily stored result. After semantic sentence segmentation is enabled, the intermediate result of the next unsegmented sentence is returned.

StashResult structure:

Parameter	Type	Description
sentenceld	Integer	The sequence number of the sentence, which starts from 1.
beginTime	Integer	The start time of the sentence.
text	String	The transcription content.
currentTime	Integer	The time of the audio stream that is being processed.

Sample code:

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechTranscriber",
    "name": "SentenceEnd",
    "status": 20000000,
    "status_message": "GATEWAY|SUCCESS|Success."
  },
  "payload": {
    "index": 1,
    "time": 3260,
    "begin_time": 1800,
    "result": "I want to buy a television this Double Eleven"
  }
}
```

5. The TranscriptionCompleted event

The TranscriptionCompleted event indicates that the speech recognition task is stopped. Sample code:

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechTranscriber",
    "name": "TranscriptionCompleted",
    "status": 20000000,
    "status_message": "GATEWAY|SUCCESS|Success."
  }
}
```