Mobile Speech Client 5.0 PLUS

开发手册

科大讯飞股份有限公司
iFLYTEK CO., LTD.

本手册内容若有变动，恕不另行通知。本手册例子中所用的公司、人名和数据若非特别声明，均属虚构。未得到科大讯飞股份有限公司明确的书面许可，不得以任何目的、以任何形式或手段（电子的或机械的）复制或传播手册的任何部分。

本文档可能涉及科大讯飞股份有限公司的专利（或正在申请的专利）、商标、版权或其他知识产权，除非得到科大讯飞股份有限公司的明确书面许可协议，本文档不授予使用这些专利（或正在申请的专利）、商标、版权或其他知识产权的任何许可协议。

本手册提及的其它产品和公司名称均可能是各自所有者的商标。

本软件产品受最终用户许可协议（EULA）中所述条款和条件的约束，该协议位于产品文档和/或软件产品的联机文档中，如果您使用本产品，表明您已阅读并接受了 EULA 的条款。

目 录

前言 1

第 1 章 概述 3

1.1 MSP 系统网络拓扑结构说明 3

1.2 名词和缩略语 3

1.3 开发包组件 4

1.4 文档说明 5

第 2 章 通用开发接口 6

2.1 通用接口简介 6

2.1.1 通用接口函数列表 6

2.1.2 返回值说明 6

2.2 函数调用 6

2.2.1 MSPLogin 6

2.2.2 MSPUploadData 8

2.2.3 MSPDownloadData 9

2.2.4 MSPSetParam 10

2.2.5 MSPGetParam 11

2.2.6 MSPSearch 11

2.2.7 MSPLogout 12

第 3 章 合成开发接口说明 14

3.1 QTTS 接口简介 14

3.1.1 QTTS 接口函数列表 14

3.1.2 返回值说明 14

3.2 函数调用 14

3.2.1 QTTSSessionBegin 14

3.2.2 QTTSTextPut 17

3.2.3 QTTSAudioGet 18

3.2.4 QTTSAudioInfo 20

3.2.5 QTTSGetParam 21

3.2.6 QTTSSetParam 22

3.2.7 QTTSSessionEnd 23

第 4 章 识别开发接口说明 25

4.1 QISR 接口简介 25
4.1.1 QISR 接口函数列表 25
4.1.2 返回值说明 25
4.2 函数调用 25
4.2.1 QISRSessionBegin 25
4.2.2 QISRAudioWrite 30
4.2.3 QISRGetResult 33
4.2.4 QISRGetParam 34
4.2.5 QISRSessionEnd 35
4.2.6 QISRBuildGrammar 36
4.2.7 QISRUpdateLexicon 38
第 5 章 唤醒开发接口说明 42
5.1 QIVW 接口简介 42
5.1.1 QIVW 接口函数列表 42
5.1.2 返回值说明 42
5.2 函数调用 42
5.2.1 QIVWSessionBegin 42
5.2.2 QIVWRegisterNotify 44
5.2.3 QIVWAudioWrite 46
5.2.4 QIVWSessionEnd 48
第 6 章 错误码的定义 50
6.1 宏 50
6.2 错误码列表 50
第 7 章 发音人列表 61
第 8 章 常见问题解答 62
第 9 章 技术支持 64

前言

欢迎使用讯飞移动语音平台（iFly Mobile Speech Platform，MSP）！

讯飞移动语音平台是基于讯飞公司已有的 ISP 和 IMS 产品，开发出的一款符合移动互联网用户使用的语音应用开发平台，提供语音合成、语音听写、语音识别、声纹识别等服务，为语音应用开发爱好者提供方便易用的开发接口，使得用户能够基于该开发接口进行多种语音应用开发。其主要功能有：

1) 实现基于 HTTP 协议的语音应用服务器，集成讯飞公司最新的语音引擎，支持语音合成、语音听写、语音识别、语音唤醒等服务；
2) 提供基于移动平台和 PC 上的语音客户端子系统，内部集成音频处理和音频编解码模块，提供关于语音合成、语音听写、语音识别和语音唤醒完善的 API。

科大讯飞作为中国最大的智能语音技术提供商，在智能语音技术领域有着长期的研究积累，并在语音合成、语音识别、口语评测等多项技术上拥有国际领先的成果。科大讯飞是我国唯一以语音技术为产业化方向的“国家 863 计划成果产业化基地”、“国家规划布局内重点软件企业”、“国家火炬计划重点高新技术企业”、“国家高技术产业化示范工程”，并被信息产业部确定为中文语音交互技术标准工作组组长单位，牵头制定中文语音技术标准。2003 年，科大讯飞获迄今中国语音产业唯一的“国家科技进步二等奖”，2005 年获中国信息产业自主创新最高荣誉“信息产业重大技术发明奖”。2006 年至 2014 年连续九届在英文语音合成国际大赛（Blizzard Challenge）中蝉联大赛第一名。

基于拥有自主知识产权的世界领先智能语音技术，科大讯飞已推出从大型电信级应用到小型嵌入式应用，从电信、金融等行业到企业和家庭用户，从 PC 到手机到 MP3/MP4/PMP 和玩具，能够满足不同应用环境的多种产品。科大讯飞占有中文语音技术 70%以上市场份额，在电信、金融、电力、社保等主流行业的份额更达 80%以上，开发伙伴超过 800 家，以讯飞为核心的中文语音产业链已初具规模。

本手册主要讲述如何利用讯飞移动语音平台客户端子系统（Mobile Speech Client 5.0，MSCV5+）提供的 API 进行语音合成、语音听写/识别和声纹识别等方面的应用开发，其适用的读者有：

□ 语音服务的提供商

如果您希望根据现有资源扩大某种特定服务的规模，但又担心为某一种服务去开发应用平台代价太大，通过阅读本手册，您会发现，只需要较少的编程就可以充分利用讯飞移动语音平台的诸多优势，实现服务拓展，缩短项目周期。

□ 语音应用的开发商

如果您从事语音系统的二次开发，面向行业级、家庭级等最终用户提供语音合成产品，讯飞移动语音平台简洁明了而又功能全面的 API 将帮助您迅速完成语音应用系统的开发，从而大大节约开发投入，丰富语音服务的内容。

□ **语音技术的爱好者**

如果您对语音应用系统具有浓厚的兴趣，并且具有一定的编程基础，您可以在本手册指导下学习语音方面的应用开发，体验语音世界的无穷乐趣。

本手册基本内容：

第 1 章 概述

简要概括讯飞移动语音平台的原理、特色、功能，以及一些名词、缩略语介绍等。

第 2 章 通用开发接口

介绍 MSCV5+通用性较强的接口功能与应用，并详细列出开发步骤与例程。

第 3 章 合成开发接口

介绍 QTTS 接口函数功能与应用，并详细列出开发步骤与例程。

第 4 章 识别开发接口

介绍 QISR 接口函数功能与应用，并详细列出开发步骤与例程。

第 5 章 唤醒开发接口

介绍 QIVW 接口函数功能与应用，并详细列出开发步骤与例程。

第 6 章 错误码定义

介绍讯飞移动语音平台开发过程中返回的错误码及其定义。

第 7 章 发音人列表

第 8 章 常见问题解答

第 9 章 技术支持

介绍本公司提供的技术支持方式，便于用户迅速获取帮助，提高应用效率。

通过阅读本手册，读者可以：

1) 了解科大讯飞语音应用技术的基本功能与特色
2) 掌握语音合成、语音识别、语音唤醒系统接口规范
3) 了解语音合成系统、语音识别、语音唤醒应用开发的基本思想和方法
4) 利用语音应用系统接口规范地进行二次开发

第 1 章 概述

1.1 MSP 系统网络拓扑结构说明

下图为 MSP 系统网络拓扑结构：

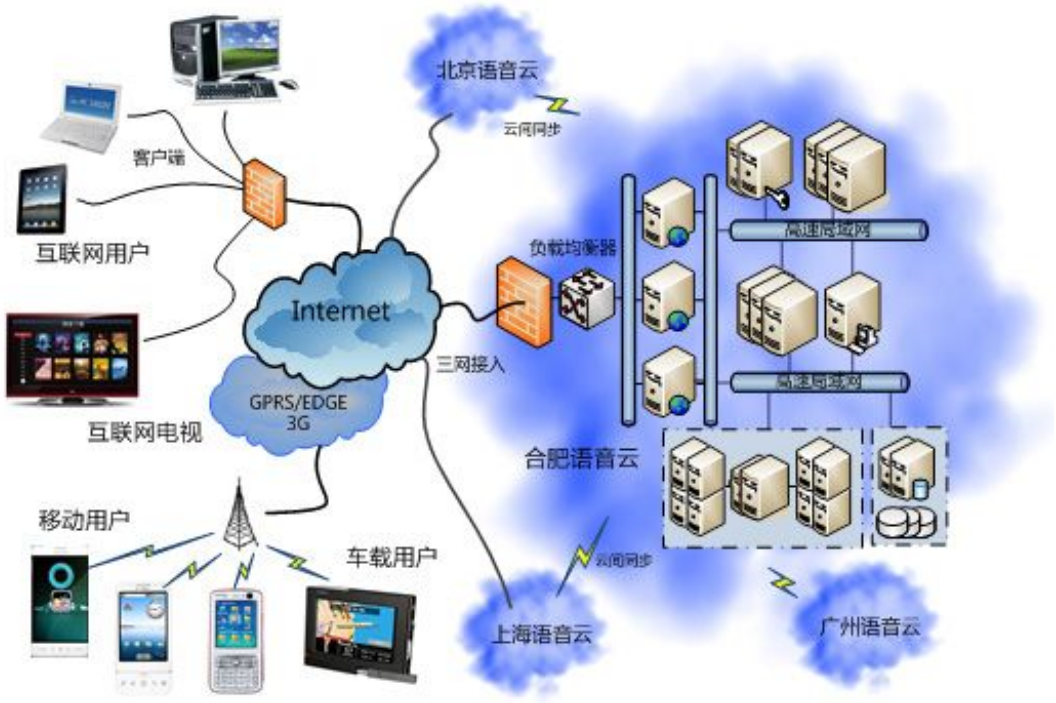

图 1 MSP 系统网络结构

从图中可以看到，完整的 MSP 平台架构在 Internet 上，分为分布式服务器端和客户端两大部分。

服务器端为 MSP 平台的核心部分，提供 HTTP、用户管理、语音服务以及云存储等服务，位于局域网内，对外统一接入 Internet，为客户端提供访问入口。其中：HTTP 服务器负责将客户端发送的服务请求发送至业务服务器，然后由业务服务器按照具体的服务类型，选择语音引擎获取语音服务，而后把结果返回给 HTTP 服务器，再回复客户端。云存储负责存储服务器端的运行日志以及用户的一些个性化数据如个性化音库等。

语音云客户端工作在用户设备上，主要功能是收集用户数据、数据简单加工处理，构建解析通信消息和网络交互等。

1.2 名词和缩略语

☐ **TTS (Text to Speech)**

语音合成（Text To Speech，TTS）技术能够自动将任意文字实时转换为连续的自然语音，是一种能够在任何时间、任何地点，向任何人提供语音信息服务的高效便捷手段，非常符合信息时代海量数据、动态更新和个性化查询的需求。

- **IAT(iFly Auto Transform) & ASR (Automatic Speech Recognition)**

  语音听写和语音识别技术是一种使计算机能够识别人通过麦克风或者电话输入的词语或语句的技术，简单的说就是能够让计算机听懂人说话。它的最终目标是使得计算机不受词汇量限制，在各种噪声环境、语音信道下，能够实时、准确地识别不同方言、口音等特点的说话人的语句。

  让机器听懂人类的语音，这是人们长期以来梦寐以求的事情。语音识别是一门交叉学科，关系到多学科的研究领域，不同领域上的研究成果都对语音识别的发展作了贡献。语音识别技术就是让机器通过识别和理解过程把语音信号转变为相应的文本或命令的技术。

- **IVW**

  iFLY Voice Wakeup，讯飞语音唤醒。

- **NLP**

  Nature Languege Procession，自然语言理解。

- **MSP**

  iFLY Mobile Speech Platform，或称为 IMSP，讯飞移动语音平台的缩写，是讯飞面向移动互联网领域开发的语音服务平台。

- **MSSP**

  Mobile Speech Service Protocol，移动语音服务协议，基于 HTTP1.1 协议扩展的语音应用协议。

1.3 开发包组件

- **Windows 平台**

开发组件	组件组成	说明
头文件	qtts.h qisr.h qivw.h msp_cmn.h msp_errors.h msp_types.h	接口函数声明，如果不使用对应接口的功能，可以不包含对应的头文件。
动态引入库	msc.lib	包含接口函数引入
运行时刻库	msc.dll	接口函数具体实现
客户端配置文件	msc.cfg	运行库相关配置项

☐ **Linux 平台**

开发组件	组件组成	说明
头文件	qtts.h qisr.h qivw.h msp_cmn.h msp_errors.h msp_types.h	接口函数声明
运行时刻库	libmsc.so	接口函数具体实现
客户端配置文件	msc.cfg	运行库相关配置项

1.4 文档说明

本文针对的读者是具有 Windows、 Linux 或者嵌入式平台开发经验的 C/C++程序员。

文档中使用的符号约定：

符号	含义
[in]	表明该参数是调用时赋值的参数——输入参数
[out]	该参数在函数返回时被赋值——输出参数
[in/out]	该参数在函数调用时作为输入、函数返回时作为输出参数

第 2 章 通用开发接口

2.1 通用接口简介

2.1.1 通用接口函数列表

MSCV5+提供如下通用接口：

函数名称	功能简介
MSPLogin	登录接口
MSPUploadData	数据上传接口
MSPDownloadData	预留
MSPSetParam	参数设置接口、离线引擎初始化接口
MSPGetParam	预留
MSPSearch	数据搜索接口
MSPLogout	登录退出接口

2.1.2 返回值说明

对于开发接口，如果调用成功，返回值为 int 型的接口都会返回 MSP_SUCCESS，否则返回错误代码，错误代码参见 msp_errors.h；返回值为指针类型的接口，函数执行失败返回空指针 0，错误代码可以通过相关的回传参数查看，函数执行成功但无数据时返回空指针 0，有数据时返回非 0 指针。

2.2 函数调用

2.2.1 MSPLogin

□ **函数原型**

int MSPAPI MSPLogin(const char* usr, const char* pwd, const char* params)

□ **功能**

用户登录，每一项语音业务必须先调用 MSPLogin，如果登录失败则无法获取任何语音服务。

□ **参数**

♦ **usr [in]**

用户名，用户可以从网站 http://member.voicecloud.cn/index.php/default/register 申

请。

♦ **pwd [in]**

密码。

♦ **params [in]**

登录时传入的参数。

参数	名称	说明
appid	应用 ID	用来标识用户应用的 ID 号，为一字符串，由数字和英文字符组成。 用户在科大讯飞语音云开发者网站上申请成功开发 SDK 后会获取到 appid 值。
engine_start	离线引擎启动	启动离线引擎，支持参数， ivw：唤醒 asr：识别、听写 tts：合成
[xxx]_res_path	离线引擎资源路径	设置 ivw、asr、tts 引擎离线资源路径， 详细格式如下： fo|[path]|[offset]|[length]|xx|xx 示例如下： 单个资源路径： ivw_res_path=fo|res/ivw/wakeupresource.jet 多个资源路径： asr_res_path=fo|res/asr/common.jet;fo|res/asr/sms.jet

□ **返回值**

如果函数调用成功返回 MSP_SUCCESS，否则返回错误代码，错误代码参见 msp_errors.h，几个主要的返回值如下：

返回值	意义
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_DB_INVALID_USER	无效的用户名
MSP_ERROR_DB_INVALID_PWD	无效的密码

□ **说明**

本接口为通用登录接口。MSP 系统所有功能都需要用户在登录的状态下使用。本接口应当在应用程序中仅调用一次，多次调用本函数（中间没有调 MSPLogout）时只有第一次调用此函数会进行实际的登录操作，后面的调用会返回成功，但不会完成实际的

登录工作。同样 MSPLogout 也应当在应用程序退出时仅调用一次。

☐ 用法示例

```
const char*     usr = NULL;
const char*     pwd =NULL;
const char*     params=“appid = ********, engine_start = ivw,ivw_res_path
=fo|res/ivw/wakeupresource.jet”;
int   ret = MSPLogin( usr, pwd, params );
if( MSP_SUCCESS != ret )
{
      printf( “MSPLogin failed, error code is: %d”, ret );/* No speech serviec will be got if MSPLogin
failed */
}
```

☐ 参见

MSPLogout。

2.2.2 MSPUploadData

☐ 函数原型

const char* MSPAPI MSPUploadData(const char* dataName, void* data, unsigned int dataLen, const char* params, int* errorCode)

☐ 功能

用户数据上传。

☐ 参数

♦ **dataName [in]**

数据名称。

♦ **data [in]**

数据缓冲区起始地址。

♦ **dataLen [in]**

数据长度。

♦ **params [in]**

数据描述参数。

参数	名称	说明
sub	服务类型	请求服务类型，支持的类型有： uup，asr
dtt	数据类型	上传数据的数据类型

		1、uup 支持的类型有： a、contact：联系人列表 b、userword：用户词表 2、asr 支持的类型有： abnf ：识别语法类型

♦ **errorCode [out]**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h，几个主要的返回值如下：

返回值	意义
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_BUSY	忙碌

□ **说明**

无

□ **用法示例**

```
const char*      dataname = userword;
FILE*      data = fopen("resource\userwords.txt ", "rb");
const char*      params= " subject =uup, data_type =userword";
int dataLen = 0;
fseek(data, 0, SEEK_END);
dataLen = ftell(data);
fseek(data, 0, SEEK_SET);
MSPUploadData( dataName, data, dataLen, params, &errorcode);
if( MSP_SUCCESS != errorcode )
{
      printf( “MSPUploadData failed, error code is: %d”, ret );
 }
```

□ **参见**

无

2.2.3 MSPDownloadData

□ **函数原型**

const void* MSPAPI MSPDownloadData(const char* params, unsigned int* dataLen, int* errorCode)

□ **功能**

预留

2.2.4 MSPSetParam

□ 函数原型

int MSPAPI MSPSetParam(const char* paramName, const char* paramValue)

□ 功能

参数设置接口、离线引擎初始化接口。

□ 参数

♦ **paramName [in]**

参数名。

可设置参数：

参数	名称	说明
engine_start	启动离线引擎	启动引擎：设置 paramName 为 engine_start，paramValue 为 engine_start=asr、tts、ivw 等离线业务后加资源所在路径,例如 engine_start=asr;tts,asr_res_path=fo|file_path|[offset]|[length]
engine_destroy	销毁离线引擎	销毁引擎： 设置 paramName 为 engine_destroy，paramValue 为 asr、tts、ivw（如 engine_destroy=tts）等离线业务

♦ **paramValue [in]**

参数值。

□ 返回值

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h，几个主要的返回值如下：

返回值	意义
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_BUSY	忙碌

□ 说明

无

□ 用法示例

```
const char*     paramsName= " engine_start ";
const char*     paramsValue=
"engine_start=tts,voice_name=xiaoyan,tts_res_path=fo|res\\tts\\dict.mp3;fo|res\\tts\\xiaoyan.mp3
";
```

```
errorcode = MSPSetParam (paramsName, paramsValue);
if( MSP_SUCCESS != errorcode )
{
      printf( “MSPSetParam failed, error code is: %d”, ret );
 }
```

□ 参见

无

2.2.5 MSPGetParam

□ 函数原型

int MSPAPI MSPGetParam(const char *paramName, char *paramValue, unsigned int *valueLen)

□ 功能

预留

2.2.6 MSPSearch

□ 函数原型

const void* MSPAPI MSPSearch(const char* params, const char* text, unsigned int* contentLen, int* errorCode)

□ 功能

语义。

□ 参数

♦ **params [in]**

可以设置的参数：

参数	名称	说明
sch	语义服务	0：不使用语义，默认 1：使用语义

♦ **text [in]**

上传文本

♦ **contentLen [in]**

上传文本长度

♦ **errorCode [out]**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h，几个主要的返回值如下：

返回值	意义
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_BUSY	忙碌

□ **说明**

无

□ **用法示例**

```
const char*    text = “附近的美食” ;
int textLen    = 0 ;
MSPSearch("wap_proxy=wifi,extid=null,aue=speex-wb,ssm=1,auf=audio/L16;rate=16000,ent=sms16k,sch=1,vaver=1.1,vascn=app,valong=117.1484463,valat=31.8322467,vapos=中国安徽省合肥市望江西路,date=2012-02-09,time=10:12:22", text, textLen, &errorcode);
if( MSP_SUCCESS != errorcode )
{
      printf( “MSPSearch failed, error code is: %d”, ret );
 }
```

□ **参见**

无

2.2.7 MSPLogout

□ **函数原型**

int MSPAPI MSPLogout()

□ **功能**

退出登录。

□ **参数**

无。

□ **返回值**

如果函数调用成功返回 MSP_SUCCESS，否则返回错误代码，错误代码参见 msp_errors.h。

□ **说明**

本接口对应于 MSPLogin 接口，在使用通用接口相关功能时最后一个被调用，用来对 MSC 的通用接口部分进行逆初始化。没有调用 MSPLogin 直接调用本接口，则不会有任何效果，调用了 MSPLogin 不调用本接口而直接结束线程，则 MSC 会产生资源泄漏。

☐ 用法示例

```
int   ret = MSPLogout( );
if( MSP_SUCCESS != ret )
{
      printf( “MSPLogout failed, error code is: %d”, ret );
}
```

☐ 参见

MSPLogin。

第 3 章 合成开发接口说明

3.1 QTTS 接口简介

3.1.1 QTTS 接口函数列表

MSCV5+关于 TTS 提供如下函数调用：

函数名称	功能简介
QTTSSessionBegin	开始合成
QTTSTextPut	输入要合成的文本
QTTSAudioGet	获取合成音频
QTTSAudioInfo	获取音频相关信息
QTTSGetParam	获取合成业务相关参数
QTTSSetParam	设置合成业务参数
QTTSSessionEnd	结束合成

3.1.2 返回值说明

对于开发接口，如果调用成功，返回值为 int 型的接口都会返回 MSP_SUCCESS，否则返回错误代码，错误代码参见 msp_errors.h；返回值为指针类型的接口，函数执行失败返回空指针 0，错误代码可以通过相关的回传参数查看，函数执行成功但无数据时返回空指针 0，有数据时返回非 0 指针。

3.2 函数调用

3.2.1 QTTSSessionBegin

□ **函数原型**

const char* MSPAPI QTTSSessionBegin(const char* params, int* errorCode)

□ **功能**

MSPLogin 成功后开始合成，并在参数中指明本次合成用到的参数。

□ **参数**

♦ **params [in]**

参数	名称	说明
engine_type	引擎类型	使用引擎类型，支持参数， cloud：在线引擎

		local：离线引擎
voice_name	发音人	不同的发音人代表了不同的音色，如男声、女声、童声等，详细请参照《发音人列表》
speed	语速	合成音频对应的语速， 取值范围：0~100，数值越大语速越快。 默认值：50
volume	音量	合成音频的音量， 取值范围：0~100，数值越大音量越大。 默认值：50
pitch	语调	合成音频的音调， 取值范围：0~100，数值越大音调越高。 默认值：50
tts_res_path	合成资源路径 （仅离线引擎支持）	合成资源所在路径，支持 fo 方式参数设置，对应格式如下： fo|[file_info]|[offset]|[length] （1）若是合并资源，则只需传入一个资源路径，如： fo| combined.jet|0|1024 （2）若是分离资源，则需传两个资源路径，如： fo|common.jet|0|1024; fo| xiaoyan.jet|0|1024
rdn	数字发音	合成音频数字发音，支持参数， 0 数值优先,1 完全数值,2 完全字符串,3 字符串优先
rcn	1 的中文发音 （仅离线引擎支持）	支持参数： 0：表示发音为 yao 1：表示发音为 yi
text_encoding	文本编码 （必传）	合成文本编码格式，支持参数， GB2312 GBK BIG5 UNICODE GB18030 UTF8
sample_rate	采样率	合成音频采样率，支持参数， 16000 8000
background_sound	背景音 （仅在线引擎支持）	合成音频中的背景音，支持参数， 0：无背景音乐 1：有背景音乐
aue	音频编码	合成音频发往客户端时服务器采用的编

	（仅在线引擎支持）	解码算法和压缩等级，格式为：压缩算法;压缩等级，如 aue=speex-wb;7 表示使用 speex-wb 算法的等级 7 对传输中的合成音频进行编解码。 编码算法： raw speex speex-wb 编码等级： raw：无等级。 speex 系列：0-10； 1、aue=raw 是对合成音频不进行压缩；16BitPCM 格式的音频可以使用 speex 系列压缩算法进行压缩，如 aue=speex-wb;7，服务器会选择 speex-wb 算法使用压缩等级 7 对合成音频进行压缩后再传给 MSC，MSC 会解压缩还原成 PCM 格式的音频后返回给用户。 2、对于等级为负的情况，客户端不会将来自服务器的合成音频进行解压缩，而是将压缩音频直接返回给用户，如用户选择 aue=speech-wb;7，用户得到的将是 speech-wb 等级 7 的压缩音频。
ttp	文本类型 （仅在线引擎支持）	合成文本类型，支持参数， text: 普通格式文本 ssml：ssml 格式文本
speed_increase	语速增强 （仅离线引擎支持）	通过设置此参数控制合成音频语速基数，取值范围， 1：正常 2：2 倍语速 4：4 倍语速
effect	合成音效 （仅离线引擎支持）	合成音频的音效，取值范围， 0 /* 无音效 */ 1 /* 忽远忽近 */ 2 /* 回声 */ 3 /* 机器人 */ 4 /* 合唱 */ 5 /* 水下 */ 6 /* 混响 */ 7 /* 阴阳怪气 */

♦ **errorCode [out]**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h，几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_CREATE_HANDLE	创建合成实例失败

□ **返回值**

MSC 为本次合成建立的 ID，用来唯一的标识本次合成，供以后调用其他函数时使用。函数调用失败则会返回 NULL。

□ **说明**

此处设定的参数在本次合成中一直有效。此函数需要和接口 QTTSSessionEnd 配对使用，在这两个接口之间可以调用实际完成合成功能的函数如写入合成文本和取音频数据等。没有调用此函数则后续的函数调用会因为没有合法的 ID 而无法完成对应的功能。

□ **用法示例**

```
const char*     params= “engine_type = local, voice_name=xiaoyan, tts_res_path =
fo|res\\tts\\xiaoyan.jet;fo|res\\tts\\common.jet, sample_rate = 16000” ;
int             ret = 0;
const char*     session_id = QTTSSessionBegin( params, &ret );
if( 0 != ret )
{
    printf( “QTTSSessionBegin failed, error code is: %d”, ret );
}
```

□ **参见**

QTTSSessionEnd。

3.2.2 QTTSTextPut

□ **函数原型**

int MSPAPI QTTSTextPut(const char* sessionID, const char* textString, unsigned int textLen, const char* params)

□ **功能**

写入要合成的文本。

□ **参数**

♦ **sessionID [in]**

由 QTTSSessionBegin 返回过来的 ID。

♦ **textString [in]**

将要进行合成的文本。

♦ **textLen [in]**

合成文本长度，单位字节。

♦ **params [in]**

本次合成所用的参数，只对本次合成的文本有效。

返回值

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_INVALID_PARA_VALUE	无效的参数值
MSP_ERROR_NO_ENOUGH_BUFFER	发送的文本长度超过最大允许长度

□ **说明**

本接口不支持连续被调用。调用本接口写入合成文本后，用户需要反复调用 QTTSAudioGet 接口来获取音频。

□ **用法示例**

```
const char*     text_str = “科大讯飞是亚太地区最大的语音上市公司”;
unsigned int    text_len = strlen( textString );          //textLen 参数为合成文本所占字节数
int ret = QTTSTextPut( session_id, text_str, text_len, NULL );
if( 0 != ret )
{
printf( “QTTSTextPut failed, error code is: %d”, ret );
}
```

□ **参见**

QTTSAudioGet。

3.2.3 QTTSAudioGet

□ **函数原型**

const void* MSPAPI QTTSAudioGet(const char* sessionID, unsigned int* audioLen, int* synthStatus, int* errorCode)

□ **功能**

获取合成音频。

□ **参数**

♦ **sessionID [in]**

由 QTTSSessionBegin 返回过来的 ID。

♦ **audioLen [out]**

合成音频长度，单位字节。

♦ **synthStatus [out]**

合成音频状态，可能的值如下：

枚举常量	简介
MSP_TTS_FLAG_STILL_HAVE_DATA = 1	音频还没取完，还有后继的音频
MSP_TTS_FLAG_DATA_END = 2	音频已经取完

♦ **errorCode [out]**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h，几个主要的值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_NO_MORE_DATA	没有更多的数据（音频已经取完）

□ **返回值**

如果函数调用失败返回 0 指针，调用成功且有音频数据时返回非 0 指针，无音频数据时返回 0 指针。

□ **说明**

用户需要反复调用此接口，在 errorCode 返回值为 0 的情况下直到 synthStatus 返回 MSP_TTS_FLAG_DATA_END（音频已经取完）为止。

在使用中，如果函数调用成功但没有获取到音频数据，用户需要将当前线程 sleep 一段时间后再次调用，以防止频繁调用浪费 CPU 资源。

□ **用法示例**

```
f_pcm = fopen("tts.pcm", "wb");
while (1) {
        const void *data = QTTSAudioGet(sessionID, &audioLen, &status, &errorCode);
        if (NULL != data)
        fwrite(data, audioLen, 1, f_pcm);
       if (status == 2 || errorCode != 0)
```

```
            break;
        }
        fclose(f_pcm);
```

□ 参见

无。

3.2.4 QTTSAudioInfo

□ 函数原型

const char* MSPAPI QTTSAudioInfo(const char* sessionID, int* errorCode)

□ 功能

获取关于合成音频的某些信息如合成音频对应的当前文本位置等。

□ 参数

♦ **sessionID [in]**

由 QTTSSessionBegin 返回过来的 ID。

♦ **errorCode [out]**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h，几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID

□ 返回值

如果函数调用成功返回字符串指针，否则返回 NULL。

□ 说明

本接口用在调用 QTTSAudioGet 后，获取关于此段音频的描述信息，目前支持的信息有：此次合成音频对应文本结束位置，用户可以利用此信息来做实时高亮度显示合成的文本等应用，格式为”ced=xxx”。

□ 用法示例

```
int             ret = 0;
const char*     audio_info = QTTSAudioInfo( session_id, &ret );
if( 0 != ret )
{
    printf( “QTTSAudioInfo failed, error code is: %d”, ret );
}
```

□ 参见

QTTSSAudioGet。

3.2.5 QTTSGetParam

□ 函数原型

int MSPAPI QTTSGetParam(const char* sessionID, const char* paramName, char* paramValue, unsigned int* valueLen)

□ 功能

查询 MSC 记录下来的一些信息如上行流量和下行流量等。

□ 参数

♦ **sessionID [in]**

由 QTTSSessionBegin 返回过来的 ID。

♦ **paramName [in]**

要获取的参数名称，MSCV5+支持的参数如下：

参数名称	意义
upflow	上行数据量。
downflow	下行数据量。

♦ **paraValue [out]**

获取到的参数值，以字符串形式返回，不支持的参数将返回空的值。

♦ **valueLen [in/out]**

输入是存放参数值字符串 paraValue 缓冲区的长度，输出为参数值字符串的长度。

□ 返回值

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数

□ 说明

调用本接口时，以获取上行流量为例，获取到的是本次合成到此时为止产生的上行流量。下行流量与此相似。

□ 用法示例

```
const char*     para_name = “upflow”;
char*           para_value[ 32 ] = “”;
unsigned int    value_len = 32;
int   ret = QTTSGetParam ( session_id, para_name, para_value, &value_len );
```

```
if( 0 != ret )
{
    printf( “QTTSGetParam failed, error code is: %d”, ret );
}
```

□ 参见

无。

3.2.6 QTTSSetParam

□ 函数原型

int MSPAPI QTTSSetParam(const char *sessionID, const char *paramName, const char *paramValue);

□ 功能

设置合成参数。

□ 参数

♦ **sessionID [in]**

由 QTTSSessionBegin 返回过来的 ID。

♦ **paramName [in]**

要设置的参数名称；支持的参数如下：

参数名称	意义
sessioninfo	添加信息到结构化日志。paraValue 格式为 json： {"xxx":"xxx"}

♦ **paraValue [in]**

设置的参数值。

□ 返回值

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数

□ 说明

无。

□ 用法示例

```
const char*     para_name = “sessioninfo”;
char*           para_value[ 32 ] = “{\”app_text_length\”:\”1234\”}”;
int   ret = QTTSSetParam ( session_id, para_name, para_value);
if( 0 != ret )
{
    printf( “QTTSGetParam failed, error code is: %d”, ret );
}
```

□ **参见**

无。

3.2.7 QTTSSessionEnd

□ **函数原型**

int MSPAPI QTTSSessionEnd(const char* sessionID, const char* hints)

□ **功能**

结束合成。

□ **参数**

♦ **sessionID [in]**

由 QTTSSessionBegin 返回过来的 ID。

♦ **hints [in]**

结束本次合成的原因描述，用于记录日志，便于用户查阅或者跟踪某些问题。

□ **返回值**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID

□ **说明**

本接口对应于 QTTSSessionBegin 接口，用来结束本次合成。调用本函数后，关于当前的所有资源（参数，合成文本，实例等）都会被释放，所以用户不应该再针对该实例做任何操作（比如使用其 ID 等）。

□ **用法示例**

```
int   ret = QTTSSessionEnd ( session_id, “normal end” );
if( 0 != ret )
{
    printf( “QTTSSessionEnd failed, error code is: %d”, ret );
```

```
}
```

□ 参见

QTTSSessionBegin

第 4 章 识别开发接口说明

4.1 QISR 接口简介

4.1.1 QISR 接口函数列表

MSCV5+关于 ISR 提供如下函数调用：

函数名称	功能简介
QISRSessionBegin	开始识别
QISRAudioWrite	写入需要识别的语音
QISRGetResult	获取识别结果
QISRGetParam	获取识别/语音听写业务相关参数
QISRSessionEnd	结束识别
QISRBuildGrammar	构建语法
QISRUpdateLexicon	更新词典

本组接口不是线程安全的，用户需要自行保证某些数据的线程安全性，如 sessionID。

4.1.2 返回值说明

对于开发接口，如果调用成功，返回值为 int 型的接口都会返回 MSP_SUCCESS，否则返回错误代码，错误代码参见 msp_errors.h；返回值为指针类型的接口，函数执行失败返回空指针 0，错误代码可以通过相关的回传参数查看，函数执行成功但无数据时返回空指针 0，有数据时返回非 0 指针。

4.2 函数调用

4.2.1 QISRSessionBegin

□ **函数原型**

const char* MSPAPI QISRSessionBegin(const char* grammarList, const char* params, int* errorCode)

□ **功能**

本接口用来开始识别，并在参数中指定本次识别用到的语法列表，本次识别所用的参数等。

□ **参数**

♦ **grammarList [in]**

【在线引擎】uri-list 格式的语法， 可以是一个语法文件的 URL 或者一个引擎内置语法列表。可以同时指定多个语法，不同的语法之间以“,”隔开。进行语音听写时不需要语法，此参数设定为 NULL 或空串即可；进行语音识别时则需要语法，语法在此参数中指定。

【离线引擎】设定为 NULL 即可。

♦ **params [in]**

参数	名称	说明
engine_type	引擎类型	可取值： cloud：在线引擎 local：离线引擎 mixed：混合引擎 默认值： cloud
domain	应用领域 （仅在线引擎支持）	可取值： iat：普通文本转写； search：热词搜索； video：视频搜索； poi：地名搜索； music：音乐搜索； 默认值：iat
language	语言	可取值： zh_cn：简体中文 zh_tw：繁体中文 en_us：英文 默认值：zh_cn
accent	语言区域	可取值： mandarin：普通话 cantonese：粤语 lmz：四川话 henanese：河南话 dongbeiese：东北话 默认值：mandarin
sample_rate	音频采样率	可取值： 16000 8000 默认值：16000

asr_ptt	标点符号标识	设置是否有标点符号，设置为 true 有标点符号，设置为 false 无标点符号，默认无标点符号
asr_sch	语义标识 （仅在线引擎支持）	是否转语义引擎，设置为 1 转语义引擎，默认不转
aue	音频编码 （仅在线引擎支持）	本参数指明客户端和服务器之间交互的音频数据的压缩算法和编码等级。 格式：算法;等级，如 aue=speex-wb;7。 默认值：speex-wb;7 编码算法： raw speex speex-wb 编码等级： raw：可以取-1。 speex 系列：-1-10；
auf	音频格式 （仅在线引擎支持）	可取值： audio/L16;rate=16000 audio/L16;rate=8000 默认值：audio/L16;rate=16000
asr_threshold	识别门限 （仅离线引擎支持）	离线语法识别结果门限值，设置只返回置信度得分大于此门限值的结果 可取值：0~100 默认值：0
asr_denoise	是否开启降噪功能 （仅离线引擎支持）	是否开启离线引擎降噪功能。 可取值： 0：不开启 1：开启 默认不开启
asr_res_path	离线识别资源路径	离线识别资源所在路径，对应格式如下： access_type1|file_info1|[offset1]|[length1];access_type2|file_info2|[offset2]|[length2] 各字段含义如下： access_type：文件访问方式，支持路径方式（fo）和文件描述符方式（fd）； file_info：此字段和 access_type 对应，文件路径对应 fo，文件描述符对应 fd，

		其中文件路径必须是包含文件名的完整路径； offset：资源文件在此传入文件中的偏移； length：资源文件大小。
grm_build_path	离线语法生成路径	构建离线语法所生成数据的保存路径（文件夹）
resut_type	结果格式	可取值：plain，json，xml 默认值：plain
text_encoding	文本编码格式	表示参数中携带的文本编码格式
result_encoding	结果编码格式	可取值： gb2312 utf-8 unicode 不同的结果格式支持的结果编码格式不尽相同： xml，plain：gb2312，utf-8，unicode json：utf8
vad_enable	VAD 功能开关	是否启用 VAD 处理
vad_timeout	头部静音最大长度	最大音频长度。如果静音长度超过了此值，则认为用户此次无有效音频输入。此参数仅在打开 VAD 功能时。 可取值：0~30000ms 默认值：10000ms
vad_speech_tail	尾部静音最大长度	尾部静音长度。如果尾部静音长度超过了此值，则认为用户音频已经结束，此参数仅在打开 VAD 功能时有效。 可取值：0~30000ms 默认值：2000ms
grammartype	语法类型 （仅在线引擎支持）	传入所用语法的语法类型 可取值： uri-list abnf 默认值： uri-list
local_grammar	离线语法 id	构建离线语法后获得的语法 ID
cloud_grammar	在线语法 id	构建在线语法后获得的语法 ID

mixed_type	混合引擎策略类型	可取值： realtime：实时，同时使用在线引擎和离线引擎，在在线引擎结果超时的情况下，使用离线引擎结果； delay：延时，先使用在线引擎，当在线引擎结果超时后自动转向离线引擎。 默认值：realtime
mixed_timeout	在线引擎结果超时时间	使用混合引擎情况下，在线引擎结果超时时间。 默认值：2000，单位：ms
mixed_threshold	混合门限	混合策略为 realtime 时使用，当离线引擎给出识别结果大于此门限值时直接给出离线结果，否则等待在线结果，若在线结果超时则给出离线结果。 可取值：0~100 默认值：60

♦ **errorCode [out]**

如果函数调用成功则其值为 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_CREATE_HANDLE	创建实例失败

□ **返回值**

MSC 为本次识别建立的 ID，用来唯一的标识本次识别，供以后调用其他函数时使用。函数调用失败则会返回 NULL。

□ **说明**

此处设定的参数在本次识别中一直有效。此函数需要和接口 QISRSessionEnd 配对使用，在这两个接口之间可以调用实际完成识别功能的函数如写入音频、获取结果等。没有调用此函数则 MSC 不会建立和当前线程有关的实例，后续的函数调用会因为没有合法的 ID 而无法完成对应的功能。

□ **用法示例**

```
/* vad_timeout 和 vad_speech_tail 两个参数只有在打开 VAD 功能时才生效 */
const char* params = “sub=iat,aue=speex-wb;7,auf=audio/L16;rate=16000,ent=sms16k,
```

```
rst=plain,vad_timeout=1000,vad_speech_tail=1000";
int   ret = 0;
const char*     session_id = QISRSessionBegin( NULL, params, &ret );
if( 0 != ret )
{
      printf( "QISRSessionBegin failed, error code is: %d", ret );
}
```

□ 参见

QISRSessionEnd。

4.2.2 QISRAudioWrite

□ 函数原型

int MSPAPI QISRAudioWrite(const char* sessionID, const void* waveData, unsigned int waveLen, int audioStatus, int *epStatus, int *rsltStatus)

□ 功能

写入本次需要识别的音频，本接口需要反复调用直到音频写完为止。

□ 参数

♦ **sessionID [in]**

由 QISRSessionBegin 返回过来的 ID。

♦ **waveData [in]**

音频数据缓冲区起始地址。

♦ **waveLen [in]**

音频数据长度，单位字节，其大小不能超过设定的 max_audio_size。

♦ **audioStatus [in]**

用来指明用户本次识别的音频是否发送完毕，可能的值如下：

枚举常量	简介
MSP_AUDIO_SAMPLE_FIRST = 1	第一块音频
MSP_AUDIO_SAMPLE_CONTINUE = 2	还有后继音频
MSP_AUDIO_SAMPLE_LAST = 4	最后一块音频

在 MSCV5+中 MSP_AUDIO_SAMPLE_LAST（0x04）用来指明当前的音频已经发送完毕，除此之外的任何值都将被 MSC 视为还有后继的音频。

♦ **epStatus [out]**

端点检测（End-point detected）器所处的状态，可能的值如下：

枚举常量	简介
MSP_EP_LOOKING_FOR_SPEECH = 0	还没有检测到音频的前端点。
MSP_EP_IN_SPEECH = 1	已经检测到了音频前端点，正在进行正常的音频处理。
MSP_EP_AFTER_SPEECH = 3	检测到音频的后端点，后继的音频会被MSC忽略。
MSP_EP_TIMEOUT = 4	超时。
MSP_EP_ERROR = 5	出现错误。
MSP_EP_MAX_SPEECH = 6	音频过大。

当 epStatus 大于等于 3 时，用户应当停止写入音频的操作，否则写入 MSC 的音频会被忽略。

♦ **rsltStatus [out]**

识别器所处的状态，可能的值如下：

枚举常量	简介
MSP_REC_STATUS_SUCCESS = 0	识别成功，此时用户可以调用QISRGetResult来获取（部分）结果。
MSP_REC_STATUS_NO_MATCH = 1	识别结束，没有识别结果。
MSP_REC_STATUS_INCOMPLETE = 2	正在识别中。
MSP_REC_STATUS_COMPLETE = 5	识别结束。

□ **返回值**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_NULL_HANDLE	空句柄
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_INVALID_PARA_VALUE	无效的参数值
MSP_ERROR_NO_ENOUGH_BUFFER	缓冲区溢出

□ **说明**

MSC 处理音频的策略是边接收、边压缩（如果音频编码格式不为 raw）、边发送。由于音频压缩速度和网络速度的限制，如果音频写入太快太急（如 20 倍于音频码率），可能会造成原始音频或压缩音频在 MSC 中积累过多，从而造成缓冲区无法再容纳更多的数据而产生 MSP_ERROR_NO_ENOUGH_BUFFER（10117，缓冲区溢出）的错误。

调用本接口时，推荐用户在写入音频时采取“边录边发”的方式，即每隔一小段时间将采集到的音频通过本接口写入 MSC。这种“边录边发”的方式可以加快结果返回的

速度：发送靠后的音频时，前面的音频或许已经被服务器处理过并将部分结果返回了。调用接口时请设置好 audioStatus 的值，并检查返回型参数 epStatus 的值，以便及时了解音频的前后端点等信息。如果当前写入的不是最后一块音频，需要将 audioStatus 的值设为 2（ISR_AUDIO_SAMPLE_CONTINUE）。如果在音频写入过程中检测到 epStatus 的值为 3（ISR_EP_AFTER_SPEECH），说明系统已经检测到音频的后端点，则应该立即结束写入音频；如果用户要在检测到音频后端点之前结束音频的写入，需要将最后一块音频数据的 audioStatus 设为 4（ISR_AUDIO_SAMPLE_LAST）。

如果识别状态 recogStatus 值为 0（ISR_REC_STATUS_SUCCESS）表示已经有部分或全部识别结果缓存在 MSC 中了，用户可以调用 QISRGetResult 获取这部分结果，再继续调用 QISRAudioWrite 以写入后续的音频数据，这种两个接口混调的方式，可以很快的获得（部分）识别结果，特别是使用较大音频进行语音听写时。

□ 用法示例

```
char            audio_data[ 5120 ] = “”;
unsigned int    audio_len = 0;
int             audio_status = 2;
int             ep_status = 0;
int             rec_status = 0;
int             ret = 0;
while(MSP_AUDIO_SAMPLE_LAST != audio_status )
{
    …   // 读取音频到缓冲区 audio_data 中，设置音频长度 audio_len，音频状态 audio_status。
ret = QISRAudioWrite( session_id, audio_data, audio_len, audio_status
, &ep_status, &rec_status );
if( 0 ! = ret )
{
    printf( “QISRAudioWrite failed, error code is: %d”, ret );
    break;
}
else if(MSP_EP_AFTER_SPEECH == ep_status )  /* 检测到音频后端点，停止写入音频 */
{
    printf( “end point of speech has been detected!” );
    break;
}

/* 如果是实时采集音频，可以省略此操作。5KB 大小的 16KPCM 持续的时间是 160 毫秒 */
Sleep( 160 );
}
```

□ 参见

QISRGetResult。

4.2.3 QISRGetResult

□ **函数原型**

const char * MSPAPI QISRGetResult(const char* sessionID, int* rsltStatus, int waitTime, int *errorCode)

□ **功能**

获取识别结果。

□ **参数**

♦ **sessionID [in]**

由 QISRSessionBegin 返回过来的 ID。

♦ **rsltStatus [out]**

识别结果的状态，其取值范围和含义请参考 QISRAudioWrite 的参数 recogStatus。

♦ **waitTime [in]**

与服务器交互的间隔时间，MSCV5+中，此参数做保留用。

♦ **errorCode [out]**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_NO_DATA	没有数据（如没有写入识别所用的音频等）

□ **返回值**

函数执行失败返回 NULL。函数执行成功并且获取到识别结果时返回识别结果，函数执行成功没有获取到识别结果时返回 NULL。

□ **说明**

本接口一般是在音频写入完毕后调用，以获取语音识别结果。实际应用中，为了及时获取（部分）结果，本接口也可以在写入音频过程中调用，写入音频过程中的调用方法请参考 QISRAudioWrite 接口的说明部分。在音频写入完毕后，用户需要反复调用此接口，直到识别结果获取完毕（rlstStatus 值为 5）或返回错误码。使用此接口时请注意，如果某次成功调用后没有获得识别结果，请将当前线程 sleep 一段时间后再次调用，以防止频繁调用浪费 CPU 资源。

□ 用法示例

```
char            rslt_str[ 2048 ] = “”;
const char*     rec_result = NULL;
int             rslt_status = 0;
int             ret = 0;
while(MSP_REC_STATUS_SPEECH_COMPLETE != rslt_status )
{
rec_result = QISRGetResult ( session_id, &rslt_status, 5000, &ret );
if( 0 != ret )
{
      printf( “QISRGetResult failed, error code is: %d”, ret );
      break;
}

if( NULL != rec_result )
{
/* 用户可以用其他的方式保存识别结果 */
strcat( rslt_str, rec_result );
continue;
}

/* sleep 一下很有必要，防止 MSC 端无缓存的识别结果时浪费 CPU 资源 */
Sleep( 200 );
}
```

□ 参见

QISRAudioWrite。

4.2.4 QISRGetParam

□ 函数原型

int MSPAPI QISRGetParam(const char* sessionID, const char* paramName, char* paramValue, unsigned int* valueLen)

□ 功能

查询 MSC 记录下来的一些信息如上行流量和下行流量，输入语音的音量等。

□ 参数

♦ **sessionID [in]**

由 QISRSessionBegin 返回过来的 ID。

♦ **paramName [in]**

要获取的参数名称，MSCV5+支持的参数如下：

参数名称	意义
upflow	上行数据量。
downflow	下行数据量。
volume	最近一次写入的音频的音量。

♦ **paraValue [out]**

获取的参数值，以字符串形式返回，不支持的参数将返回空的值。

♦ **valueLen [out]**

参数值的长度。

□ **返回值**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数

□ **说明**

流量信息的获取方法和含义请参考 3.2.6 节 QTTSGetParam 接口的说明部分，由于音量的获取只能在一次识别中进行，所以使用本接口获取音量信息时参数 sessionID 不能为 NULL。

□ **用法示例**

参见 3.2.5 节 QTTSGetParam。

□ **参见**

QTTSGetParam。

4.2.5 QISRSessionEnd

□ **函数原型**

int MSPAPI QISRSessionEnd(const char* sessionID, const char* hints)

□ **功能**

结束识别。

□ **参数**

♦ **sessionID [in]**

由 QISRSessionBegin 返回过来的 ID。

♦ **hints [in]**

结束本次识别的原因描述，用于记录日志，便于用户查阅或者跟踪某些问题。

☐ 返回值

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h 和 msp_error.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID

☐ 说明

本接口需要和 QISRSessionBegin 配合使用，用来结束本次识别。

调用本函数后，关于本次识别的所有资源（参数，语法，音频，实例等）都会被释放，所以用户不应该再针对该实例做任何操作（比如使用其 SessionID 等）。

☐ 用法示例

```
int   ret = QISRSessionEnd ( session_id, “normal end” );
if( 0 != ret )
{
      printf( “QISRSessionEnd failed, error code is: %d”, ret );
}
```

☐ 参见

QISRSessionBegin

4.2.6 QISRBuildGrammar

☐ 函数原型

int MSPAPI QISRBuildGrammar (const char* grammarType, const char* grammarContent, int grammarLength, const char* params, GrammarCallBack callback, void* userData);

☐ 功能

构建语法，生成语法 ID。

☐ 参数

♦ **grammarType [in]**

语法类型，在线识别采用 abnf 格式语法，离线识别采用 bnf 格式语法。

♦ **grammarContent [in]**

语法内容。

♦ **grammarLength [in]**

语法文件长度。

♦ **params [in]**

参数	名称	说明
engine_type	引擎类型	可取值： cloud：在线引擎 local：离线引擎 mixed：混合引擎 默认值： cloud
sample_rate	音频采样率	可取值： 16000 8000 默认值：16000
asr_res_path	离线识别资源路径	离线识别资源所在路径，对应格式如下： access_type1|file_info1|[offset1]|[length1];access_type2|file_info2|[offset2]|[length2] 各字段含义如下： access_type：文件访问方式，支持路径方式（fo）和文件描述符方式（fd）； file_info:此字段和access_type对应，文件路径对应fo，文件描述符对应fd，其中文件路径必须是包含文件名的完整路径； offset：资源文件在此传入文件中的偏移； length：资源文件大小。
grm_build_path	离线语法生成路径	构建离线语法所生成数据的保存路径（文件夹）

♦ callback[in]

构建语法回调接口。

typedef int (*GrammarCallBack)(int errorCode, const char* info, void* userData);

- errorCode [out]

  成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。

- info [out]

  成功返回语法 ID，失败返回错误信息。

- userData[out]

用户数据。

♦ **userData[in]**

用户数据。

□ **返回值**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数

□ **说明**

无。

□ **用法示例**

□ **参见**

4.2.7 QISRUpdateLexicon

□ **函数原型**

```
int MSPAPI QISRUpdateLexicon (const char* lexiconName, const char*
lexiconContent,int lexiconLength,const char* params,LexiconCallBack callback
,void* userData);
```

□ **功能**

更新词典，包括本地语法词典、云端联系人、云端个性化词表。

□ **参数**

♦ **lexiconName [in]**

词典名称。

更新本地语法词典：传递语法中需要更新的词典槽名称

♦ **lexiconContent [in]**

词典内容。

本地语法词典：词典内容为换行符分割的字符串列表,如：

”词条 1\n 词条 2\n 词条 3\n 词条 4”

联系人：词典内容为换行符分割的字符串列表,如：

”联系人 1\n 联系人 2\n 联系人 3\n 联系人 4”

个性化词表：json 格式，格式

```
{"userword":[
        {"name":"词组 A","words":["词条 1","词条 2"]},
        {"name":"词组 B","words":["词条 1","词条 2"]}
]}
```

♦ **lexiconLength [in]**

词典内容长度。

♦ **params [in]**

参数	名称	说明
engine_type	引擎类型	可取值： cloud：在线引擎 local：离线引擎 mixed：混合引擎 默认值： cloud
subject	业务类型	必须由用户指定，更新云端词典设置为 uup，离线引擎不支持此参数设置
data_type	数据类型	必须由用户指定。 更新云端联系人：contact 更新云端个性化词表：userword 离线引擎不支持此参数设置
sample_rate	音频采样率	可取值： 16000 8000 默认值：16000
asr_res_path	离线识别资源路径	离线识别资源所在路径，对应格式如下： access_type1|file_info1|[offset1]|[length1];access_type2|file_info2|[offset2]|[length2] 各字段含义如下： access_type：文件访问方式，支持路径方式（fo）和文件描述符方式（fd）； file_info:此字段和access_type 对应，文件路径对应 fo，文件描述符对应 fd，其中文件路径必须是包含文件名的完整路径；

		offset：资源文件在此传入文件中的偏移； length：资源文件大小。
grm_build_path	离线语法生成路径	构建离线语法所生成数据的保存路径（文件夹）
text_encoding	文本编码格式	表示参数中携带的文本编码格式
grammar_list	语法 id 列表 （仅离线引擎支持）	指定需要更新的离线语法 id 列表，支持一次性更新多个语法，格式如下： id1;id2

♦ **callback[in]**

更新词典回调接口。

```
typedef int ( *LexiconCallBack)( int errorCode, const char* info, void* userData);
```

- errorCode [out]

  成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。

- info [out]

  成功：离线返回更新成功的语法 ID 列表，在线无返回信息。

  失败：返回错误信息。

- userData[out]

  用户数据。

♦ **userData[in]**

用户数据。

□ **返回值**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数

□ **说明**

无。

□ **用法示例**

□ **参见**

无。

第 5 章 唤醒开发接口说明

5.1 QIVW 接口简介

5.1.1 QIVW 接口函数列表

MSCV5+关于 IVW 提供如下函数调用：

函数名称	功能简介
QIVWSessionBegin	开始唤醒功能
QIVWRegisterNotify	注册回调函数
QIVWAudioWrite	写入用来唤醒（或唤醒+识别）的语音
QIVWSessionEnd	结束唤醒功能

说明：本组接口提供单独唤醒和唤醒+识别功能，具体功能使用通过 QIVWSessionBegin 参数指定。本组接口不是线程安全的，用户需要自行保证某些数据的线程安全性。

5.1.2 返回值说明

对于开发接口，如果调用成功，返回值为 int 型的接口都会返回 MSP_SUCCESS，否则返回错误代码，错误代码参见 msp_errors.h；返回值为指针类型的接口，函数执行失败返回空指针 0，错误代码可以通过相关的回传参数查看，函数执行成功但无数据时返回空指针 0，有数据时返回非 0 指针。

5.2 函数调用

5.2.1 QIVWSessionBegin

□ **函数原型**

const char* MSPAPI QIVWSessionBegin(const char *grammarList, const char *params, int *errorCode)

□ **功能**

本接口用来开始唤醒功能，并在参数中指定唤醒(唤醒+识别时)用到的语法列表，本次唤醒所用的参数等。

□ **参数**

♦ **grammarList [in]**

bnf 格式的语法，一个引擎内置语法列表。可以同时指定多个语法，不同的语法之间

以“,”隔开。进行语音唤醒时不需要语法，此参数设定为 NULL 或空串即可；进行语音唤醒+识别时则需要语法，语法可以在此参数中指定。

♦ **params [in]**

本路唤醒使用的参数：

参数	名称	说明
sst	业务类型	唤醒业务类型,可以设置如下参数： wakeup：语音唤醒(默认) oneshot：唤醒加识别
ivw_threshold	唤醒词门限	多唤醒时门限设置格式,如下： id0:xx;id1:xx;….。 示例：0:15;1:-10 表示设置第一个唤醒词的门限值为 15，第二个唤醒的门限值是-10，门限值越低越容易唤醒成功。注意：设置的门限值的个数多于唤醒资源中唤醒词的个数时会报错。
ivw_shot_word	音频是否包含唤醒词	用于唤醒加识别时场景，将音频送入识别引擎时是否包含唤醒词音频： 0：不包含 1：包含(默认)

说明：在使用“唤醒+识别”业务时，在唤醒成功之后进行识别业务，识别业务参数设置参见 第四章 识别接口说明。

♦ **errorCode [out]**

如果函数调用成功则其值为 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_CREATE_HANDLE	创建实例失败

□ **返回值**

MSC 为本次唤醒建立的 ID，用来唯一的标识本次唤醒，供以后调用其他函数时使用。函数调用失败则会返回 NULL。

□ 说明

此处设定的参数本次唤醒中一直有效。此函数需要和接口 QIVWSessionEnd 配对使用，在这两个接口之间可以调用实际完成唤醒功能的函数如写入音频。没有调用此函数则 MSC 不会建立和当前线程有关的实例，后续的函数调用会因为没有合法的 ID 而无法完成对应的功能。

□ 用法示例

```
const char*     params = " ivw_threshold=0:15, ivw_res_path =fo|res/ivw/50287829_讯飞雨点.irf";
int   ret = 0;
const char*     session_id = QIVWSessionBegin( NULL, params, &ret );
if( 0 != ret )
{
      printf( “QIVWSessionBegin failed, error code is: %d”, ret );
}
```

□ 参见

QIVWSessionEnd。

5.2.2 QIVWRegisterNotify

□ 函数原型

int MSPAPI QIVWRegisterNotify(const char *sessionID, ivw_ntf_handler msgProcCb, void *userData)

□ 功能

注册回调。

□ 参数

♦ **sessionID [in]**

由 QISRSessionBegin 返回过来的 ID。

♦ **msgProcCb [in]**

注册通知的回调函数，唤醒结果将在此注册回调中返回。格式为：

typedef int(*ivw_ntf_handler)(const char *sessionID, int msg, int param1, int param2, const void *info, void *userData);

参数说明:

参数	说明		
sessionID	由 QIVWSessionBegin 返回过来的 ID		
msg	MSP_IVW_MSG_WAKEUP=1	唤醒消息，在 info	

		中给出唤醒结果缓存首地址，param2 给出唤醒结果的长度。
	MSP_IVW_MSG_ERROR=2	出错通知消息，在 param1 中给出错误吗。
	MSP_IVW_MSG_ISR_RESULT=3	唤醒+识别结果消息，在 info 中给出识别结果缓存首地址，param2 给出识别结果的长度。param1 中给出给出结果状态，结果状态值参见 QISRAudioWrite 接口中结果状态说明。
	MSP_IVW_MSG_ISR_EPS=4	唤醒+识别结果中 vad 端点检测消息，param1 给出端点检测状态，状态值参见 QISRAudioWrite 接口中端点检测状态说明。
param1	参见 msg 消息说明	
param2	参见 msg 消息说明	
info	参见 msg 消息说明	
userData	用户数据	

✧ **唤醒结果成员说明**

成员名	参数解释
sst	本次业务标识：wakeup 表示语音唤醒；enroll 表示唤醒词训练（当前版本不支持）
id	当前唤醒词的 id
score	当前唤醒得分
bos	当前唤醒音频的前端点
eos	当前唤醒音频的尾端点

♦ **userData [in]**

用户数据。

□ **返回值**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_INVALID_PARA	无效的参数

□ **说明**

通过此函数注册回调函数到 msc。如果唤醒成功，msc 调用回调函数通知唤醒成功息同时给出相应唤醒数据。如果出错，msc 调用回调函数给出错误信息。

□ **用法示例**

```
int cb_ivw_msg_proc( const char *sessionID, int msg, int param1, int param2, const void *info,
void *userData )
{
    if (MSP_IVW_MSG_ERROR == msg) //唤醒出错消息
    {
        printf("\n\nMSP_IVW_MSG_ERROR errCode = %d\n\n", param1);
    }
    else if (MSP_IVW_MSG_WAKEUP == msg) //唤醒成功消息
    {
        printf("\n\nMSP_IVW_MSG_WAKEUP result = %s\n\n", info);
    }
    return 0;
}

err_code = QIVWRegisterNotify(session_id, cb_ivw_msg_proc,NULL);
if (err_code != MSP_SUCCESS)
{
    printf("QIVWRegisterNotify failed! error code:%d\n",err_code);
}
```

□ **参见**

无

5.2.3 QIVWAudioWrite

□ **函数原型**

int MSPAPI QIVWAudioWrite(const char *sessionID, const void *audioData, unsigned int audioLen, int audioStatus)

□ 功能

写入本次唤醒的音频，本接口需要反复调用直到音频写完为止。

□ 参数

♦ **sessionID [in]**

由 QIVWSessionBegin 返回过来的 ID。

♦ **audioData [in]**

音频数据缓冲区起始地址。

♦ **audioLen [in]**

音频数据长度，单位字节。

♦ **audioStatus [in]**

用来指明用户本次识别的音频是否发送完毕，可能的值如下：

枚举常量	简介
MSP_AUDIO_SAMPLE_FIRST = 1	第一块音频
MSP_AUDIO_SAMPLE_CONTINUE = 2	还有后继音频
MSP_AUDIO_SAMPLE_LAST = 4	最后一块音频

在 MSCV5+中，MSP_AUDIO_SAMPLE_LAST（0x04）用来指明当前的音频已经发送完毕，除此之外的任何值都将被 MSC 视为还有后继的音频。

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID
MSP_ERROR_NULL_HANDLE	空句柄
MSP_ERROR_INVALID_PARA	无效的参数
MSP_ERROR_INVALID_PARA_VALUE	无效的参数值
MSP_ERROR_NO_ENOUGH_BUFFER	缓冲区溢出

□ 说明

调用本接口时，推荐用户在写入音频时采取“边录边写”的方式，即每隔一小段时间将采集到的音频通过本接口写入 MSC。

□ 用法示例

```
char          audio_data[ 5120 ] = “”;
unsigned int  audio_len = 0;
int           audio_status = 2;
int           ret = 0;
```

```
while(MSP_AUDIO_SAMPLE_LAST != audio_status )
{
    …   // 读取音频到缓冲区 audio_data 中，设置音频长度 audio_len，音频状态 audio_status。
ret = QIVWAudioWrite( session_id, audio_data, audio_len, audio_status
);
if( 0 ! = ret )
{
    printf( “QIVWAudioWrite failed, error code is: %d”, ret );
    break;
}
}
```

□ **参见**

无

5.2.4 QIVWSessionEnd

□ **函数原型**

int MSPAPI QIVWSessionEnd(const char *sessionID, const char *hints)

□ **功能**

结束语音唤醒。

□ **参数**

♦ **sessionID [in]**

由 QIVWSessionBegin 返回过来的 ID。

♦ **hints [in]**

结束本次唤醒的原因描述，用于记录日志，便于用户查阅或者跟踪某些问题。

□ **返回值**

如果函数调用成功返回 0，否则返回错误代码，错误代码参见 msp_errors.h 和 msp_error.h。几个主要的返回值如下：

返回值	意义
MSP_ERROR_NOT_INIT	未初始化
MSP_ERROR_INVALID_HANDLE	无效的 ID

□ **说明**

本接口需要和 QIVWSessionBegin 配合使用，用来本次语音唤醒。

调用本函数后，关于当前语音唤醒的所有资源（参数，语法，音频，实例等）都会被释放，所以用户不应该再针对该实例做任何操作（比如使用其 SessionID 等）。

□ **用法示例**

```
int   ret = QIVWSessionEnd ( session_id, “normal end” );
```

```
if( 0 != ret )
{
	printf( “QIVWSessionEnd failed, error code is: %d”, ret );
}
```

□ 参见

QIVWSessionBegin

第 6 章 错误码的定义

MSCv5+ 客户端子系统返回的错误码都在 msp_errors.h 中定义，大致可以分为一般错误、网络错误、资源错误、登录错误和 HTTP 错误等。

6.1 宏

和错误码有关的宏定义：

```
#define MSP_HTTP_ERROR (x) ((x) + MSP_ERROR_HTTP_BASE)
```

将 HTTP 错误码×××映射为 12×××，即关于 HTTP 的错误码后三位同 HTTP 协议中定义的错误码是一致的。

6.2 错误码列表

错误码	*错误值*	*意义*
MSC		
MSP_SUCCESS	0	函数执行成功
MSP_ERROR_FAIL	-1	失败
MSP_ERROR_EXCEPTION	-2	异常
/* General errors 10100(0x2774) */		
MSP_ERROR_GENERAL	10100	基码
MSP_ERROR_OUT_OF_MEMORY	10101	内存越界
MSP_ERROR_FILE_NOT_FOUND	10102	文件没有发现
MSP_ERROR_NOT_SUPPORT	10103	不支持
MSP_ERROR_NOT_IMPLEMENT	10104	没有实现
MSP_ERROR_ACCESS	10105	没有权限
MSP_ERROR_INVALID_PARA	10106	无效的参数
MSP_ERROR_INVALID_PARA_VALUE	10107	无效的参数值
MSP_ERROR_INVALID_HANDLE	10108	无效的句柄
MSP_ERROR_INVALID_DATA	10109	无效的数据
MSP_ERROR_NO_LICENSE	10110	没有授权许可
MSP_ERROR_NOT_INIT	10111	没有初始化
MSP_ERROR_NULL_HANDLE	10112	空句柄
MSP_ERROR_OVERFLOW	10113	溢出
MSP_ERROR_TIME_OUT	10114	超时
MSP_ERROR_OPEN_FILE	10115	打开文件出错
MSP_ERROR_NOT_FOUND	10116	没有发现
MSP_ERROR_NO_ENOUGH_BUFFER	10117	没有足够的内存
MSP_ERROR_NO_DATA	10118	没有数据

MSP_ERROR_NO_MORE_DATA	10119	没有更多的数据
MSP_ERROR_NO_RESPONSE_DATA	10120	跳过
MSP_ERROR_ALREADY_EXIST	10121	已经存在
MSP_ERROR_LOAD_MODULE	10122	加载模块失败
MSP_ERROR_BUSY	10123	忙碌
MSP_ERROR_INVALID_CONFIG	10124	无效的配置项
MSP_ERROR_VERSION_CHECK	10125	版本错误
MSP_ERROR_CANCELED	10126	取消
MSP_ERROR_INVALID_MEDIA_TYPE	10127	无效的媒体类型
MSP_ERROR_CONFIG_INITIALIZE	10128	初始化 Config 实例
MSP_ERROR_CREATE_HANDLE	10129	建立句柄
MSP_ERROR_CODING_LIB_NOT_LOAD	10130	编解码库未加载
MSP_ERROR_USER_CANCELLED	10131	用户取消
MSP_ERROR_INVALID_OPERATION	10132	无效的操作
MSP_ERROR_MESSAGE_NOT_COMPLETE	10133	
MSP_ERROR_USER_ACTIVE_ABORT	10136	
MSP_ERROR_BUSY_GRMBUILDING	10137	
MSP_ERROR_BUSY_LEXUPDATING	10138	
/* Error codes of network 10200(0x27D8)*/		
MSP_ERROR_NET_GENERAL	10200	网络一般错误
MSP_ERROR_NET_OPENSOCK	10201	打开套接字
MSP_ERROR_NET_CONNECTSOCK	10202	套接字连接
MSP_ERROR_NET_ACCEPTSOCK	10203	套接字接收
MSP_ERROR_NET_SENDSOCK	10204	发送错误
MSP_ERROR_NET_RECVSOCK	10205	接收错误
MSP_ERROR_NET_INVALIDSOCK	10206	无效的套接字
MSP_ERROR_NET_BADADDRESS	10207	无效的地址
MSP_ERROR_NET_BINDSEQUENCE	10208	绑定次序
MSP_ERROR_NET_NOTOPENSOCK	10209	套接字没有打开
MSP_ERROR_NET_NOTBIND	10210	没有绑定
MSP_ERROR_NET_NOTLISTEN	10211	没有监听
MSP_ERROR_NET_CONNECTCLOSE	10212	连接关闭
MSP_ERROR_NET_NOTDGRAMSOCK	10213	非数据报套接字
MSP_ERROR_NET_DNS	10214	DNS 解析错误
MSP_ERROR_NET_INIT	10215	网络初始化错误
/* Error codes of mssp message 10300(0x283C) */		
MSP_ERROR_MSG_GENERAL	10300	消息一般错误
MSP_ERROR_MSG_PARSE_ERROR	10301	解析出错
MSP_ERROR_MSG_BUILD_ERROR	10302	构建出错
MSP_ERROR_MSG_PARAM_ERROR	10303	参数出错

MSP_ERROR_MSG_CONTENT_EMPTY	10304	Content 为空
MSP_ERROR_MSG_INVALID_CONTENT_TYPE	10305	Content 类型无效
MSP_ERROR_MSG_INVALID_CONTENT_LENGTH	10306	Content 长度无效
MSP_ERROR_MSG_INVALID_CONTENT_ENCODE	10307	Content 编码无效
MSP_ERROR_MSG_INVALID_KEY	10308	Key 无效
MSP_ERROR_MSG_KEY_EMPTY	10309	Key 为空
MSP_ERROR_MSG_SESSION_ID_EMPTY	10310	会话 ID 为空
MSP_ERROR_MSG_LOGIN_ID_EMPTY	10311	登录 ID 为空
MSP_ERROR_MSG_SYNC_ID_EMPTY	10312	同步 ID 为空
MSP_ERROR_MSG_APP_ID_EMPTY	10313	应用 ID 为空
MSP_ERROR_MSG_EXTERN_ID_EMPTY	10314	扩展 ID 为空
MSP_ERROR_MSG_INVALID_CMD	10315	无效的命令
MSP_ERROR_MSG_INVALID_SUBJECT	10316	无效的主题
MSP_ERROR_MSG_INVALID_VERSION	10317	无效的版本
MSP_ERROR_MSG_NO_CMD	10318	没有命令
MSP_ERROR_MSG_NO_SUBJECT	10319	没有主题
MSP_ERROR_MSG_NO_VERSION	10320	没有版本号
MSP_ERROR_MSG_MSSP_EMPTY	10321	消息为空
MSP_ERROR_MSG_NEW_RESPONSE	10322	新建响应消息失败
MSP_ERROR_MSG_NEW_CONTENT	10323	新建 Content 失败
MSP_ERROR_MSG_INVALID_SESSION_ID	10324	无效的 ID
MSP_ERROR_MSG_INVALID_CONTENT	10325	
/* Error codes of DataBase 10400(0x28A0)*/		
MSP_ERROR_DB_GENERAL	10400	数据库一般错误
MSP_ERROR_DB_EXCEPTION	10401	异常
MSP_ERROR_DB_NO_RESULT	10402	没有结果
MSP_ERROR_DB_INVALID_USER	10403	无效的用户
MSP_ERROR_DB_INVALID_PWD	10404	无效的密码
MSP_ERROR_DB_CONNECT	10405	连接出错
MSP_ERROR_DB_INVALID_SQL	10406	无效的 SQL
MSP_ERROR_DB_INVALID_APPID	10407	无效的应用 ID
/* Error codes of Resource 10500(0x2904)*/		
MSP_ERROR_RES_GENERAL	10500	资源一般错误
MSP_ERROR_RES_LOAD	10501	没有加载
MSP_ERROR_RES_FREE	10502	空闲
MSP_ERROR_RES_MISSING	10503	缺失

MSP_ERROR_RES_INVALID_NAME	10504	无效的名称
MSP_ERROR_RES_INVALID_ID	10505	无效的 ID
MSP_ERROR_RES_INVALID_IMG	10506	无效的映像
MSP_ERROR_RES_WRITE	10507	写操作错误
MSP_ERROR_RES_LEAK	10508	泄露
MSP_ERROR_RES_HEAD	10509	资源头部错误
MSP_ERROR_RES_DATA	10510	数据出错
MSP_ERROR_RES_SKIP	10511	跳过
/* Error codes of TTS 10600(0x2968)*/		
MSP_ERROR_TTS_GENERAL	10600	合成一般错误
MSP_ERROR_TTS_TEXTEND	10601	文本结束
MSP_ERROR_TTS_TEXT_EMPTY	10602	文本为空
MSP_ERROR_TTS_LTTS_ERROR	10603	
/* Error codes of Recognizer 10700(0x29CC) */		
MSP_ERROR_REC_GENERAL	10700	一般错误
MSP_ERROR_REC_INACTIVE	10701	处于不活跃状态
MSP_ERROR_REC_GRAMMAR_ERROR	10702	语法错误
MSP_ERROR_REC_NO_ACTIVE_GRAMMARS	10703	没有活跃的语法
MSP_ERROR_REC_DUPLICATE_GRAMMAR	10704	语法重复
MSP_ERROR_REC_INVALID_MEDIA_TYPE	10705	无效的媒体类型
MSP_ERROR_REC_INVALID_LANGUAGE	10706	无效的语言
MSP_ERROR_REC_URI_NOT_FOUND	10707	没有对应的 URI
MSP_ERROR_REC_URI_TIMEOUT	10708	获取 URI 内容超时
MSP_ERROR_REC_URI_FETCH_ERROR	10709	获取 URI 内容时出错
/* Error codes of Speech Detector 10800(0x2A30) */		
MSP_ERROR_EP_GENERAL	10800	（EP）一般错误
MSP_ERROR_EP_NO_SESSION_NAME	10801	（EP）链接没有名字
MSP_ERROR_EP_INACTIVE	10802	（EP）不活跃
MSP_ERROR_EP_INITIALIZED	10803	（EP）初始化出错
/* Error codes of TUV */		
MSP_ERROR_TUV_GENERAL	10900	
MSP_ERROR_TUV_GETHIDPARAM	10901	
MSP_ERROR_TUV_TOKEN	10902	
MSP_ERROR_TUV_CFGFILE	10903	
MSP_ERROR_TUV_RECV_CONTENT	10904	

MSP_ERROR_TUV_VERFAIL	10905	
/* Error codes of IMTV */		
MSP_ERROR_LOGIN_SUCCESS	11000	登录成功
MSP_ERROR_LOGIN_NO_LICENSE	11001	无授权
MSP_ERROR_LOGIN_SESSIONID_INVALID	11002	无效的SessionID
MSP_ERROR_LOGIN_SESSIONID_ERROR	11003	错误的SessionID
MSP_ERROR_LOGIN_UNLOGIN	11004	未登录
MSP_ERROR_LOGIN_INVALID_USER	11005	无效的用户
MSP_ERROR_LOGIN_INVALID_PWD	11006	无效的密码
MSP_ERROR_LOGIN_SYSTEM_ERROR	11099	系统错误
/* Error Codes using in local engine */		
MSP_ERROR_AUTH_NO_LICENSE	11200	
MSP_ERROR_AUTH_NO_ENOUGH_LICENSE	11201	
MSP_ERROR_AUTH_INVALID_LICENSE	11202	
MSP_ERROR_AUTH_LICENSE_EXPIRED	11203	
MSP_ERROR_AUTH_NEED_MORE_DATA	11204	
MSP_ERROR_AUTH_LICENSE_TO_BE_EXPIRED	11205	
MSP_ERROR_AUTH_INVALID_MACHINE_ID	11206	
MSP_ERROR_AUTH_LOCAL_ASR_FORBIDDEN	11207	
MSP_ERROR_AUTH_LOCAL_TTS_FORBIDDEN	11208	
MSP_ERROR_AUTH_LOCAL_IVW_FORBIDDEN	11209	
/* Error codes of HCR */		
MSP_ERROR_HCR_GENERAL	11100	
MSP_ERROR_HCR_RESOURCE_NOT_EXIST	11101	
MSP_ERROR_HCR_CREATE	11102	
MSP_ERROR_HCR_DESTROY	11103	
MSP_ERROR_HCR_START	11104	
MSP_ERROR_HCR_APPEND_STROKES	11105	
MSP_ERROR_HCR_INIT	11106	
MSP_ERROR_HCR_POINT_DECODE	11107	
MSP_ERROR_HCR_DISPATCH	11108	
MSP_ERROR_HCR_GETRESULT	11109	
MSP_ERROR_HCR_RESOURCE	11110	
/* Error codes of http 12000(0x2EE0) */		
MSP_ERROR_HTTP_BASE	12000	HTTP 错误基码

/*Error codes of ISV */		
MSP_ERROR_ISV_NO_USER	13000	
/* Error codes of Lua scripts */		
MSP_ERROR_LUA_BASE	14000	
MSP_ERROR_LUA_YIELD	14001	
MSP_ERROR_LUA_ERRRUN	14002	
MSP_ERROR_LUA_ERRSYNTAX	14003	
MSP_ERROR_LUA_ERRMEM	14004	
MSP_ERROR_LUA_ERRERR	14005	
MSP_ERROR_LUA_INVALID_PARAM	14006	
/* Error codes of MMP */		
MSP_ERROR_MMP_BASE	15000	
MSP_ERROR_MMP_MYSQL_INITFAIL	15001	
MSP_ERROR_MMP_REDIS_INITFAIL	15002	
MSP_ERROR_MMP_NETDSS_INITFAIL	15003	
MSP_ERROR_MMP_TAIR_INITFAIL	15004	
MSP_ERROR_MMP_MAIL_SESSION_FAIL	15006	
MSP_ERROR_MMP_MAIL_LOGON_FAIL	15007	
MSP_ERROR_MMP_MAIL_USER_ILLEGAL	15008	
MSP_ERROR_MMP_MAIL_PWD_ERR	15009	
MSP_ERROR_MMP_MAIL_SOCKET_ERR	15010	
MSP_ERROR_MMP_MAIL_INIT_FAIL	15011	
MSP_ERROR_MMP_STORE_MNR_NO_INIT	15012	
MSP_ERROR_MMP_STORE_MNR_POOL_FULL	15013	
MSP_ERROR_MMP_STRATGY_PARAM_ILLEGAL	15014	
MSP_ERROR_MMP_STRATGY_PARAM_TOOLOOG	15015	
MSP_ERROR_MMP_PARAM_NULL	15016	
MSP_ERROR_MMP_ERR_MORE_TOTAL	15017	
MSP_ERROR_MMP_PROC_THRESHOLD	15018	
MSP_ERROR_MMP_SERVER_THRESHOLD	15019	
MSP_ERROR_MMP_PYTHON_NO_EXIST	15020	
MSP_ERROR_MMP_PYTHON_IMPORT_FAILED	15021	
MSP_ERROR_MMP_PYTHON_BAD_FUNC	15022	
MSP_ERROR_MMP_DB_DATA_ILLEGAL	15023	
MSP_ERROR_MMP_REDIS_NOT_CONN	15024	
MSP_ERROR_MMP_PMA_NOT_FOUND_STRATEGY	15025	
MSP_ERROR_MMP_TAIR_CONNECT	15026	
MSP_ERROR_MMP_PMC_SERVINFO_INVALID	15027	
MSP_ERROR_MMP_ALARM_GROUP_NULL	15028	
MSP_ERROR_MMP_ALARM_CONTXT_NULL	15029	

/* Error codes of MSC(lmod loader) */		
MSP_ERROR_LMOD_BASE	16000	
MSP_ERROR_LMOD_NOT_FOUND	16001	
MSP_ERROR_LMOD_UNEXPECTED_BIN	16002	
MSP_ERROR_LMOD_LOADCODE	16003	
MSP_ERROR_LMOD_PRECALL	16004	
MSP_ERROR_LMOD_RUNTIME_EXCEPTION	16005	
MSP_ERROR_LMOD_ALREADY_LOADED	16006	
// Error code of Third Business		
MSP_ERROR_BIZ_BASE	17000	
//Error of Nginx errlog file increase		
MSP_ERROR_NGX_LOG_MORE_TOTEL_SIZE	18000	
Speech plus		
/*Error Code Of Speech plus*/		
SPEECH_ERROR_NO_NETWORK	20001	
SPEECH_ERROR_NETWORK_TIMEOUT	20002	
SPEECH_ERROR_NET_EXPECTION	20003	
SPEECH_ERROR_INVALID_RESULT	20004	
SPEECH_ERROR_NO_MATCH	20005	
SPEECH_ERROR_AUDIO_RECORD	20006	
SPEECH_ERROR_NO_SPPECH	20007	
SPEECH_ERROR_SPEECH_TIMEOUT	20008	
SPEECH_ERROR_EMPTY_UTTERANCE	20009	
SPEECH_ERROR_FILE_ACCESS	20010	
SPEECH_ERROR_PLAY_MEDIA	20011	
SPEECH_ERROR_INVALID_PARAM	20012	
SPEECH_ERROR_TEXT_OVERFLOW	20013	
SPEECH_ERROR_INVALID_DATA	20014	
SPEECH_ERROR_LOGIN	20015	
SPEECH_ERROR_PERMISSION_DENIED	20016	
SPEECH_ERROR_INTERRUPT	20017	
SPEECH_ERROR_VERSION_LOWER	20018	
SPEECH_ERROR_UNKNOWN	20999	

SPEECH_ERROR_COMPONENT_NOT_INSTALLED	21001	
SPEECH_ERROR_ENGINE_NOT_SUPPORTED	21002	
SPEECH_ERROR_ENGINE_INIT_FAIL	21003	
SPEECH_ERROR_ENGINE_CALL_FAIL	21004	
SPEECH_ERROR_ENGINE_BUSY	21005	
SPEECH_ERROR_LOCAL_NO_INIT	22001	
SPEECH_ERROR_LOCAL_RESOURCE	22002	
SPEECH_ERROR_LOCAL_ENGINE	22003	
SPEECH_ERROR_IVW_INTERRUPT	22004	
Speech iflytek Engines		
/*Error Code Of Local iflytek Engines*/		
/*Error Code Of AiTalk*/		
/*Error Code Of AiTalk Operation*/		
SPEECH_SUCCESS	0	
SPEECH_ERROR_ASR_CLIENT	23000	
SPEECH_ERROR_ASR_INVALID_PARA	23001	
SPEECH_ERROR_ASR_INVALID_PARA_VALUE	23002	
SPEECH_ERROR_ASR_OUT_OF_MEMORY	23003	
SPEECH_ERROR_ASR_CREATE_HANDLE_FAILED	23004	
SPEECH_ERROR_ASR_ENGINE_INIT_FAILED	23005	
SPEECH_ERROR_ASR_ENGINE_STARTED	23006	
SPEECH_ERROR_ASR_ENGINE_UNINIT	23007	
SPEECH_ERROR_ASR_SPEECH_TIMEOUT	23008	
SPEECH_ERROR_ASR_NO_RECOGNIZED_RESULT	23009	
SPEECH_ERROR_ASR_INVALID_HANDLE	23010	
SPEECH_ERROR_ASR_FILE_ACCESS	23011	
/*Error Code Of AiTalk Engine*/		
SPEECH_ERROR_AITALK_FALSE	23100	
/* For license check */		
SPEECH_ERROR_AITALK_PERMISSION_DENIED	23101	
/* General */		
SPEECH_ERROR_AITALK_INVALID_PARA	23102	
SPEECH_ERROR_AITALK_BUFFER_OVERFLOW	23103	
SPEECH_ERROR_AITALK_FAILED	23104	

SPEECH_ERROR_AITALK_NOT_SUPPORTED	23105	
SPEECH_ERROR_AITALK_OUT_OF_MEMORY	23106	
SPEECH_ERROR_AITALK_INVALID_RESOURCE	23107	
SPEECH_ERROR_AITALK_NOT_FOUND	23108	
SPEECH_ERROR_AITALK_INVALID_GRAMMAR	23109	
/* For object status */		
SPEECH_ERROR_AITALK_INVALID_CALL	23110	
/* For ASR Input */		
SPEECH_ERROR_AITALK_SYNTAX_ERROR	23111	
/* For Message Call Back */		
SPEECH_ERROR_AITALK_RESET	23112	
SPEECH_ERROR_AITALK_ENDED	23113	
SPEECH_ERROR_AITALK_IDLE	23114	
SPEECH_ERROR_AITALK_CANNOT_SAVE_FILE	23115	
/* For Lexicon name */		
SPEECH_ERROR_AITALK_INVALID_GRAMMAR_NAME	23116	
SPEECH_ERROR_AITALK_BUFFER_EMPTY	23117	
SPEECH_ERROR_AITALK_GET_RESULT	23118	
SPEECH_ERROR_AITALK_REACT_OUT_TIME	23119	
SPEECH_ERROR_AITALK_SPEECH_OUT_TIME	23120	
SPEECH_ERROR_AITALK_AUDIO_CUT	23121	
SPEECH_ERROR_AITALK_AUDIO_LOWER	23122	
SPEECH_ERROR_AITALK_INSUFFICIENT_PERMISSIONS	23123	
SPEECH_ERROR_AITALK_RESULT_ERROR	23124	
SPEECH_ERROR_AITALK_SHORT_PAUSE	23125	
SPEECH_ERROR_AITALK_BUSY	23126	
SPEECH_ERROR_AITALK_GRM_NOT_UPDATE	23127	
SPEECH_ERROR_AITALK_STARTED	23128	
SPEECH_ERROR_AITALK_STOPPED	23129	
SPEECH_ERROR_AITALK_ALREADY_STARTED	23130	
SPEECH_ERROR_AITALK_ALREADY_STOPPED	23131	
SPEECH_ERROR_AITALK_TOO_MANY_COMMAND	23132	
SPEECH_ERROR_AITALK_WAIT	23133	
SPEECH_ERROR_AITALK_MAE_RIGHT	23134	

SPEECH_ERROR_AITALK_MAE_WRONG	23135	
SPEECH_ERROR_AITALK_GRM_ERR	23300	
/*Error Code Of AiSound*/		
/*Error Code Of AiSound Operation*/		
SPEECH_ERROR_TTS_INVALID_PARA	24000	
SPEECH_ERROR_TTS_INVALID_PARA_VALUE	24001	
SPEECH_ERROR_TTS_OUT_OF_MEMORY	24002	
SPEECH_ERROR_TTS_INVALID_HANDLE	24003	
SPEECH_ERROR_TTS_CREATE_HANDLE_FAILED	24004	
SPEECH_ERROR_TTS_INVALID_RESOURCE	24005	
SPEECH_ERROR_TTS_INVALID_VOICE_NAME	24006	
SPEECH_ERROR_TTS_ENGINE_UNINIT	24007	
SPEECH_ERROR_TTS_ENGINE_INIT_FAILED	24008	
SPEECH_ERROR_TTS_ENGINE_BUSY	24009	
/*Error Code Of AiSound Engine*/		
SPEECH_ERROR_AISOUND_BASE	24100	
SPEECH_ERROR_AISOUND_UNIMPEMENTED	24100	
SPEECH_ERROR_AISOUND_UNSUPPORTED	24101	
SPEECH_ERROR_AISOUND_INVALID_HANDLE	24102	
SPEECH_ERROR_AISOUND_INVALID_PARA	24103	
SPEECH_ERROR_AISOUND_INSUFFICIENT_HEAP	24104	
SPEECH_ERROR_AISOUND_STATE_REFUSE	24105	
SPEECH_ERROR_AISOUND_INVALID_PARA_ID	24106	
SPEECH_ERROR_AISOUND_INVALID_PARA_VALUE	24107	
SPEECH_ERROR_AISOUND_RESOURCE	24108	
SPEECH_ERROR_AISOUND_RESOURCE_READ	24109	
SPEECH_ERROR_AISOUND_LBENDIAN	24110	
SPEECH_ERROR_AISOUND_HEADFILE	24111	
SPEECH_ERROR_AISOUND_BUFFER_OVERFLOW	24112	
SPEECH_ERROR_AISOUND_INVALID_ISAMPA	24113	
SPEECH_ERROR_AISOUND_INVALID_CSSML	24114	
/*Error Code Of IVW*/		
/*Error Code Of IVW Operation*/		
SPEECH_ERROR_IVW_ENGINE_UNINI	25000	

SPEECH_ERROR_IVW_RESVER_NOMATCH	25001	
/*Error Code Of IVW Engine*/		
SPEECH_ERROR_IVW_INVALID_CALL	25101	
SPEECH_ERROR_IVW_INVALID_ARG	25102	
SPEECH_ERROR_IVW_TELL_SIZE	25103	
SPEECH_ERROR_IVW_OUT_OF_MEMORY	25104	
SPEECH_ERROR_IVW_OUT_BUFFER_FULL	25105	
SPEECH_ERROR_IVW_OUT_BUFFER_EMPTY	25106	
SPEECH_ERROR_IVW_INVALID_RESOURCE	25107	
SPEECH_ERROR_IVW_REPETITIOPN_ENTER	25108	
SPEECH_ERROR_IVW_NOT_SUPPORT	25109	
SPEECH_ERROR_IVW_NOT_FOUND	25110	
SPEECH_ERROR_IVW_INVALID_SN	25111	
SPEECH_ERROR_IVW_LIMITTED	25112	
SPEECH_ERROR_IVW_TIME_OUT	25113	
SPEECH_ERROR_IVW_ENROLL1_SUCESS	25114	
SPEECH_ERROR_IVW_ENROLL1_FAILED	25115	
SPEECH_ERROR_IVW_ENROLL2_SUCESS	25116	
SPEECH_ERROR_IVW_ENROLL2_FAILED	25117	
SPEECH_ERROR_IVW_ENROLL3_SUCESS	25118	
SPEECH_ERROR_IVW_ENROLL3_FAILED	25119	
SPEECH_ERROR_IVW_SPEECH_TOO_SHORT	25120	
SPEECH_ERROR_IVW_SPEECH_STOP	25121	

第 7 章 发音人列表

发音人	参数名称	语种/方言	音色
小燕	xiaoyan	普通话	青年女声
燕平	yanping	普通话	青年女声
宇峰	yufeng	普通话	青年男声
晓婧	jinger	普通话	青年女声
唐老鸭	donaldduck	普通话	卡通
许小宝	baybyxu	普通话	童声
楠楠	nannan	普通话	童声
晓梦	xiaomeng	普通话	青年女声
晓琳	xiaolin	台湾普通话	青年女声
晓倩	xiaoqian	东北话	青年女声
晓蓉	xiaorong	四川话	青年女声
小坤	xiaokun	河南话	青年男声
小强	xiaoqiang	湖南话	青年男声
晓美	xiaomei	粤语	青年女声
大龙	dalong	粤语	青年男声
Catherine	catherine	美式纯英文	青年女声
John	john	美式纯英文	青年男声
henry	henry	英文	青年男声
古丽	Guli	维吾尔语	青年女声
玛丽安	Mariane	法语	青年女声
阿拉本	Allabent	俄罗斯语	青年女声
加芙列拉	Gabriela	西班牙语	青年女声
艾伯哈	Abha	印地语	青年女声
小云	XiaoYun	越南语	青年女声

第 8 章 常见问题解答

问题 1：调用 MSPLogin / MSPSetParam /QISRSessionBegin / QTTSSessionBegin 返回 10106-无效的参数。

答：原因可能是：

1) params 字符串没有按照格式书写，正确的是以逗号隔开的参数对（参数名=参数值）组成的字符串。
2) params 字符串不是合法的 C 风格字符串，有些平台（如 Android）需要手动在字符串末尾加’\0’字符。

问题 2：拿到了合成音频但不知道如何来播放。

答：合成拿到的音频是没有音频头的，音频头中含有音频格式、采样率、音频长度等播放音频所需信息。拿到合成音频后，用户可以添加音频头，然后使用常规播放器来播放；也可以使用 Cool Edit 等软件手动选择音频参数来播放。

问题 3：如何进行大文本的合成

答：语音云一次语音合成允许的合成文本最大不超过 8192 个字节，所以对于长度超过此值的大合成文本，用户可以采用“分段合成”的方式，即先将大文本按照标点符号如句号进行切分，然后对每一段文本分别进行合成，可以为每一段文本使用一路独立的会话完成合成。

问题 4：获取不到识别/听写结果。

答：原因可能是：

1) QISRSessionBegin 的参数设置不正确，如没有设置好正确的引擎类型等。
2) 音频格式不对，客户端支持的音频编解码算法只支持 16 位 Intel PCM 格式的音频。

问题 5：能获取到语音听写结果但是不全。

答：此问题主要是在调用 QISRAudioWrite 时没有正确设置参数 audioStatus 所致，此参数在写入非最后一个音频数据块时需要设置为 2，写入最后一个数据块时需要设置为 4，以告诉 MSC 音频写入完毕。如果只有一个音频数据块，audioStatus 也需要设置为 4。

问题 6：可以拿到识别或转写结果但是响应很慢。

答：此问题可以尝试如下方法来解决：

1) 调用 QISRAudioWrite 接口写音频数据时，尽量做到“匀速发送”——周期性的

发送定长数据，做到边录边发，避免一次发送数据量过大的音频。

2) 采用 QISRAudioWrite 接口和 QISRGetResult 接口混调的方式。在调用 QISRAudioWrite 接口时，可以检查 out 型参数 recogStatus，如果其值为 0，表明已经有（部分）识别结果缓存在 MSC 中了，此时可以调用 QISRGetResult 来获取结果。

第 9 章 技术支持

如果您在安装、使用或开发过程中遇到任何问题或者建议，请与我们联系！

联系时对问题的描述请尽量包含以下内容：

- 系统配置（包括 CPU、内存、硬盘、操作系统及产品版本等信息）
- 问题细节（包括问题的重现过程及合成的文本内容、识别音频等）
- 问题重现（包括详细的操作过程和运行日志等）

科大讯飞提供以下方式的技术支持：

- 电话支持

请于周一～周五，北京时间 9：00～17：00 间，拨打电话： 0551－65331813 获得技术支持信息。

- 电子邮件支持

请将问题的详细描述发至：msp_support@iflytek.com。

- 在线支持

请登录我们的论坛 http://club.voicecloud.cn/forum.php。

- 信件支持

请将问题的详细描述发至：

中国安徽省合肥市望江西路 666 号科大讯飞语音产业基地 邮编 230088

或传真至： 0551－65331801 65331802