Graph.cpp

#include "graph.h"

Graph::Graph(QWidget \*parent) : QWidget(parent) //类的构造

{

graph\_space = 0; //相关数据的初始化

for (int i = 0; i < 450; ++i) {

graph\_data[i] = 0;

}

}

void Graph::startTime(double timer\_t) //计时器的开始

{

timerID = startTimer(timer\_t);

}

void Graph::stopTime() //计时器的结束

{

killTimer(timerID);

}

void Graph::paintEvent(QPaintEvent \*) //用于进行图像的重绘

{

QPainter painter(this); //新建画笔

painter.setPen(Qt::gray); //画笔颜色-表示边框颜色

painter.setBrush(Qt::white); //花刷颜色-表示内部颜色

painter.drawRect(0, 0, 465, 405);

for (int i = 1; i < 4; ++i) { //绘制横向坐标线

painter.drawLine(0, i \* 101, 465, i \* 101);

}

for (int i = 1; i < 5; ++i) { //绘制纵向坐标线

painter.drawLine(i \* 93, 0, i \* 93, 405);

}

painter.setPen(QColor(30, 160, 200)); //设置画笔颜色

int old\_y = 0, new\_y;

for (int i = 0; i < 450; ++i) {

new\_y = (405 / 2) \* (1 - graph\_data[(graph\_space + i) % 450]); //计算当前数值在位图中的像素位置

painter.drawLine(i - 1, old\_y, i, new\_y); //线的绘制是从前点至后点

old\_y = new\_y;

}

}

void Graph::timerEvent(QTimerEvent \*) //用于触发计时器的操作

{

graph\_data[graph\_space] = paragraph\_voice / qPow(2, SIZE); //取出数据存放到位图中

graph\_space++; //位标后移

update(); //更新位图

}

Graph.h

#ifndef GRAPH\_H

#define GRAPH\_H

#include "voice.h"

#include <QWidget>

#include <qpainter.h> //涉及到图像的更新操作

#include <QtCore/qmath.h> //用于支持qpow

class Graph : public QWidget //本类用于描述波形的变化-显示类

{

Q\_OBJECT

public:

explicit Graph(QWidget \*parent = 0);

void startTime(double timer\_t); //计时器的开始

void stopTime(); //计时器的结束

signals:

public slots :

private:

float graph\_data[450]; //描述显示图中的数据的内容（采用队列的方式减少内存消耗）

int graph\_space; //描述当前输出的位于graph\_data中的位置

int timerID; //计时器的ID，用于结束时使用

void paintEvent(QPaintEvent\*); //用于进行图像的重绘

void timerEvent(QTimerEvent\*); //用于触发计时器的操作

};

Main.cpp

#endif // GRAPH\_H

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#define BUFFER\_SIZE 2048

#define HINTS\_SIZE 100

#define GRAMID\_LEN 128

#define FRAME\_LEN 640

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

connect(ui->pushButton\_conc, SIGNAL(clicked()), this, SLOT(ClickButton\_Conc())); //UI函数绑定

connect(ui->pushButton\_start, SIGNAL(clicked()), this, SLOT(ClickButton\_Start())); //UI函数绑定

connect(ui->pushButton\_stop, SIGNAL(clicked()), this, SLOT(ClickButton\_Stop())); //UI函数绑定

connect(ui->pushButton\_play, SIGNAL(clicked()), this, SLOT(ClickButton\_Play())); //UI函数绑定

connect(ui->pushButton\_reco, SIGNAL(clicked()), this, SLOT(ClickButton\_Reco())); //UI函数绑定

control\_state = false;

paragraph\_voice = 0;

drive\_ready = false;

QTextCodec::setCodecForTr(QTextCodec::codecForName("UTF-8")); //使得本软件支持中文

QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8")); //使得本软件支持中文

QTextCodec::setCodecForCStrings(QTextCodec::codecForName("UTF-8")); //使得本软件支持中文

}

MainWindow::~MainWindow()

{

::close(voice\_fp);

::close(play\_fp);

delete ui;

}

void MainWindow::paintEvent(QPaintEvent \*)

{

}

bool MainWindow::connectDrive() //连接设备

{

int status;

if (OpenDrive(&voice\_fp, &play\_fp) == false) {

QMessageBox::critical(this, tr("Error"), tr("File open failed !"), QMessageBox::Yes);

return false;

}

if ((status = SetDrive(&voice\_fp, &play\_fp)) == -1) {

QMessageBox::critical(this, tr("Error"), tr("File set failed !"), QMessageBox::Yes);

return false;

}

else if (status == 0) {

QMessageBox::StandardButton rebutton =

QMessageBox::warning(this, tr("Warning"), tr("Some set defferent !"), QMessageBox::Yes | QMessageBox::Ignore);

if (rebutton == QMessageBox::Yes) {

return false;

}

}

readThread.setAudio(voice\_fp);

playThread.setAudio(play\_fp);

return true;

}

void MainWindow::createVoice() //创建语音

{

int dataLen = speech\_voice.size() \* BUFSIZE \* sizeof(ElemType);

Wav\_file default\_wavfile = {

{ 'R', 'I', 'F', 'F' },

dataLen + sizeof(Wav\_file) - 8,

{'W', 'A', 'V', 'E'},

{'f', 'm', 't', ' '},

16,

1,

CHANNELS,

RATE,

RATE \* SIZE / 8,

SIZE / 8,

SIZE,

{'d', 'a', 't', 'a'},

dataLen

};

FILE\* fp = NULL;

if ((fp = fopen("wav/upload.wav", "wb")) == NULL) {

QMessageBox::critical(this, tr("Error"), tr("File open failed !"), QMessageBox::Yes);

return;

}

fwrite(&default\_wavfile, sizeof(Wav\_file), 1, fp);

int temp = dataLen + sizeof(Wav\_file) - 8;

fseek(fp, 4, 0);

fwrite(&temp, sizeof(int), 1, fp);

fseek(fp, 40, 0);

fwrite(&dataLen, sizeof(int), 1, fp);

for (int i = 0; i < speech\_voice.size(); ++i) {

fwrite(speech\_voice[i], BUFSIZE \* sizeof(ElemType), 1, fp);

}

fclose(fp);

}

void MainWindow::ClickButton\_Conc() //连接硬件按钮响应函数

{

if (connectDrive()) {

ui->pushButton\_conc->setEnabled(false); //若连接完设备则按钮无用

drive\_ready = true;

}

else {

drive\_ready = false;

}

}

void MainWindow::ClickButton\_Start() //开始录音按钮响应函数

{

if (!drive\_ready) {

QMessageBox::warning(this, tr("Warning"), tr("You should connect the audio !"), QMessageBox::Yes);

return;

}

if (control\_state == true) {

return;

}

ui->graph->startTime(GRAPH\_SPEED); //图像开始刷新

readThread.start(); //开始录音

control\_state = true;

}

void MainWindow::ClickButton\_Stop() //停止录音按钮响应函数

{

if (!drive\_ready) {

QMessageBox::warning(this, tr("Warning"), tr("You should connect the audio !"), QMessageBox::Yes);

return;

}

if (control\_state == false) {

return;

}

readThread.stop(); //录音结束

ui->graph->stopTime(); //图像停止刷新

control\_state = false;

createVoice();

}

void MainWindow::ClickButton\_Play() //播放录音按钮响应函数

{

if (!drive\_ready) {

QMessageBox::warning(this, tr("Warning"), tr("You should connect the audio !"), QMessageBox::Yes);

return;

}

playThread.start();

}

void MainWindow::ClickButton\_Reco() //识别录音按钮响应函数

{

if (!drive\_ready) {

QMessageBox::warning(this, tr("Warning"), tr("You should connect the audio !"), QMessageBox::Yes);

return;

}

bool reco\_f = true;

char\* result = NULL;

if (recognitiononlineThread.isLogin == false) {

if ((reco\_f = recognitiononlineThread.doLogin(NULL, NULL)) == false) {

while (!recognitiononlineThread.errorInfo.empty()) {

QMessageBox::critical(this, tr("Error"), recognitiononlineThread.errorInfo.pop(), QMessageBox::Yes);

}

return;

}

}

if ((reco\_f = recognitiononlineThread.uploadData("prammar.abnf", "dtt = abnf, sub = asr")) == false) {

while (!recognitiononlineThread.errorInfo.empty()) {

QMessageBox::critical(this, tr("Error"), recognitiononlineThread.errorInfo.pop(), QMessageBox::Yes);

}

return;

}

else {

ui->textBrowser\_gram->clear();

ui->textBrowser\_gram->append(recognitiononlineThread.getGrammarId());

}

result = recognitiononlineThread.recognition("wav/upload.wav", "sub = asr, result\_type = plain, result\_encoding = utf8");

if (result == NULL) {

while (!recognitiononlineThread.errorInfo.empty()) {

QMessageBox::critical(this, tr("Error"), recognitiononlineThread.errorInfo.pop(), QMessageBox::Yes);

}

return;

}

ui->textBrowser\_resu->clear();

ui->textBrowser\_resu->append(recognitiononlineThread.getTopResult());

}

Mainwindows.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QtGui>

#include "readaudio.h"

#include "playaudio.h"

#include <QMainWindow>

#include "recognitiononline.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

private:

Ui::MainWindow \*ui;

void paintEvent(QPaintEvent\*); //重绘函数-用户嵌入式设备全屏使用

QPainter \*paint;

bool control\_state; //标识当前是否处于录完音状态

int voice\_fp; //录音文件ID

int play\_fp; //播放文件ID

bool drive\_ready; //标识设备是否已经准备完成

ReadAudio readThread; //录音线程

PlayAudio playThread; //播放线程

RecognitionOnline recognitiononlineThread; //识别线程

bool connectDrive(); //连接设备

void createVoice(); //创建语音

public slots:

void ClickButton\_Conc(); //连接硬件按钮响应函数

void ClickButton\_Start(); //开始录音按钮响应函数

void ClickButton\_Stop(); //停止录音按钮响应函数

void ClickButton\_Play(); //播放录音按钮响应函数

void ClickButton\_Reco(); //识别录音按钮响应函数

};

#endif // MAINWINDOW\_H

Mainwindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>551</height>

</rect>

</property>

<property name="windowTitle">

<string>Medical Speech Recognition Demo</string>

</property>

<widget class="QWidget" name="centralWidget">

<widget class="QLabel" name="label\_y">

<property name="geometry">

<rect>

<x>20</x>

<y>240</y>

<width>17</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Y</string>

</property>

</widget>

<widget class="QLabel" name="label\_x">

<property name="geometry">

<rect>

<x>260</x>

<y>465</y>

<width>17</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>X</string>

</property>

</widget>

<widget class="QTextBrowser" name="textBrowser\_resu">

<property name="geometry">

<rect>

<x>540</x>

<y>140</y>

<width>221</width>

<height>141</height>

</rect>

</property>

</widget>

<widget class="QLabel" name="label\_text">

<property name="geometry">

<rect>

<x>550</x>

<y>110</y>

<width>131</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Recognition result</string>

</property>

</widget>

<widget class="Graph" name="graph" native="true">

<property name="enabled">

<bool>true</bool>

</property>

<property name="geometry">

<rect>

<x>40</x>

<y>50</y>

<width>470</width>

<height>410</height>

</rect>

</property>

</widget>

<widget class="QLabel" name="label\_graph">

<property name="geometry">

<rect>

<x>50</x>

<y>20</y>

<width>91</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Speech graph</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_start">

<property name="geometry">

<rect>

<x>540</x>

<y>360</y>

<width>99</width>

<height>27</height>

</rect>

</property>

<property name="text">

<string>Start</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_stop">

<property name="geometry">

<rect>

<x>540</x>

<y>410</y>

<width>99</width>

<height>27</height>

</rect>

</property>

<property name="text">

<string>Stop</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_play">

<property name="geometry">

<rect>

<x>660</x>

<y>330</y>

<width>99</width>

<height>27</height>

</rect>

</property>

<property name="text">

<string>Play</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_reco">

<property name="geometry">

<rect>

<x>660</x>

<y>380</y>

<width>99</width>

<height>27</height>

</rect>

</property>

<property name="text">

<string>Recognition</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_conc">

<property name="geometry">

<rect>

<x>540</x>

<y>310</y>

<width>99</width>

<height>27</height>

</rect>

</property>

<property name="text">

<string>Connect</string>

</property>

</widget>

<widget class="QLabel" name="label\_grammar">

<property name="geometry">

<rect>

<x>550</x>

<y>20</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Grammar id</string>

</property>

</widget>

<widget class="QTextBrowser" name="textBrowser\_gram">

<property name="geometry">

<rect>

<x>540</x>

<y>50</y>

<width>221</width>

<height>41</height>

</rect>

</property>

</widget>

<widget class="QLabel" name="label\_supt">

<property name="geometry">

<rect>

<x>620</x>

<y>490</y>

<width>171</width>

<height>21</height>

</rect>

</property>

<property name="text">

<string>语音技术由科大讯飞提供</string>

</property>

</widget>

</widget>

<widget class="QToolBar" name="mainToolBar">

<attribute name="toolBarArea">

<enum>TopToolBarArea</enum>

</attribute>

<attribute name="toolBarBreak">

<bool>false</bool>

</attribute>

</widget>

<widget class="QStatusBar" name="statusBar"/>

<widget class="QMenuBar" name="menuBar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>31</height>

</rect>

</property>

</widget>

</widget>

<layoutdefault spacing="6" margin="11"/>

<customwidgets>

<customwidget>

<class>Graph</class>

<extends>QWidget</extends>

<header location="global">graph.h</header>

<container>1</container>

</customwidget>

</customwidgets>

<resources/>

<connections/>

</ui>

Palyaudio.cpp

#include "playaudio.h"

#include <QMessageBox>

PlayAudio::PlayAudio()

{

stopped = false;

}

void PlayAudio::setAudio(int out) //设置输出文件ID

{

audio\_fp = out;

}

void PlayAudio::run() //播放语音数据

{

int len;

if (!stopped) {

for (int i = 0; i < speech\_voice.size(); ++i) {

len = ::write(audio\_fp, speech\_voice[i], BUFSIZE \* sizeof(ElemType));

if (len != BUFSIZE \* sizeof(ElemType)) {

//Todo : Showing warning information !

}

}

stopped = true;

}

}

void PlayAudio::stop()

{

stopped = true;

}

Playaudio.h

#ifndef PLAYAUDIO\_H

#define PLAYAUDIO\_H

#include "voice.h"

#include <QThread>

class PlayAudio :public QThread //输出喇叭语音数据-线程

{

Q\_OBJECT

public:

PlayAudio();

void setAudio(int out); //设置输出文件ID

void stop();

protected:

int audio\_fp; //硬件ID

void run(); //播放语音数据

private:

volatile bool stopped;

};

#endif // PLAYAUDIO\_H

Readaudio.cpp

#include "readaudio.h"

ReadAudio::ReadAudio()

{

stopped = false;

}

void ReadAudio::setAudio(int in) //设置读取文件的ID

{

audio\_fp = in;

}

void ReadAudio::run() //读取语音数据操作

{

qDeleteAll(speech\_voice); //清除上一次读取的语音数据

speech\_voice.clear();

ElemType \*graph = NULL;

if (!stopped) {

while (true) {

graph = new ElemType[BUFSIZE];

int len = ::read(audio\_fp, graph, BUFSIZE \* sizeof(ElemType)); //读取语音数据

if (len != BUFSIZE \* sizeof(ElemType)) {

//Todo : Showing warning information !

}

speech\_voice.push\_back(graph);

paragraph\_voice = graph[0];

}

}

stopped = false;

}

void ReadAudio::stop()

{

stopped = true;

}

Readaudio.cpp

#ifndef READAUDIO\_H

#define READAUDIO\_H

#include "voice.h"

#include <QThread>

class ReadAudio :public QThread //读取喇叭语音数据-线程

{

Q\_OBJECT

public:

ReadAudio();

void setAudio(int in); //设置读取文件的ID

void stop();

protected:

int audio\_fp; //喇叭硬件文件ID

void run(); //读取语音数据操作

private:

volatile bool stopped;

};

#endif // READAUDIO\_H

Recognitiononline.cpp

#include <stdio.h>

#include <QLatin1String>

#include "recognitiononline.h"

RecognitionOnline::RecognitionOnline()

{

grammar\_id\_len = 128;

login\_params = "appid = 587de813, work\_dir = ."; //登陆参数，appid表示在线注册的应用ID

grammar\_id = NULL;

result = NULL;

isLogin = false;

}

bool RecognitionOnline::doLogin(const char\* username, const char\* password) //登陆操作

{

int info\_code = MSPLogin(username, password, login\_params);

if (info\_code != MSP\_SUCCESS) { //错误处理

errorInfo.push(QString(QLatin1String("Error : Login failed . ")) + QString::number(info\_code, 10));

return false;

}

isLogin = true;

return true;

}

bool RecognitionOnline::uploadData(const char\* location, const char\* mode) //上传语义

{

FILE\* fp = NULL;

int info\_code = -1;

char\* grammar = NULL;

int grammar\_len = -1;

if ((fp = fopen(location, "rb")) == NULL) { //打开语义文件

errorInfo.push(QString(QLatin1String("Error : Upload data file open failed . ")));

return false;

}

fseek(fp, 0, SEEK\_END); //计算语义文件长度

grammar\_len = ftell(fp);

fseek(fp, 0, SEEK\_SET);

grammar = new char[grammar\_len + 1]; //为语义文件申请内存空间

if (grammar == NULL) {

errorInfo.push(QString(QLatin1String("Error : Allocate upload data file memory failed . ")));

return false;

}

else {

fread(grammar, 1, grammar\_len, fp); //读取语义文件数据

grammar[grammar\_len] = 0;

}

char\* tmp\_grammar\_id = new char[100];

strcpy(tmp\_grammar\_id, MSPUploadData("usergram", grammar, grammar\_len, mode, &info\_code));

if (info\_code != MSP\_SUCCESS) {

errorInfo.push(QString(QLatin1String("Error : Upload data file failed . ")) + QString::number(info\_code, 10));

delete grammar;

return false;

}

grammar\_id = new char[strlen(tmp\_grammar\_id) + 1];

strcpy(grammar\_id, tmp\_grammar\_id);

fclose(fp);

delete grammar;

delete tmp\_grammar\_id;

return true;

}

char\* RecognitionOnline::recognition(const char\* location, const char\* mode) //识别过程

{

FILE\* fp = NULL;

char\* voice = NULL;

int voice\_len = -1;

int info\_code = -1;

char\* session\_id = NULL;

int result\_len = 0;

int audio\_status = MSP\_AUDIO\_SAMPLE\_CONTINUE; //音频进行状态

int endpo\_status = MSP\_EP\_LOOKING\_FOR\_SPEECH; //断点检测状态-from server

int recog\_status = MSP\_REC\_STATUS\_SUCCESS; //语音识别状态

if ((fp = fopen(location, "rb")) == NULL) { //打开语音文件

errorInfo.push(QString(QLatin1String("Error : Wav voice file open failed . ")));

return NULL;

}

fseek(fp, 0, SEEK\_END); //计算语音文件的大小

voice\_len = ftell(fp);

fseek(fp, 0, SEEK\_SET);

voice = new char[voice\_len];

if (voice == NULL) {

errorInfo.push(QString(QLatin1String("Error : Allocate voice file memory failed . ")));

return NULL;

}

fread(voice, 1, voice\_len, fp);

char\* tmp\_session\_id = new char[100];

strcpy(tmp\_session\_id, QISRSessionBegin(grammar\_id, mode, &info\_code)); //开始进行session连接

if (info\_code != MSP\_SUCCESS) {

errorInfo.push(QString(QLatin1String("Error : Recognition online begin failed . ")) + QString::number(info\_code, 10));

delete voice;

delete tmp\_session\_id;

return NULL;

}

session\_id = new char[strlen(tmp\_session\_id) + 1];

strcpy(session\_id, tmp\_session\_id);

delete tmp\_session\_id;

int tmp\_space = 0;

while (true) {

int upload\_len = RATE \* SIZE / 5 / 8;

if (voice\_len < 2 \* upload\_len) { //如果剩下的上传需要一次多一点，那么就一次上传结束

upload\_len = voice\_len;

}

audio\_status = tmp\_space == 0 ? MSP\_AUDIO\_SAMPLE\_FIRST : MSP\_AUDIO\_SAMPLE\_CONTINUE;

info\_code = QISRAudioWrite(session\_id, (const void\*)&voice[tmp\_space], upload\_len, audio\_status, &endpo\_status, &recog\_status);

if (info\_code != MSP\_SUCCESS) {

errorInfo.push(QString(QLatin1String("Error : Recognition online write audio failed . ")) + QString::number(info\_code, 10));

delete voice;

return NULL;

}

tmp\_space += upload\_len;

voice\_len -= upload\_len;

if (endpo\_status == MSP\_EP\_AFTER\_SPEECH || voice\_len == 0) {

break;

}

//usleep(200\*1000); //本人觉得不需要这句

}

info\_code = QISRAudioWrite(session\_id, NULL, 0, MSP\_AUDIO\_SAMPLE\_LAST, &endpo\_status, &recog\_status); //告知服务器上传结束

delete voice;

if (info\_code != MSP\_SUCCESS) {

errorInfo.push(QString(QLatin1String("Error : Recognition online write audio (end) failed . ")) + QString::number(info\_code, 10));

return NULL;

}

result = new char[200];

while (recog\_status != MSP\_REC\_STATUS\_COMPLETE) {

const char\* tmp\_result = QISRGetResult(session\_id, &recog\_status, 0, &info\_code);

if (info\_code != MSP\_SUCCESS) {

errorInfo.push(QString(QLatin1String("Error : Recognition online get result failed . ")) + QString::number(info\_code, 10));

return NULL;

}

if (tmp\_result != NULL) {

unsigned int tmp\_result\_len = strlen(tmp\_result);

result\_len += tmp\_result\_len;

if (result\_len > 200) {

break;

}

strncat(result, tmp\_result, tmp\_result\_len);

}

//usleep(150 \* 1000); //防止频繁占用CPU 本人觉得不需要

}

QISRSessionEnd(session\_id, "info : End recognition online . ");

delete session\_id;

return result;

}

char\* RecognitionOnline::getGrammarId() //获取grammar\_id

{

return grammar\_id;

}

char\* RecognitionOnline::getTopResult() //获取识别度最高的识别结果

{

char\* first = strstr(result, "input="); //去除最前方的乱码

first += 6;

char\* second = strstr(first, "confidence");

char\* tmp\_result = new char[100];

strncpy(tmp\_result, first, (second - first));

return tmp\_result;

}

char\* RecognitionOnline::getResult() //获取识别的所有结果（包括相关格式）

{

return result;

}

Recognitiononline.h

#include <QStack>

#include "voice.h"

#include "xfMSC/qisr.h"

#include "xfMSC/msp\_cmn.h"

#include "xfMSC/msp\_errors.h"

class RecognitionOnline

{

private:

int grammar\_id\_len; //标识语义的grammar\_id长度

char\* login\_params; //登陆的相关参数

char\* grammar\_id; //标识上传语义的ID

char\* result; //识别结果

public:

bool isLogin; //表示是否已经登陆到讯飞

QStack<QString> errorInfo; //识别过程中的错误栈

RecognitionOnline();

bool doLogin(const char\* username, const char\* password); //登陆操作

bool uploadData(const char\* location, const char\* mode); //上传语义

char\* recognition(const char\* location, const char\* mode); //识别过程

char\* getGrammarId(); //获取grammar\_id

char\* getTopResult(); //获取识别度最高的识别结果

char\* getResult(); //获取识别的所有结果（包括相关格式）

};

Voice.cpp

#include "voice.h"

int paragraph\_voice;

QVector<ElemType\*> speech\_voice; //用户说话的语音-以链的方式保存

bool OpenDrive(int \*in) //打开硬件

{

\*in = ::open("/dev/audio1", O\_RDONLY); //audio1是设备中的新增喇叭文件-只读方式打开

if (\*in == -1) {

return false;

}

return true;

}

bool OpenDrive(int \*in, int \*out) //打开硬件

{

\*in = ::open("/dev/audio1", O\_RDONLY); //audio1是设备中的新增喇叭文件-只读方式打开

\*out = ::open("/dev/audio", O\_WRONLY); //audio是设备中的放音喇叭文件-只写方式打开

if (\*in == -1 || \*out == -1) {

return false;

}

return true;

}

int SetDrive(int \*in) //设置硬件控制参数

{

int arg;

bool flag = true;

int status;

arg = SIZE;

status = ioctl(\*in, SOUND\_PCM\_WRITE\_BITS, &arg); //设置设备采样位数

if (status == -1) {

return -1;

}

if (arg != SIZE) {

status = false;

}

arg = CHANNELS;

status = ioctl(\*in, SOUND\_PCM\_WRITE\_CHANNELS, &arg); //设置设备采样声道数

if (status == -1) {

return -1;

}

if (arg != CHANNELS) {

status = false;

}

arg = RATE;

status = ioctl(\*in, SOUND\_PCM\_WRITE\_RATE, &arg); //设置设备采样频率

if (status == -1) {

return -1;

}

if (arg != RATE) {

status = false;

}

if (flag) {

return 1;

}

else {

return 0;

}

}

int SetDrive(int \*in, int \*out) //设置硬件控制参数

{

int f\_in = SetDrive(in);

int f\_out = SetDrive(out);

if (f\_in == -1 || f\_out == -1) {

return -1;

}

else if (f\_in == 0 || f\_out == 0) {

return 0;

}

else {

return 1;

}

}

Voice.h

#ifndef VOICE\_H

#define VOICE\_H

#include <QVector>

#include <fcntl.h> //linux中对于硬件文件的操作支持

#include <unistd.h> //linux中对于硬件文件的操作支持

#include <sys/stat.h> //linux中对于硬件文件的操作支持

#include <sys/ioctl.h> //linux中对于硬件文件的操作支持

#include <sys/types.h> //linux中对于硬件文件的操作支持

#include <sys/soundcard.h>//linux中对于声卡文件的操作支持

#define SIZE 16 //采样位数

#define RATE 16000 //采样频率

#define CHANNELS 1 //声道数

#define BUFSIZE 1600 //（讯飞）单位上传的数据个数

#define GRAPH\_SPEED 100 //图像刷新的时间

typedef short ElemType; //随着SIZE的变化而变化 /\* SIZE type 8 char 16 short 32 int 64 long \*/

typedef struct Wav\_file

{

char id\_RIFF[4]; //RIFF块标志

int fileSize; //文件的总字节数

char id\_WAVE[4]; //WAV标志

char id\_FMT[4]; //格式块标志

int formatLength; //格式块长度，16为正常，18说明有附加信息

short int formatTag; //格式类别，值=1 表示编码方式为PCMu律编码

short int channelsNumber; //声道数

int sampleRate; //每秒的样本数

int secondBytes; //每秒数据的字节数

short int chunkAlign; //采样字节数

short int sampleBits; //采样位数

char id\_DATA[5]; //附加块标志

int dataSize; //数据部分字节数

}Wav\_file;

extern int paragraph\_voice;

extern QVector<ElemType\*> speech\_voice; //用户说话的语音-以链的方式保存

bool OpenDrive(int \*in); //打开硬件

bool OpenDrive(int \*in, int \*out); //打开硬件

int SetDrive(int \*in); //设置硬件控制参数

int SetDrive(int \*in, int \*out); //设置硬件控制参数

#endif // VOICE\_H

下面为讯飞支持头：

/\*\*

\* @file msp\_cmn.h

\* @brief Mobile Speech Platform Common Interface Header File

\*

\* This file contains the quick common programming interface (API) declarations

\* of MSP. Developer can include this file in your project to build applications.

\* For more information, please read the developer guide.

\* Use of this software is subject to certain restrictions and limitations set

\* forth in a license agreement entered into between iFLYTEK, Co,LTD.

\* and the licensee of this software. Please refer to the license

\* agreement for license use rights and restrictions.

\*

\* Copyright (C) 1999 - 2012 by ANHUI USTC iFLYTEK, Co,LTD.

\* All rights reserved.

\*

\* @author Speech Dept. iFLYTEK.

\* @version 1.0

\* @date 2012/09/01

\*

\* @see

\*

\* History:

\* index version date author notes

\* 0 1.0 2012/09/01 MSC40 Create this file

\*/

#ifndef \_\_MSP\_CMN\_H\_\_

#define \_\_MSP\_CMN\_H\_\_

#include "msp\_types.h"

#ifdef \_\_cplusplus

extern "C" {

#endif /\* C++ \*/

//#ifdef MSP\_WCHAR\_SUPPORT

/\*\*

\* @fn Wchar2Mbytes

\* @brief wchar to mbytes

\*

\* User login.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const wchar\_t\* wcstr - [in] Null-terminated source string(wchar\_t \*).

\* @param char\* mbstr - [in] Destination string(char \*).

\* @param int len - [in] The maximum number of bytes that can be stored in the multibyte output string.

\* @see

\*/

char \*Wchar2Mbytes(const wchar\_t\* wcstr);

/\*\*

\* @fn Mbytes2Wchar

\* @brief mbytes to wchar

\*

\* User login.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* mbstr - [in] Null-terminated source string(char \*).

\* @param wchar\_t\* wcstr - [in] Destination string(wchar\_t \*).

\* @param int wlen - [in] The maximum number of multibyte characters to convert.

\* @see

\*/

wchar\_t \*Mbytes2Wchar(const char \*mbstr);

//#endif /\*MSP\_WCHAR\_SUPPORT\*/

/\*\*

\* @fn MSPLogin

\* @brief user login interface

\*

\* User login.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* usr - [in] user name.

\* @param const char\* pwd - [in] password.

\* @param const char\* params - [in] parameters when user login.

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_MSPLogin)(const char\* usr, const char\* pwd, const char\* params);

//#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI MSPLoginW(const wchar\_t\* usr, const wchar\_t\* pwd, const wchar\_t\* params);

typedef int (MSPAPI \*Proc\_MSPLoginW)(const wchar\_t\* usr, const wchar\_t\* pwd, const wchar\_t\* params);

//#endif/\*MSP\_WCHAR\_SUPPORT\*/

/\*\*

\* @fn MSPLogout

\* @brief user logout interface

\*

\* User logout

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @see

\*/

int MSPAPI MSPLogout();

typedef int (MSPAPI \*Proc\_MSPLogout)();

//#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI MSPLogoutW();

typedef int (MSPAPI \*Proc\_MSPLogoutW)();

//#endif/\*MSP\_WCHAR\_SUPPORT\*/

/\*\*

\* @fn MSPUpload

\* @brief Upload User Specific Data

\*

\* Upload data such as user config, custom grammar, etc.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* dataName - [in] data name, should be unique to diff other data.

\* @param const char\* params - [in] parameters about uploading data.

\* @param const char\* dataID - [in] id of the data to be operated.

\* @see

\*/

int MSPAPI MSPUpload( const char\* dataName, const char\* params, const char\* dataID);

typedef int (MSPAPI\* Proc\_MSPUpload)( const char\* dataName, const char\* params, const char\* dataID);

/\*\*

\* @fn MSPDownload

\* @brief Download User Specific Data

\*

\* Download data such as user config, etc.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* params - [in] parameters about data to be downloaded.

\* @see

\*/

typedef int (\*DownloadStatusCB)(int errorCode, long param1, const void \*param2, void \*userData);

typedef int (\*DownloadResultCB)(const void \*data, long dataLen, void \*userData);

int MSPAPI MSPDownload(const char\* dataName, const char\* params, DownloadStatusCB statusCb, DownloadResultCB resultCb, void\*userData);

typedef int (MSPAPI\* Proc\_MSPDownload)(const char\* dataName, const char\* params, DownloadStatusCB statusCb, DownloadResultCB resultCb, void\*userData);

int MSPAPI MSPDownloadW(const wchar\_t\* wdataName, const wchar\_t\* wparams, DownloadStatusCB statusCb, DownloadResultCB resultCb, void\*userData);

typedef int (MSPAPI\* Proc\_MSPDownloadW) (const wchar\_t\* wdataName, const wchar\_t\* wparams, DownloadStatusCB statusCb, DownloadResultCB resultCb, void\*userData);

/\*\*

\* @fn MSPAppendData

\* @brief Append Data.

\*

\* Write data to msc, such as data to be uploaded, searching text, etc.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param void\* data - [in] the data buffer pointer, data could be binary.

\* @param unsigned int dataLen - [in] length of data.

\* @param unsigned int dataStatus - [in] data status, 2: first or continuous, 4: last.

\* @see

\*/

int MSPAPI MSPAppendData(void\* data, unsigned int dataLen, unsigned int dataStatus);

typedef int (MSPAPI\* Proc\_MSPAppendData)(void\* data, unsigned int dataLen, unsigned int dataStatus);

/\*\*

\* @fn MSPGetResult

\* @brief Get Result

\*

\* Get result of uploading, downloading or searching, etc.

\*

\* @return const char\* MSPAPI - Return result of uploading, downloading or searching, etc.

\* @param int\* rsltLen - [out] Length of result returned.

\* @param int\* rsltStatus - [out] Status of result returned.

\* @param int\* errorCode - [out] Return 0 in success, otherwise return error code.

\* @see

\*/

const char\* MSPAPI MSPGetResult(unsigned int\* rsltLen, int\* rsltStatus, int \*errorCode);

typedef const char \* (MSPAPI \*Proc\_MSPGetResult)(unsigned int\* rsltLen, int\* rsltStatus, int \*errorCode);

/\*\*

\* @fn MSPSetParam

\* @brief set params of msc

\*

\* set param of msc

\*

\* @return int - Return 0 if success, otherwise return errcode.

\* @param const char\* paramName - [in] param name.

\* @param const char\* paramValue - [in] param value

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_MSPSetParam)(const char\* paramName, const char\* paramValue);

/\*\*

\* @fn MSPGetParam

\* @brief get params of msc

\*

\* get param of msc

\*

\* @return int - Return 0 if success, otherwise return errcode.

\* @param const char\* paramName - [in] param name.

\* @param const char\* paramValue - [out] param value

\* @param const char\* valueLen - [in/out] param value (buffer) length

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_MSPGetParam)( const char \*paramName, char \*paramValue, unsigned int \*valueLen );

/\*\*

\* @fn MSPUploadData

\* @brief Upload User Specific Data

\*

\* Upload data such as user config, custom grammar, etc.

\*

\* @return const char\* MSPAPI - data id returned by Server, used for special command.

\* @param const char\* dataName - [in] data name, should be unique to diff other data.

\* @param void\* data - [in] the data buffer pointer, data could be binary.

\* @param unsigned int dataLen - [in] length of data.

\* @param const char\* params - [in] parameters about uploading data.

\* @param int\* errorCode - [out] Return 0 in success, otherwise return error code.

\* @see

\*/

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

typedef const char\* (MSPAPI\* Proc\_MSPUploadData)(const char\* dataName, void\* data, unsigned int dataLen, const char\* params, int\* errorCode);

/\*\*

\* @fn MSPDownloadData

\* @brief Download User Specific Data

\*

\* Download data such as user config, etc.

\*

\* @return const void\* MSPAPI - received data buffer pointer, data could be binary, NULL if failed or data does not exsit.

\* @param const char\* params - [in] parameters about data to be downloaded.

\* @param unsigned int\* dataLen - [out] length of received data.

\* @param int\* errorCode - [out] Return 0 in success, otherwise return error code.

\* @see

\*/

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

typedef const void\* (MSPAPI\* Proc\_MSPDownloadData)(const char\* params, unsigned int\* dataLen, int\* errorCode);

//#ifdef MSP\_WCHAR\_SUPPORT

const void\* MSPAPI MSPDownloadDataW(const wchar\_t\* params, unsigned int\* dataLen, int\* errorCode);

typedef const void\* (MSPAPI\* Proc\_MSPDownloadDataW)(const wchar\_t\* params, unsigned int\* dataLen, int\* errorCode);

//#endif/\*MSP\_WCHAR\_SUPPORT\*/

/\*\*

\* @fn MSPSearch

\* @brief Search text for result

\*

\* Search text content, and got text result

\*

\* @return const void\* MSPAPI - received data buffer pointer, data could be binary, NULL if failed or data does not exsit.

\* @param const char\* params - [in] parameters about data to be downloaded.

\* @param unsigned int\* dataLen - [out] length of received data.

\* @param int\* errorCode - [out] Return 0 in success, otherwise return error code.

\* @see

\*/

const char\* MSPAPI MSPSearch(const char\* params, const char\* text, unsigned int\* dataLen, int\* errorCode);

typedef const char\* (MSPAPI\* Proc\_MSPSearch)(const char\* params, const char\* text, unsigned int\* dataLen, int\* errorCode);

typedef int (\*NLPSearchCB)(const char \*sessionID, int errorCode, int status, const void\* result, long rsltLen, void \*userData);

const char\* MSPAPI MSPNlpSearch(const char\* params, const char\* text, unsigned int textLen, int \*errorCode, NLPSearchCB callback, void \*userData);

typedef const char\* (MSPAPI\* Proc\_MSPNlpSearch)(const char\* params, const char\* text, unsigned int textLen, int \*errorCode, NLPSearchCB callback, void \*userData);

int MSPAPI MSPNlpSchCancel(const char \*sessionID, const char \*hints);

/\*\*

\* @fn MSPRegisterNotify

\* @brief Register a Callback

\*

\* Register a Callback

\*

\* @return int -

\* @param msp\_status\_ntf\_handler statusCb - [in] notify handler

\* @param void \*userData - [in] userData

\* @see

\*/

typedef void ( \*msp\_status\_ntf\_handler)( int type, int status, int param1, const void \*param2, void \*userData );

int MSPAPI MSPRegisterNotify( msp\_status\_ntf\_handler statusCb, void \*userData );

typedef const char\* (MSPAPI\* Proc\_MSPRegisterNotify)( msp\_status\_ntf\_handler statusCb, void \*userData );

/\*\*

\* @fn MSPGetVersion

\* @brief Get version of MSC or Local Engine

\*

\* Get version of MSC or Local Engine

\*

\* @return const char \* MSPAPI - Return version value if success, NULL if fail.

\* @param const char \*verName - [in] version name, could be "msc", "aitalk", "aisound", "ivw".

\* @param int \*errorCode - [out] Return 0 in success, otherwise return error code.

\* @see

\*/

const char\* MSPAPI MSPGetVersion(const char \*verName, int \*errorCode);

typedef const char\* (MSPAPI \* Proc\_MSPGetVersion)(const char \*verName, int \*errorCode);

#ifdef \_\_cplusplus

} /\* extern "C" \*/

#endif /\* C++ \*/

#endif /\* \_\_MSP\_CMN\_H\_\_ \*/

#ifndef \_\_MSP\_ERRORS\_H\_\_

#define \_\_MSP\_ERRORS\_H\_\_

#define MSP\_HTTP\_ERROR(x) ((x) + MSP\_ERROR\_HTTP\_BASE )

enum

{

MSP\_SUCCESS = 0,

MSP\_ERROR\_FAIL = -1,

MSP\_ERROR\_EXCEPTION = -2,

/\* General errors 10100(0x2774) \*/

MSP\_ERROR\_GENERAL = 10100, /\* 0x2774 \*/

MSP\_ERROR\_OUT\_OF\_MEMORY = 10101, /\* 0x2775 \*/

MSP\_ERROR\_FILE\_NOT\_FOUND = 10102, /\* 0x2776 \*/

MSP\_ERROR\_NOT\_SUPPORT = 10103, /\* 0x2777 \*/

MSP\_ERROR\_NOT\_IMPLEMENT = 10104, /\* 0x2778 \*/

MSP\_ERROR\_ACCESS = 10105, /\* 0x2779 \*/

MSP\_ERROR\_INVALID\_PARA = 10106, /\* 0x277A \*/ /\* 缺少参数 \*/

MSP\_ERROR\_INVALID\_PARA\_VALUE = 10107, /\* 0x277B \*/ /\* 无效参数值 \*/

MSP\_ERROR\_INVALID\_HANDLE = 10108, /\* 0x277C \*/

MSP\_ERROR\_INVALID\_DATA = 10109, /\* 0x277D \*/

MSP\_ERROR\_NO\_LICENSE = 10110, /\* 0x277E \*/ /\* 引擎授权不足 \*/

MSP\_ERROR\_NOT\_INIT = 10111, /\* 0x277F \*/ /\* 引擎未初始化,可能是引擎崩溃 \*/

MSP\_ERROR\_NULL\_HANDLE = 10112, /\* 0x2780 \*/

MSP\_ERROR\_OVERFLOW = 10113, /\* 0x2781 \*/ /\* 单用户下模型数超上限(10个), \*/

/\* 只出现在测试时对一个用户进行并发注册 \*/

MSP\_ERROR\_TIME\_OUT = 10114, /\* 0x2782 \*/ /\* 超时 \*/

MSP\_ERROR\_OPEN\_FILE = 10115, /\* 0x2783 \*/

MSP\_ERROR\_NOT\_FOUND = 10116, /\* 0x2784 \*/ /\* 数据库中模型不存在 \*/

MSP\_ERROR\_NO\_ENOUGH\_BUFFER = 10117, /\* 0x2785 \*/

MSP\_ERROR\_NO\_DATA = 10118, /\* 0x2786 \*/ /\* 从客户端读音频或从引擎段获取结果时无数据 \*/

MSP\_ERROR\_NO\_MORE\_DATA = 10119, /\* 0x2787 \*/

MSP\_ERROR\_NO\_RESPONSE\_DATA = 10120, /\* 0x2788 \*/

MSP\_ERROR\_ALREADY\_EXIST = 10121, /\* 0x2789 \*/ /\* 数据库中模型已存在 \*/

MSP\_ERROR\_LOAD\_MODULE = 10122, /\* 0x278A \*/

MSP\_ERROR\_BUSY = 10123, /\* 0x278B \*/

MSP\_ERROR\_INVALID\_CONFIG = 10124, /\* 0x278C \*/

MSP\_ERROR\_VERSION\_CHECK = 10125, /\* 0x278D \*/

MSP\_ERROR\_CANCELED = 10126, /\* 0x278E \*/

MSP\_ERROR\_INVALID\_MEDIA\_TYPE = 10127, /\* 0x278F \*/

MSP\_ERROR\_CONFIG\_INITIALIZE = 10128, /\* 0x2790 \*/

MSP\_ERROR\_CREATE\_HANDLE = 10129, /\* 0x2791 \*/

MSP\_ERROR\_CODING\_LIB\_NOT\_LOAD = 10130, /\* 0x2792 \*/

MSP\_ERROR\_USER\_CANCELLED = 10131, /\* 0x2793 \*/

MSP\_ERROR\_INVALID\_OPERATION = 10132, /\* 0x2794 \*/

MSP\_ERROR\_MESSAGE\_NOT\_COMPLETE = 10133, /\* 0x2795 \*/ /\* flash \*/

MSP\_ERROR\_NO\_EID = 10134, /\* 0x2795 \*/

MSP\_ERROE\_OVER\_REQ = 10135, /\* 0x2797 \*/ /\* client Redundancy request \*/

MSP\_ERROR\_USER\_ACTIVE\_ABORT = 10136, /\* 0x2798 \*/ /\* user abort \*/

MSP\_ERROR\_BUSY\_GRMBUILDING = 10137, /\* 0x2799 \*/

MSP\_ERROR\_BUSY\_LEXUPDATING = 10138, /\* 0x279A \*/

MSP\_ERROR\_SESSION\_RESET = 10139, /\* 0x279B \*/ /\* msc主动终止会话，准备重传 \*/

MSP\_ERROR\_BOS\_TIMEOUT = 10140, /\* 0x279C \*/ /\* VAD前端点超时 \*/

MSP\_ERROR\_STREAM\_FILTER = 10141, /\* 0X279D \*/ /\* AIUI当前Stream被过滤 \*/

/\* Error codes of network 10200(0x27D8)\*/

MSP\_ERROR\_NET\_GENERAL = 10200, /\* 0x27D8 \*/

MSP\_ERROR\_NET\_OPENSOCK = 10201, /\* 0x27D9 \*/ /\* Open socket \*/

MSP\_ERROR\_NET\_CONNECTSOCK = 10202, /\* 0x27DA \*/ /\* Connect socket \*/

MSP\_ERROR\_NET\_ACCEPTSOCK = 10203, /\* 0x27DB \*/ /\* Accept socket \*/

MSP\_ERROR\_NET\_SENDSOCK = 10204, /\* 0x27DC \*/ /\* Send socket data \*/

MSP\_ERROR\_NET\_RECVSOCK = 10205, /\* 0x27DD \*/ /\* Recv socket data \*/

MSP\_ERROR\_NET\_INVALIDSOCK = 10206, /\* 0x27DE \*/ /\* Invalid socket handle \*/

MSP\_ERROR\_NET\_BADADDRESS = 10207, /\* 0x27EF \*/ /\* Bad network address \*/

MSP\_ERROR\_NET\_BINDSEQUENCE = 10208, /\* 0x27E0 \*/ /\* Bind after listen/connect \*/

MSP\_ERROR\_NET\_NOTOPENSOCK = 10209, /\* 0x27E1 \*/ /\* Socket is not opened \*/

MSP\_ERROR\_NET\_NOTBIND = 10210, /\* 0x27E2 \*/ /\* Socket is not bind to an address \*/

MSP\_ERROR\_NET\_NOTLISTEN = 10211, /\* 0x27E3 \*/ /\* Socket is not listening \*/

MSP\_ERROR\_NET\_CONNECTCLOSE = 10212, /\* 0x27E4 \*/ /\* The other side of connection is closed \*/

MSP\_ERROR\_NET\_NOTDGRAMSOCK = 10213, /\* 0x27E5 \*/ /\* The socket is not datagram type \*/

MSP\_ERROR\_NET\_DNS = 10214, /\* 0x27E6 \*/ /\* domain name is invalid or dns server does not function well \*/

MSP\_ERROR\_NET\_INIT = 10215, /\* 0x27E7 \*/ /\* ssl ctx create failed \*/

/\*nfl error\*/

MSP\_ERROR\_NFL\_INNER\_ERROR = 10216, /\* NFL inner error \*/

MSP\_ERROR\_MSS\_TIME\_OUT = 10217, /\* MSS TIMEOUT \*/

MSP\_ERROT\_CLIENT\_TIME\_OUT = 10218, /\* CLIENT TIMEOUT \*/

MSP\_ERROR\_CLIENT\_CLOSE = 10219, /\* CLIENT CLOSED CONNECTION \*/

MSP\_ERROR\_CLIENT\_AREA\_CHANGE = 10220,

MSP\_ERROR\_NET\_SSL\_HANDSHAKE = 10221,

MSP\_ERROR\_NET\_INVALID\_ROOT\_CERT = 10222,

MSP\_ERROR\_NET\_INVALID\_CLIENT\_CERT = 10223,

MSP\_ERROR\_NET\_INVALID\_SERVER\_CERT = 10224,

MSP\_ERROR\_NET\_INVALID\_KEY = 10225,

MSP\_ERROR\_NET\_CERT\_VERIFY\_FAILED = 10226,

/\* Error codes of mssp message 10300(0x283C) \*/

MSP\_ERROR\_MSG\_GENERAL = 10300, /\* 0x283C \*/

MSP\_ERROR\_MSG\_PARSE\_ERROR = 10301, /\* 0x283D \*/

MSP\_ERROR\_MSG\_BUILD\_ERROR = 10302, /\* 0x283E \*/

MSP\_ERROR\_MSG\_PARAM\_ERROR = 10303, /\* 0x283F \*/

MSP\_ERROR\_MSG\_CONTENT\_EMPTY = 10304, /\* 0x2840 \*/

MSP\_ERROR\_MSG\_INVALID\_CONTENT\_TYPE = 10305, /\* 0x2841 \*/

MSP\_ERROR\_MSG\_INVALID\_CONTENT\_LENGTH = 10306, /\* 0x2842 \*/

MSP\_ERROR\_MSG\_INVALID\_CONTENT\_ENCODE = 10307, /\* 0x2843 \*/

MSP\_ERROR\_MSG\_INVALID\_KEY = 10308, /\* 0x2844 \*/

MSP\_ERROR\_MSG\_KEY\_EMPTY = 10309, /\* 0x2845 \*/

MSP\_ERROR\_MSG\_SESSION\_ID\_EMPTY = 10310, /\* 0x2846 \*/ /\* 会话ID为空 \*/

MSP\_ERROR\_MSG\_LOGIN\_ID\_EMPTY = 10311, /\* 0x2847 \*/ /\* 音频序列ID为空 \*/

MSP\_ERROR\_MSG\_SYNC\_ID\_EMPTY = 10312, /\* 0x2848 \*/

MSP\_ERROR\_MSG\_APP\_ID\_EMPTY = 10313, /\* 0x2849 \*/

MSP\_ERROR\_MSG\_EXTERN\_ID\_EMPTY = 10314, /\* 0x284A \*/

MSP\_ERROR\_MSG\_INVALID\_CMD = 10315, /\* 0x284B \*/

MSP\_ERROR\_MSG\_INVALID\_SUBJECT = 10316, /\* 0x284C \*/

MSP\_ERROR\_MSG\_INVALID\_VERSION = 10317, /\* 0x284D \*/

MSP\_ERROR\_MSG\_NO\_CMD = 10318, /\* 0x284E \*/

MSP\_ERROR\_MSG\_NO\_SUBJECT = 10319, /\* 0x284F \*/

MSP\_ERROR\_MSG\_NO\_VERSION = 10320, /\* 0x2850 \*/

MSP\_ERROR\_MSG\_MSSP\_EMPTY = 10321, /\* 0x2851 \*/

MSP\_ERROR\_MSG\_NEW\_RESPONSE = 10322, /\* 0x2852 \*/

MSP\_ERROR\_MSG\_NEW\_CONTENT = 10323, /\* 0x2853 \*/

MSP\_ERROR\_MSG\_INVALID\_SESSION\_ID = 10324, /\* 0x2854 \*/ /\* 无效的会话ID(sid) \*/

MSP\_ERROR\_MSG\_INVALID\_CONTENT = 10325, /\* 0x2855 \*/

/\* Error codes of DataBase 10400(0x28A0)\*/

MSP\_ERROR\_DB\_GENERAL = 10400, /\* 0x28A0 \*/ /\* 数据库异常 \*/

MSP\_ERROR\_DB\_EXCEPTION = 10401, /\* 0x28A1 \*/

MSP\_ERROR\_DB\_NO\_RESULT = 10402, /\* 0x28A2 \*/ /\* redis中没有找到会话ID(sid) \*/

MSP\_ERROR\_DB\_INVALID\_USER = 10403, /\* 0x28A3 \*/

MSP\_ERROR\_DB\_INVALID\_PWD = 10404, /\* 0x28A4 \*/

MSP\_ERROR\_DB\_CONNECT = 10405, /\* 0x28A5 \*/

MSP\_ERROR\_DB\_INVALID\_SQL = 10406, /\* 0x28A6 \*/

MSP\_ERROR\_DB\_INVALID\_APPID = 10407, /\* 0x28A7 \*/

MSP\_ERROR\_DB\_NO\_UID = 10408,

/\* Error codes of Resource 10500(0x2904)\*/

MSP\_ERROR\_RES\_GENERAL = 10500, /\* 0x2904 \*/

MSP\_ERROR\_RES\_LOAD = 10501, /\* 0x2905 \*/ /\* Load resource \*/

MSP\_ERROR\_RES\_FREE = 10502, /\* 0x2906 \*/ /\* Free resource \*/

MSP\_ERROR\_RES\_MISSING = 10503, /\* 0x2907 \*/ /\* Resource File Missing \*/

MSP\_ERROR\_RES\_INVALID\_NAME = 10504, /\* 0x2908 \*/ /\* Invalid resource file name \*/

MSP\_ERROR\_RES\_INVALID\_ID = 10505, /\* 0x2909 \*/ /\* Invalid resource ID \*/

MSP\_ERROR\_RES\_INVALID\_IMG = 10506, /\* 0x290A \*/ /\* Invalid resource image pointer \*/

MSP\_ERROR\_RES\_WRITE = 10507, /\* 0x290B \*/ /\* Write read-only resource \*/

MSP\_ERROR\_RES\_LEAK = 10508, /\* 0x290C \*/ /\* Resource leak out \*/

MSP\_ERROR\_RES\_HEAD = 10509, /\* 0x290D \*/ /\* Resource head currupt \*/

MSP\_ERROR\_RES\_DATA = 10510, /\* 0x290E \*/ /\* Resource data currupt \*/

MSP\_ERROR\_RES\_SKIP = 10511, /\* 0x290F \*/ /\* Resource file skipped \*/

/\* Error codes of TTS 10600(0x2968)\*/

MSP\_ERROR\_TTS\_GENERAL = 10600, /\* 0x2968 \*/

MSP\_ERROR\_TTS\_TEXTEND = 10601, /\* 0x2969 \*/ /\* Meet text end \*/

MSP\_ERROR\_TTS\_TEXT\_EMPTY = 10602, /\* 0x296A \*/ /\* no synth text \*/

MSP\_ERROR\_TTS\_LTTS\_ERROR = 10603, /\* 0x296B \*/

/\* Error codes of Recognizer 10700(0x29CC) \*/

MSP\_ERROR\_REC\_GENERAL = 10700, /\* 0x29CC \*/ /\* 引擎异常 \*/

MSP\_ERROR\_REC\_INACTIVE = 10701, /\* 0x29CD \*/

MSP\_ERROR\_REC\_GRAMMAR\_ERROR = 10702, /\* 0x29CE \*/

MSP\_ERROR\_REC\_NO\_ACTIVE\_GRAMMARS = 10703, /\* 0x29CF \*/

MSP\_ERROR\_REC\_DUPLICATE\_GRAMMAR = 10704, /\* 0x29D0 \*/

MSP\_ERROR\_REC\_INVALID\_MEDIA\_TYPE = 10705, /\* 0x29D1 \*/

MSP\_ERROR\_REC\_INVALID\_LANGUAGE = 10706, /\* 0x29D2 \*/

MSP\_ERROR\_REC\_URI\_NOT\_FOUND = 10707, /\* 0x29D3 \*/

MSP\_ERROR\_REC\_URI\_TIMEOUT = 10708, /\* 0x29D4 \*/

MSP\_ERROR\_REC\_URI\_FETCH\_ERROR = 10709, /\* 0x29D5 \*/

MSP\_ERROR\_REC\_PROC\_MOD = 10710, /\* 0x29D6 \*/

/\* Error codes of Speech Detector 10800(0x2A30) \*/

MSP\_ERROR\_EP\_GENERAL = 10800, /\* 0x2A30 \*/

MSP\_ERROR\_EP\_NO\_SESSION\_NAME = 10801, /\* 0x2A31 \*/

MSP\_ERROR\_EP\_INACTIVE = 10802, /\* 0x2A32 \*/

MSP\_ERROR\_EP\_INITIALIZED = 10803, /\* 0x2A33 \*/

/\* Error codes of TUV \*/

MSP\_ERROR\_TUV\_GENERAL = 10900, /\* 0x2A94 \*/

MSP\_ERROR\_TUV\_GETHIDPARAM = 10901, /\* 0x2A95 \*/ /\* Get Busin Param huanid\*/

MSP\_ERROR\_TUV\_TOKEN = 10902, /\* 0x2A96 \*/ /\* Get Token \*/

MSP\_ERROR\_TUV\_CFGFILE = 10903, /\* 0x2A97 \*/ /\* Open cfg file \*/

MSP\_ERROR\_TUV\_RECV\_CONTENT = 10904, /\* 0x2A98 \*/ /\* received content is error \*/

MSP\_ERROR\_TUV\_VERFAIL = 10905, /\* 0x2A99 \*/ /\* Verify failure \*/

/\* Error codes of IMTV \*/

MSP\_ERROR\_LOGIN\_SUCCESS = 11000, /\* 0x2AF8 \*/ /\* 成功 \*/

MSP\_ERROR\_LOGIN\_NO\_LICENSE = 11001, /\* 0x2AF9 \*/ /\* 试用次数结束，用户需要付费 \*/

MSP\_ERROR\_LOGIN\_SESSIONID\_INVALID = 11002, /\* 0x2AFA \*/ /\* SessionId失效，需要重新登录通行证 \*/

MSP\_ERROR\_LOGIN\_SESSIONID\_ERROR = 11003, /\* 0x2AFB \*/ /\* SessionId为空，或者非法 \*/

MSP\_ERROR\_LOGIN\_UNLOGIN = 11004, /\* 0x2AFC \*/ /\* 未登录通行证 \*/

MSP\_ERROR\_LOGIN\_INVALID\_USER = 11005, /\* 0x2AFD \*/ /\* 用户ID无效 \*/

MSP\_ERROR\_LOGIN\_INVALID\_PWD = 11006, /\* 0x2AFE \*/ /\* 用户密码无效 \*/

MSP\_ERROR\_LOGIN\_SYSTEM\_ERROR = 11099, /\* 0x2B5B \*/ /\* 系统错误 \*/

/\* 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 using in local engine \*/

MSP\_ERROR\_AUTH\_NO\_LICENSE = 11200, /\* 0x2BC0 \*/ /\* 无授权 \*/

MSP\_ERROR\_AUTH\_NO\_ENOUGH\_LICENSE = 11201, /\* 0x2BC1 \*/ /\* 授权不足 \*/

MSP\_ERROR\_AUTH\_INVALID\_LICENSE = 11202, /\* 0x2BC2 \*/ /\* 无效的授权 \*/

MSP\_ERROR\_AUTH\_LICENSE\_EXPIRED = 11203, /\* 0x2BC3 \*/ /\* 授权过期 \*/

MSP\_ERROR\_AUTH\_NEED\_MORE\_DATA = 11204, /\* 0x2BC4 \*/ /\* 无设备信息 \*/

MSP\_ERROR\_AUTH\_LICENSE\_TO\_BE\_EXPIRED = 11205, /\* 0x2BC5 \*/ /\* 授权即将过期，警告性错误码 \*/

MSP\_ERROR\_AUTH\_INVALID\_MACHINE\_ID = 11206, /\* 0x2BC6 \*/ /\* 无效的机器码 \*/

MSP\_ERROR\_AUTH\_LOCAL\_ASR\_FORBIDDEN = 11207, /\* 0x2BC7 \*/ /\* 禁止使用本地识别引擎 \*/

MSP\_ERROR\_AUTH\_LOCAL\_TTS\_FORBIDDEN = 11208, /\* 0x2BC8 \*/ /\* 禁止使用本地合成引擎 \*/

MSP\_ERROR\_AUTH\_LOCAL\_IVW\_FORBIDDEN = 11209, /\* 0x2BC9 \*/ /\* 禁止使用本地唤醒引擎 \*/

MSP\_ERROR\_AUTH\_APPID\_NOT\_MATCH = 11210, /\* 0x2BCA \*/ /\* 资源appid和应用appid不匹配 \*/

MSP\_ERROR\_AUTH\_UID\_NOT\_MATCH = 11211, /\* 0x2BCB \*/ /\* 资源uid和登录用户uid不匹配 \*/

MSP\_ERROR\_AUTH\_TRIAL\_EXPIRED = 11212, /\* 0x2BCC \*/ /\* 试用资源过期 \*/

MSP\_ERROR\_AUTH\_LOCAL\_IFD\_FORBIDDEN = 11213, /\* 0x2BC9 \*/ /\* 禁止使用本地人脸引擎 \*/

/\*Error Codes of Authorization\*/

MSP\_ERROR\_AUTH\_DVC\_NO\_LICENSE = 11300,

MSP\_ERROR\_AUTH\_DVC\_NO\_ENOUGH\_LICENSE = 11301,

MSP\_ERROR\_AUTH\_DVC\_INVALID\_LICENSE = 11302,

MSP\_ERROR\_AUTH\_DVC\_LICENSE\_EXPIRED = 11303,

MSP\_ERROR\_AUTH\_DVC\_NEED\_MORE\_DATA = 11304,

MSP\_ERROR\_AUTH\_DVC\_LICENSE\_TO\_BE\_EXPIRED = 11305,

MSP\_ERROR\_AUTH\_DVC\_EXCEED\_LICENSE = 11306,

/\* Error codes of Ise \*/

MSP\_ERROR\_ASE\_EXCEP\_SILENCE = 11401,

MSP\_ERROR\_ASE\_EXCEP\_SNRATIO = 11402,

MSP\_ERROR\_ASE\_EXCEP\_PAPERDATA = 11403,

MSP\_ERROR\_ASE\_EXCEP\_PAPERCONTENTS = 11404,

MSP\_ERROR\_ASE\_EXCEP\_NOTMONO = 11405,

MSP\_ERROR\_ASE\_EXCEP\_OTHERS = 11406,

MSP\_ERROR\_ASE\_EXCEP\_PAPERFMT = 11407,

MSP\_ERROR\_ASE\_EXCEP\_ULISTWORD = 11408,

/\* Error codes of IVP \*/

MSP\_ERROR\_IVP\_GENERAL = 11600, // 内核异常

MSP\_ERROR\_IVP\_EXTRA\_RGN\_SOPPORT = 11601, // 注册时向引擎所写音频条数超过上限(9次)

MSP\_ERROR\_IVP\_TRUNCATED = 11602, // 音频截幅(因信号波形的幅度太大，而超出系统的线性范围)，如记录尖叫声的音频

MSP\_ERROR\_IVP\_MUCH\_NOISE = 11603, // 音频信噪比过低

MSP\_ERROR\_IVP\_TOO\_LOW = 11604, // 音频能量过低

MSP\_ERROR\_IVP\_ZERO\_AUDIO = 11605, // 无音频

MSP\_ERROR\_IVP\_UTTER\_TOO\_SHORT = 11606, // 音频太短

MSP\_ERROR\_IVP\_TEXT\_NOT\_MATCH = 11607, // 1.音频和文本不匹配，常见原因1.抢读(在按下录音键之前读)

// 2.录音机的启动电流被录入表现在音频上是在音频首有冲击电流 3.确实不匹配"

MSP\_ERROR\_IVP\_NO\_ENOUGH\_AUDIO = 11608, // 音频不够，注册自由说，而写入的音频又不够长时会报，告诉调用者继续传音频

MSP\_ERROR\_IVP\_MODEL\_NOT\_FOUND\_IN\_HBASE = 11610, // 模型在hbase中没找到

/\* Error codes of Face \*/

MSP\_ERROR\_IFR\_NOT\_FACE\_IMAGE = 11700, // 【无人脸，对应的引擎错误码是20200 】

MSP\_ERROR\_FACE\_IMAGE\_FULL\_LEFT = 11701, // 【人脸向左，对应的引擎错误码是20201】

MSP\_ERROR\_FACE\_IMAGE\_FULL\_RIGHT = 11702, // 【人脸向右，对应的引擎错误码是20202】

MSP\_ERROR\_IMAGE\_CLOCKWISE\_WHIRL = 11703, // 【顺时针旋转，对应的引擎错误码是20203】

MSP\_ERROR\_IMAGE\_COUNTET\_CLOCKWISE\_WHIRL = 11704, // 【逆时针旋转，对应的引擎错误码是20204】

MSP\_ERROR\_VALID\_IMAGE\_SIZE = 11705, // 【图片大小异常 ，对应的引擎错误码是20205】

MSP\_ERROR\_ILLUMINATION = 11706, // 【光照异常，对应的引擎错误码是20206】

MSP\_ERROR\_FACE\_OCCULTATION = 11707, // 【人脸被遮挡，对应的引擎错误码是20207】

MSP\_ERROR\_FACE\_INVALID\_MODEL = 11708, // 【非法模型数据，对应的引擎错误码是20208】

MSP\_ERROR\_FUSION\_INVALID\_INPUT\_TYPE = 11709, // 【输入数据类型非法，对应的引擎错误码是20300】

MSP\_ERROR\_FUSION\_NO\_ENOUGH\_DATA = 11710, // 【输入的数据不完整，对应的引擎错误码是20301】

MSP\_ERROR\_FUSION\_ENOUGH\_DATA = 11711, // 【输入的数据过多，对应的引擎错误码是20302】

/\*Error Codes of AIUI\*/

MSP\_ERROR\_AIUI\_CID\_EXPIRED = 11800,

/\* Error codes of http 12000(0x2EE0) \*/

MSP\_ERROR\_HTTP\_BASE = 12000, /\* 0x2EE0 \*/

MSP\_ERROR\_HTTP\_400 = 12400,

MSP\_ERROR\_HTTP\_401 = 12401,

MSP\_ERROR\_HTTP\_402 = 12402,

MSP\_ERROR\_HTTP\_403 = 12403,

MSP\_ERROR\_HTTP\_404 = 12404,

MSP\_ERROR\_HTTP\_405 = 12405,

MSP\_ERROR\_HTTP\_406 = 12406,

MSP\_ERROR\_HTTP\_407 = 12407,

MSP\_ERROR\_HTTP\_408 = 12408,

MSP\_ERROR\_HTTP\_409 = 12409,

MSP\_ERROR\_HTTP\_410 = 12410,

MSP\_ERROR\_HTTP\_411 = 12411,

MSP\_ERROR\_HTTP\_412 = 12412,

MSP\_ERROR\_HTTP\_413 = 12413,

MSP\_ERROR\_HTTP\_414 = 12414,

MSP\_ERROR\_HTTP\_415 = 12415,

MSP\_ERROR\_HTTP\_416 = 12416,

MSP\_ERROR\_HTTP\_417 = 12417,

MSP\_ERROR\_HTTP\_500 = 12500,

MSP\_ERROR\_HTTP\_501 = 12501,

MSP\_ERROR\_HTTP\_502 = 12502,

MSP\_ERROR\_HTTP\_503 = 12503,

MSP\_ERROR\_HTTP\_504 = 12504,

MSP\_ERROR\_HTTP\_505 = 12505,

/\*Error codes of ISV \*/

MSP\_ERROR\_ISV\_NO\_USER = 13000, /\* 32C8 \*/ /\* the user doesn't exist \*/

/\* Error codes of Lua scripts \*/

MSP\_ERROR\_LUA\_BASE = 14000, /\* 0x36B0 \*/

MSP\_ERROR\_LUA\_YIELD = 14001, /\* 0x36B1 \*/

MSP\_ERROR\_LUA\_ERRRUN = 14002, /\* 0x36B2 \*/

MSP\_ERROR\_LUA\_ERRSYNTAX = 14003, /\* 0x36B3 \*/

MSP\_ERROR\_LUA\_ERRMEM = 14004, /\* 0x36B4 \*/

MSP\_ERROR\_LUA\_ERRERR = 14005, /\* 0x36B5 \*/

MSP\_ERROR\_LUA\_INVALID\_PARAM = 14006, /\* 0x36B6 \*/

/\* Error codes of MMP \*/

MSP\_ERROR\_MMP\_BASE = 15000, /\* 0x3A98 \*/

MSP\_ERROR\_MMP\_MYSQL\_INITFAIL = 15001, /\* 0x3A99 \*/

MSP\_ERROR\_MMP\_REDIS\_INITFAIL = 15002, /\* 0x3A9A \*/

MSP\_ERROR\_MMP\_NETDSS\_INITFAIL = 15003, /\* 0x3A9B \*/

MSP\_ERROR\_MMP\_TAIR\_INITFAIL = 15004, /\* 0x3A9C \*/

MSP\_ERROR\_MMP\_MAIL\_SESSION\_FAIL = 15006, /\* 0x3A9E \*/ /\* 邮件登陆服务器时，会话错误。\*/

MSP\_ERROR\_MMP\_MAIL\_LOGON\_FAIL = 15007, /\* 0x3A9F \*/ /\* 邮件登陆服务器时，拒绝登陆。\*/

MSP\_ERROR\_MMP\_MAIL\_USER\_ILLEGAL = 15008, /\* 0x3AA0 \*/ /\* 邮件登陆服务器时，用户名非法。\*/

MSP\_ERROR\_MMP\_MAIL\_PWD\_ERR = 15009, /\* 0x3AA1 \*/ /\* 邮件登陆服务器时，密码错误。\*/

MSP\_ERROR\_MMP\_MAIL\_SOCKET\_ERR = 15010, /\* 0x3AA2 \*/ /\* 邮件发送过程中套接字错误\*/

MSP\_ERROR\_MMP\_MAIL\_INIT\_FAIL = 15011, /\* 0x3AA3 \*/ /\* 邮件初始化错误\*/

MSP\_ERROR\_MMP\_STORE\_MNR\_NO\_INIT = 15012, /\* 0x3AA4 \*/ /\* store\_manager未初始化，或初始化失败\*/

MSP\_ERROR\_MMP\_STORE\_MNR\_POOL\_FULL = 15013, /\* 0x3AA5 \*/ /\* store\_manager的连接池满了\*/

MSP\_ERROR\_MMP\_STRATGY\_PARAM\_ILLEGAL = 15014, /\* 0x3AA6 \*/ /\* 报警策略表达式非法\*/

MSP\_ERROR\_MMP\_STRATGY\_PARAM\_TOOLOOG = 15015, /\* 0x3AA7 \*/ /\* 报警策略表达式太长\*/

MSP\_ERROR\_MMP\_PARAM\_NULL = 15016, /\* 0x3AA8 \*/ /\* 函数参数为空\*/

MSP\_ERROR\_MMP\_ERR\_MORE\_TOTAL = 15017, /\* 0x3AA9 \*/ /\* pms插入数据库中错误汇总表的数据，错误次数 > 总次数。\*/

MSP\_ERROR\_MMP\_PROC\_THRESHOLD = 15018, /\* 0x3AAA \*/ /\* 进程监控阀值设置错误\*/

MSP\_ERROR\_MMP\_SERVER\_THRESHOLD = 15019, /\* 0x3AAB \*/ /\* 服务器监控阀值设置错误\*/

MSP\_ERROR\_MMP\_PYTHON\_NO\_EXIST = 15020, /\* 0x3AAC \*/ /\* python脚本文件不存在 \*/

MSP\_ERROR\_MMP\_PYTHON\_IMPORT\_FAILED = 15021, /\* 0x3AAD \*/ /\* python脚本导入出错 \*/

MSP\_ERROR\_MMP\_PYTHON\_BAD\_FUNC = 15022, /\* 0x3AAE \*/ /\* python脚本函数格式错误 \*/

MSP\_ERROR\_MMP\_DB\_DATA\_ILLEGAL = 15023, /\* 0x3AAF \*/ /\* 插入数据库中的数据格式有误 \*/

MSP\_ERROR\_MMP\_REDIS\_NOT\_CONN = 15024, /\* 0x3AB0 \*/ /\* redis没有连接到服务端 \*/

MSP\_ERROR\_MMP\_PMA\_NOT\_FOUND\_STRATEGY = 15025, /\* 0x3AB1 \*/ /\* 没有找到报警策略 \*/

MSP\_ERROR\_MMP\_TAIR\_CONNECT = 15026, /\* 0x3AB2 \*/ /\* 连接tair集群失败 \*/

MSP\_ERROR\_MMP\_PMC\_SERVINFO\_INVALID = 15027, /\* Ox3AB3 \*/ /\* 此pmc的服务器信息已经无效 \*/

MSP\_ERROR\_MMP\_ALARM\_GROUP\_NULL = 15028, /\* Ox3AB4 \*/ /\* 服务器报警的短信报警组与邮件报警组均为空 \*/

MSP\_ERROR\_MMP\_ALARM\_CONTXT\_NULL = 15029, /\* Ox3AB5 \*/ /\* 服务器报警的报警内容为空 \*/

/\* Error codes of MSC(lmod loader) \*/

MSP\_ERROR\_LMOD\_BASE = 16000, /\* 0x3E80 \*/

MSP\_ERROR\_LMOD\_NOT\_FOUND = 16001, /\* 0x3E81 \*/ /\* 没找到lmod文件 \*/

MSP\_ERROR\_LMOD\_UNEXPECTED\_BIN = 16002, /\* 0x3E82 \*/ /\* 无效的lmod \*/

MSP\_ERROR\_LMOD\_LOADCODE = 16003, /\* 0x3E83 \*/ /\* 加载lmod指令失败 \*/

MSP\_ERROR\_LMOD\_PRECALL = 16004, /\* 0x3E84 \*/ /\* 初始化lmod失败 \*/

MSP\_ERROR\_LMOD\_RUNTIME\_EXCEPTION = 16005, /\* 0x3E85 \*/ /\* lmod运行时异常 \*/

MSP\_ERROR\_LMOD\_ALREADY\_LOADED = 16006, /\* 0x3E86 \*/ /\* lmod重复加载 \*/

// Error code of Third Business

MSP\_ERROR\_BIZ\_BASE = 17000, /\* 0x4268 \*/ /\* 三方业务错误码 \*/

//Error of Nginx errlog file increase exception

MSP\_ERROR\_NGX\_LOG\_MORE\_TOTEL\_SIZE = 18000, /\*nginx错误日志大小异常\*/

//Error of Flash client when network checking

MSP\_ERROR\_FLASH\_NETWORK\_CONNECT\_FIALED = 19000, /\*flash服务端网络连接失败\*/

MSP\_ERROR\_FLASH\_NETWORK\_CHECK\_FIALED = 19001, /\*flash服务端响应了异常消息\*/

MSP\_ERROR\_FLASH\_NETWORK\_CHECK\_TIMEOUT = 19002, /\*flash服务端网络超时\*/

MSP\_ERROR\_FLASH\_NETWORK\_CLOSED\_EXCEPTION = 19003, /\*flash服务端网络异常关闭\*/

/\*Error Code Of Speech plus\*/

SPEECH\_ERROR\_NO\_NETWORK = 20001, /\* 无有效的网络连接\*/

SPEECH\_ERROR\_NETWORK\_TIMEOUT = 20002, /\* 网络连接超时\*/

SPEECH\_ERROR\_NET\_EXPECTION = 20003, /\* 网络异常\*/

SPEECH\_ERROR\_INVALID\_RESULT = 20004, /\* 无有效的结果\*/

SPEECH\_ERROR\_NO\_MATCH = 20005, /\* 无匹配结果 \*/

SPEECH\_ERROR\_AUDIO\_RECORD = 20006, /\* 录音失败 \*/

SPEECH\_ERROR\_NO\_SPPECH = 20007, /\* 未检测到语音\*/

SPEECH\_ERROR\_SPEECH\_TIMEOUT = 20008, /\* 音频输入超时\*/

SPEECH\_ERROR\_EMPTY\_UTTERANCE = 20009, /\* 无效的文本输入 \*/

SPEECH\_ERROR\_FILE\_ACCESS = 20010, /\* 文件读写失败 \*/

SPEECH\_ERROR\_PLAY\_MEDIA = 20011, /\* 音频播放失败 \*/

SPEECH\_ERROR\_INVALID\_PARAM = 20012, /\* 无效的参数\*/

SPEECH\_ERROR\_TEXT\_OVERFLOW = 20013, /\* 文本溢出 \*/

SPEECH\_ERROR\_INVALID\_DATA = 20014, /\* 无效数据 \*/

SPEECH\_ERROR\_LOGIN = 20015, /\* 用户未登陆\*/

SPEECH\_ERROR\_PERMISSION\_DENIED = 20016, /\* 无效授权 \*/

SPEECH\_ERROR\_INTERRUPT = 20017, /\* 被异常打断 \*/

SPEECH\_ERROR\_VERSION\_LOWER = 20018, /\* 版本过低 \*/

SPEECH\_CLIENT\_ERROR\_ISUSING = 20019, /\* 录音机被占用(iOS平台) \*/

SPEECH\_ERROR\_SYSTEM\_PREINSTALL = 20020, /\* 系统预置版本 \*/

SPEECH\_ERROR\_UNSATISFIED\_LINK = 20021, /\* 未实现的Native函数引用 \*/

SPEECH\_ERROR\_UNKNOWN = 20999, /\* 未知错误 \*/

SPEECH\_ERROR\_COMPONENT\_NOT\_INSTALLED = 21001, /\* 没有安装语音组件 \*/

SPEECH\_ERROR\_ENGINE\_NOT\_SUPPORTED = 21002, /\* 引擎不支持 \*/

SPEECH\_ERROR\_ENGINE\_INIT\_FAIL = 21003, /\* 初始化失败 \*/

SPEECH\_ERROR\_ENGINE\_CALL\_FAIL = 21004, /\* 调用失败 \*/

SPEECH\_ERROR\_ENGINE\_BUSY = 21005, /\* 引擎繁忙 \*/

SPEECH\_ERROR\_LOCAL\_NO\_INIT = 22001, /\* 本地引擎未初始化 \*/

SPEECH\_ERROR\_LOCAL\_RESOURCE = 22002, /\* 本地引擎无资源 \*/

SPEECH\_ERROR\_LOCAL\_ENGINE = 22003, /\* 本地引擎内部错误 \*/

SPEECH\_ERROR\_IVW\_INTERRUPT = 22004, /\* 本地唤醒引擎被异常打断 \*/

/\*Error Code Of Local iflytek Engines\*/

/\*Error Code Of AiTalk\*/

/\*Error Code Of AiTalk Operation\*/

SPEECH\_SUCCESS = 0 , // ivErr\_OK = 0 /\*成功状态\*/

SPEECH\_ERROR\_ASR\_CLIENT = 23000, /\*客户端应用程序错误\*///?????????

SPEECH\_ERROR\_ASR\_INVALID\_PARA = 23001, /\*无效的参数\*/

SPEECH\_ERROR\_ASR\_INVALID\_PARA\_VALUE = 23002, /\*无效的参数值\*/

SPEECH\_ERROR\_ASR\_OUT\_OF\_MEMORY = 23003, /\*内存耗尽\*/

SPEECH\_ERROR\_ASR\_CREATE\_HANDLE\_FAILED = 23004, /\*创建句柄失败\*/

SPEECH\_ERROR\_ASR\_ENGINE\_INIT\_FAILED = 23005, /\*引擎初始化失败\*/

SPEECH\_ERROR\_ASR\_ENGINE\_STARTED = 23006, /\*引擎已经启动\*/

SPEECH\_ERROR\_ASR\_ENGINE\_UNINIT = 23007, /\*引擎未初始化\*/

SPEECH\_ERROR\_ASR\_SPEECH\_TIMEOUT = 23008, /\*识别超时（VAD没开启或没有检测到后端点）\*/

SPEECH\_ERROR\_ASR\_NO\_RECOGNIZED\_RESULT = 23009, /\*无识别结果\*/

SPEECH\_ERROR\_ASR\_INVALID\_HANDLE = 23010, /\*无效的句柄\*/

SPEECH\_ERROR\_ASR\_FILE\_ACCESS = 23011, /\*打开文件失败\*/

/\*Error Code Of AiTalk Engine\*/

SPEECH\_ERROR\_AITALK\_FALSE = 23100, // ivErr\_FALSE = 1

/\* For license check \*/

SPEECH\_ERROR\_AITALK\_PERMISSION\_DENIED = 23101, // ivErr\_InvSN = 2

/\* General \*/

SPEECH\_ERROR\_AITALK\_INVALID\_PARA = 23102, // ivErr\_InvArg = 3

SPEECH\_ERROR\_AITALK\_BUFFER\_OVERFLOW = 23103, // ivErr\_BufferFull = 4 /\*音频数据缓冲区已满\*/

SPEECH\_ERROR\_AITALK\_FAILED = 23104, // ivErr\_Failed = 5

SPEECH\_ERROR\_AITALK\_NOT\_SUPPORTED = 23105, // ivErr\_NotSupport = 6 /\*引擎不支持\*/

SPEECH\_ERROR\_AITALK\_OUT\_OF\_MEMORY = 23106, // ivErr\_OutOfMemory = 7

SPEECH\_ERROR\_AITALK\_INVALID\_RESOURCE = 23107, // ivErr\_InvResource = 8 /\*资源无效\*/

SPEECH\_ERROR\_AITALK\_NOT\_FOUND = 23108, // ivErr\_NotFound = 9 /\*打开文件失败\*/

SPEECH\_ERROR\_AITALK\_INVALID\_GRAMMAR = 23109, // ivErr\_InvGrmr = 10 /\*识别语法错误\*/

/\* For object status \*/

SPEECH\_ERROR\_AITALK\_INVALID\_CALL = 23110, // ivErr\_InvCall = 11 /\*无效调用\*/

/\* For ASR Input \*/

SPEECH\_ERROR\_AITALK\_SYNTAX\_ERROR = 23111, // ivErr\_InvCall = 12

/\* For Message Call Back \*/

SPEECH\_ERROR\_AITALK\_RESET = 23112, // ivErr\_Reset = 13

SPEECH\_ERROR\_AITALK\_ENDED = 23113, // ivErr\_Ended = 14

SPEECH\_ERROR\_AITALK\_IDLE = 23114, // ivErr\_Idle = 15

SPEECH\_ERROR\_AITALK\_CANNOT\_SAVE\_FILE = 23115, // ivErr\_CanNotSaveFile = 16

/\* For Lexicon name \*/

SPEECH\_ERROR\_AITALK\_INVALID\_GRAMMAR\_NAME = 23116, // ivErr\_InvName = 17 /\*文法或词典名称非法\*/

SPEECH\_ERROR\_AITALK\_BUFFER\_EMPTY = 23117, // ivErr\_BufferEmpty = 18

SPEECH\_ERROR\_AITALK\_GET\_RESULT = 23118, // ivErr\_GetResult = 19

SPEECH\_ERROR\_AITALK\_REACT\_OUT\_TIME = 23119, // ivErr\_ReactOutTime = 20 /\*反应超时\*/

SPEECH\_ERROR\_AITALK\_SPEECH\_OUT\_TIME = 23120, // ivErr\_SpeechOutTime = 21 /\*语音超时\*/

SPEECH\_ERROR\_AITALK\_AUDIO\_CUT = 23121, // ivErr\_CUT = 22 /\*录音质量过高\*/

SPEECH\_ERROR\_AITALK\_AUDIO\_LOWER = 23122, // ivErr\_LOWER = 23 /\*录音质量过低\*/

SPEECH\_ERROR\_AITALK\_INSUFFICIENT\_PERMISSIONS = 23123, // ivErr\_Limitted = 24 /\*授权不够\*/

SPEECH\_ERROR\_AITALK\_RESULT\_ERROR = 23124, // ivErr\_ResultError = 25 /\*解码器Wfst输出后，依然有cmd输出\*/

SPEECH\_ERROR\_AITALK\_SHORT\_PAUSE = 23125, // ivErr\_ShortPause = 26

SPEECH\_ERROR\_AITALK\_BUSY = 23126, // ivErr\_Busy = 27

SPEECH\_ERROR\_AITALK\_GRM\_NOT\_UPDATE = 23127, // ivErr\_GrmNotUpdate = 28 /\*语法未更新\*/

SPEECH\_ERROR\_AITALK\_STARTED = 23128, // ivErr\_Started = 29

SPEECH\_ERROR\_AITALK\_STOPPED = 23129, // ivErr\_Stopped = 30

SPEECH\_ERROR\_AITALK\_ALREADY\_STARTED = 23130, // ivErr\_AlreadyStarted = 31

SPEECH\_ERROR\_AITALK\_ALREADY\_STOPPED = 23131, // ivErr\_AlreadyStopped = 32

SPEECH\_ERROR\_AITALK\_TOO\_MANY\_COMMAND = 23132, // ivErr\_TooManyCmd = 33

SPEECH\_ERROR\_AITALK\_WAIT = 23133, // ivErr\_Wait = 34 /\*程序可能在做一些操作，主线程需要等待\*/

SPEECH\_ERROR\_AITALK\_MAE\_RIGHT = 23134, // ivErr\_MAERight = 35

SPEECH\_ERROR\_AITALK\_MAE\_WRONG = 23135, // ivErr\_MAEWrong = 36

SPEECH\_ERROR\_AITALK\_GRM\_ERR = 23300, // 语法错误

/\*Error Code Of AiSound\*/

/\*Error Code Of AiSound Operation\*/

SPEECH\_ERROR\_TTS\_INVALID\_PARA = 24000, /\* 错误参数 \*/

SPEECH\_ERROR\_TTS\_INVALID\_PARA\_VALUE = 24001, /\* 无效的参数值\*/

SPEECH\_ERROR\_TTS\_OUT\_OF\_MEMORY = 24002, /\* 内存不足\*/

SPEECH\_ERROR\_TTS\_INVALID\_HANDLE = 24003, /\* 无效的句柄\*/

SPEECH\_ERROR\_TTS\_CREATE\_HANDLE\_FAILED = 24004, /\* 创建句柄失败\*/

SPEECH\_ERROR\_TTS\_INVALID\_RESOURCE = 24005, /\* 无效资源 \*/

SPEECH\_ERROR\_TTS\_INVALID\_VOICE\_NAME = 24006, /\* 无效发言人\*/

SPEECH\_ERROR\_TTS\_ENGINE\_UNINIT = 24007, /\* 引擎未初始化 \*/

SPEECH\_ERROR\_TTS\_ENGINE\_INIT\_FAILED = 24008, /\* 引擎初始化失败 \*/

SPEECH\_ERROR\_TTS\_ENGINE\_BUSY = 24009, /\* 引擎忙 \*/

/\*Error Code Of AiSound Engine\*/

SPEECH\_ERROR\_AISOUND\_BASE = 24100,

SPEECH\_ERROR\_AISOUND\_UNIMPEMENTED = 24100, /\* unimplemented function \*/

SPEECH\_ERROR\_AISOUND\_UNSUPPORTED = 24101, /\* unsupported on this platform \*/

SPEECH\_ERROR\_AISOUND\_INVALID\_HANDLE = 24102, /\* invalid handle \*/

SPEECH\_ERROR\_AISOUND\_INVALID\_PARA = 24103, /\* invalid parameter(s) \*/

SPEECH\_ERROR\_AISOUND\_INSUFFICIENT\_HEAP = 24104, /\* insufficient heap size \*/

SPEECH\_ERROR\_AISOUND\_STATE\_REFUSE = 24105, /\* refuse to do in current state \*/

SPEECH\_ERROR\_AISOUND\_INVALID\_PARA\_ID = 24106, /\* invalid parameter ID \*/

SPEECH\_ERROR\_AISOUND\_INVALID\_PARA\_VALUE = 24107, /\* invalid parameter value \*/

SPEECH\_ERROR\_AISOUND\_RESOURCE = 24108, /\* Resource is error \*/

SPEECH\_ERROR\_AISOUND\_RESOURCE\_READ = 24109, /\* read resource error \*/

SPEECH\_ERROR\_AISOUND\_LBENDIAN = 24110, /\* the Endian of SDK is error \*/

SPEECH\_ERROR\_AISOUND\_HEADFILE = 24111, /\* the HeadFile is different of the SDK \*/

SPEECH\_ERROR\_AISOUND\_BUFFER\_OVERFLOW = 24112, /\* get data size exceed the data buffer \*/

SPEECH\_ERROR\_AISOUND\_INVALID\_ISAMPA = 24113, /\* !Invalid iSampa format or input iSampa text contain invalid alphabet\*/

SPEECH\_ERROR\_AISOUND\_INVALID\_CSSML = 24114, /\* !Invalid cssml format \*/

/\*Error Code Of ivw\*/

/\*Error Code Of ivw Operation\*/

SPEECH\_ERROR\_IVW\_ENGINE\_UNINI = 25000, /\* 引擎未初始化 \*/

SPEECH\_ERROR\_IVW\_RESVER\_NOMATCH = 25001, /\* 资源版本不匹配 \*/

SPEECH\_ERROR\_IVW\_BUFFERED\_AUDIOD\_LITTLE = 25002, /\* 唤醒加识别缓存音频过少 \*/

SPEECH\_ERROR\_IVW\_INVALID\_RESTYPE = 25003, /\* 不合法的资源类型 \*/

/\*Error Code Of ivw Engine\*/

SPEECH\_ERROR\_IVW\_INVALID\_CALL = 25101, // IvwErr\_InvCal = 1

SPEECH\_ERROR\_IVW\_INVALID\_ARG = 25102, // IvwErr\_InvArg = 2

SPEECH\_ERROR\_IVW\_TELL\_SIZE = 25103, // IvwErr\_TellSize = 3

SPEECH\_ERROR\_IVW\_OUT\_OF\_MEMORY = 25104, // IvwErr\_OutOfMemory = 4

SPEECH\_ERROR\_IVW\_OUT\_BUFFER\_FULL = 25105, // IvwErr\_BufferFull = 5

SPEECH\_ERROR\_IVW\_OUT\_BUFFER\_EMPTY = 25106, // IvwErr\_BufferEmpty = 6

SPEECH\_ERROR\_IVW\_INVALID\_RESOURCE = 25107, // IvwErr\_InvRes = 7

SPEECH\_ERROR\_IVW\_REPETITIOPN\_ENTER = 25108, // IvwErr\_ReEnter = 8

SPEECH\_ERROR\_IVW\_NOT\_SUPPORT = 25109, // IvwErr\_NotSupport = 9

SPEECH\_ERROR\_IVW\_NOT\_FOUND = 25110, // IvwErr\_NotFound = 10

SPEECH\_ERROR\_IVW\_INVALID\_SN = 25111, // IvwErr\_InvSN = 11

SPEECH\_ERROR\_IVW\_LIMITTED = 25112, // IvwErr\_Limitted = 12

SPEECH\_ERROR\_IVW\_TIME\_OUT = 25113, // IvwErr\_TimeOut = 13

SPEECH\_ERROR\_IVW\_ENROLL1\_SUCESS = 25114, // IvwErr\_Enroll1\_Success = 14

SPEECH\_ERROR\_IVW\_ENROLL1\_FAILED = 25115, // IvwErr\_Enroll1\_Failed = 15

SPEECH\_ERROR\_IVW\_ENROLL2\_SUCESS = 25116, // IvwErr\_Enroll2\_Success = 16

SPEECH\_ERROR\_IVW\_ENROLL2\_FAILED = 25117, // IvwErr\_Enroll2\_Failed = 17

SPEECH\_ERROR\_IVW\_ENROLL3\_SUCESS = 25118, // IvwErr\_Enroll3\_Success = 18

SPEECH\_ERROR\_IVW\_ENROLL3\_FAILED = 25119, // IvwErr\_Enroll3\_Failed = 19

SPEECH\_ERROR\_IVW\_SPEECH\_TOO\_SHORT = 25120, // IvwErr\_SpeechTooShort = 20

SPEECH\_ERROR\_IVW\_SPEECH\_STOP = 25121 // IvwErr\_SpeechStop = 21

};

#endif /\* \_\_MSP\_ERRORS\_H\_\_ \*/

#ifndef \_\_MSP\_TYPES\_H\_\_

#define \_\_MSP\_TYPES\_H\_\_

#if !defined(MSPAPI)

#if defined(WIN32) || defined(WINPHONE8) || defined(WIN8)

#define MSPAPI \_\_stdcall

#else

#define MSPAPI

#endif /\* WIN32 \*/

#endif /\* MSPAPI \*/

/\*\*

\* MSPSampleStatus indicates how the sample buffer should be handled

\* MSP\_AUDIO\_SAMPLE\_FIRST - The sample buffer is the start of audio

\* If recognizer was already recognizing, it will discard

\* audio received to date and re-start the recognition

\* MSP\_AUDIO\_SAMPLE\_CONTINUE - The sample buffer is continuing audio

\* MSP\_AUDIO\_SAMPLE\_LAST - The sample buffer is the end of audio

\* The recognizer will cease processing audio and

\* return results

\* Note that sample statii can be combined; for example, for file-based input

\* the entire file can be written with SAMPLE\_FIRST | SAMPLE\_LAST as the

\* status.

\* Other flags may be added in future to indicate other special audio

\* conditions such as the presence of AGC

\*/

enum

{

MSP\_AUDIO\_SAMPLE\_INIT = 0x00,

MSP\_AUDIO\_SAMPLE\_FIRST = 0x01,

MSP\_AUDIO\_SAMPLE\_CONTINUE = 0x02,

MSP\_AUDIO\_SAMPLE\_LAST = 0x04,

};

/\*

\* The enumeration MSPRecognizerStatus contains the recognition status

\* MSP\_REC\_STATUS\_SUCCESS - successful recognition with partial results

\* MSP\_REC\_STATUS\_NO\_MATCH - recognition rejected

\* MSP\_REC\_STATUS\_INCOMPLETE - recognizer needs more time to compute results

\* MSP\_REC\_STATUS\_NON\_SPEECH\_DETECTED - discard status, no more in use

\* MSP\_REC\_STATUS\_SPEECH\_DETECTED - recognizer has detected audio, this is delayed status

\* MSP\_REC\_STATUS\_COMPLETE - recognizer has return all result

\* MSP\_REC\_STATUS\_MAX\_CPU\_TIME - CPU time limit exceeded

\* MSP\_REC\_STATUS\_MAX\_SPEECH - maximum speech length exceeded, partial results may be returned

\* MSP\_REC\_STATUS\_STOPPED - recognition was stopped

\* MSP\_REC\_STATUS\_REJECTED - recognizer rejected due to low confidence

\* MSP\_REC\_STATUS\_NO\_SPEECH\_FOUND - recognizer still found no audio, this is delayed status

\*/

enum

{

MSP\_REC\_STATUS\_SUCCESS = 0,

MSP\_REC\_STATUS\_NO\_MATCH = 1,

MSP\_REC\_STATUS\_INCOMPLETE = 2,

MSP\_REC\_STATUS\_NON\_SPEECH\_DETECTED = 3,

MSP\_REC\_STATUS\_SPEECH\_DETECTED = 4,

MSP\_REC\_STATUS\_COMPLETE = 5,

MSP\_REC\_STATUS\_MAX\_CPU\_TIME = 6,

MSP\_REC\_STATUS\_MAX\_SPEECH = 7,

MSP\_REC\_STATUS\_STOPPED = 8,

MSP\_REC\_STATUS\_REJECTED = 9,

MSP\_REC\_STATUS\_NO\_SPEECH\_FOUND = 10,

MSP\_REC\_STATUS\_FAILURE = MSP\_REC\_STATUS\_NO\_MATCH,

};

/\*\*

\* The enumeration MSPepState contains the current endpointer state

\* MSP\_EP\_LOOKING\_FOR\_SPEECH - Have not yet found the beginning of speech

\* MSP\_EP\_IN\_SPEECH - Have found the beginning, but not the end of speech

\* MSP\_EP\_AFTER\_SPEECH - Have found the beginning and end of speech

\* MSP\_EP\_TIMEOUT - Have not found any audio till timeout

\* MSP\_EP\_ERROR - The endpointer has encountered a serious error

\* MSP\_EP\_MAX\_SPEECH - Have arrive the max size of speech

\*/

enum

{

MSP\_EP\_LOOKING\_FOR\_SPEECH = 0,

MSP\_EP\_IN\_SPEECH = 1,

MSP\_EP\_AFTER\_SPEECH = 3,

MSP\_EP\_TIMEOUT = 4,

MSP\_EP\_ERROR = 5,

MSP\_EP\_MAX\_SPEECH = 6,

MSP\_EP\_IDLE = 7 // internal state after stop and before start

};

/\* Synthesizing process flags \*/

enum

{

MSP\_TTS\_FLAG\_STILL\_HAVE\_DATA = 1,

MSP\_TTS\_FLAG\_DATA\_END = 2,

MSP\_TTS\_FLAG\_CMD\_CANCELED = 4,

};

/\* Handwriting process flags \*/

enum

{

MSP\_HCR\_DATA\_FIRST = 1,

MSP\_HCR\_DATA\_CONTINUE = 2,

MSP\_HCR\_DATA\_END = 4,

};

/\*ivw message type \*/

enum

{

MSP\_IVW\_MSG\_WAKEUP = 1,

MSP\_IVW\_MSG\_ERROR = 2,

MSP\_IVW\_MSG\_ISR\_RESULT = 3,

MSP\_IVW\_MSG\_ISR\_EPS = 4,

MSP\_IVW\_MSG\_VOLUME = 5,

MSP\_IVW\_MSG\_ENROLL = 6,

MSP\_IVW\_MSG\_RESET = 7

};

/\* Upload data process flags \*/

enum

{

MSP\_DATA\_SAMPLE\_INIT = 0x00,

MSP\_DATA\_SAMPLE\_FIRST = 0x01,

MSP\_DATA\_SAMPLE\_CONTINUE = 0x02,

MSP\_DATA\_SAMPLE\_LAST = 0x04,

};

#endif /\* \_\_MSP\_TYPES\_H\_\_ \*/

/\*\*

\* @file qisr.h

\* @brief iFLY Speech Recognizer Header File

\*

\* This file contains the quick application programming interface (API) declarations

\* of ISR. Developer can include this file in your project to build applications.

\* For more information, please read the developer guide.

\* Use of this software is subject to certain restrictions and limitations set

\* forth in a license agreement entered into between iFLYTEK, Co,LTD.

\* and the licensee of this software. Please refer to the license

\* agreement for license use rights and restrictions.

\*

\* Copyright (C) 1999 - 2007 by ANHUI USTC iFLYTEK, Co,LTD.

\* All rights reserved.

\*

\* @author Speech Dept. iFLYTEK.

\* @version 1.0

\* @date 2008/12/12

\*

\* @see

\*

\* History:

\* index version date author notes

\* 0 1.0 2008/12/12 Speech Create this file

\*/

#ifndef \_\_QISR\_H\_\_

#define \_\_QISR\_H\_\_

#ifdef \_\_cplusplus

extern "C" {

#endif /\* C++ \*/

#include "msp\_types.h"

/\*\*

\* @fn QISRSessionBegin

\* @brief Begin a Recognizer Session

\*

\* Create a recognizer session to recognize audio data

\*

\* @return return sessionID of current session, NULL is failed.

\* @param const char\* grammarList - [in] grammars list, inline grammar support only one.

\* @param const char\* params - [in] parameters when the session created.

\* @param int \*errorCode - [out] return 0 on success, otherwise return error code.

\* @see

\*/

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

typedef const char\* (MSPAPI \*Proc\_QISRSessionBegin)(const char\* grammarList, const char\* params, int \*result);

#ifdef MSP\_WCHAR\_SUPPORT

const wchar\_t\* MSPAPI QISRSessionBeginW(const wchar\_t\* grammarList, const wchar\_t\* params, int \*result);

typedef const wchar\_t\* (MSPAPI \*Proc\_QISRSessionBeginW)(const wchar\_t\* grammarList, const wchar\_t\* params, int \*result);

#endif

/\*\*

\* @fn QISRAudioWrite

\* @brief Write Audio Data to Recognizer Session

\*

\* Writing binary audio data to recognizer.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* sessionID - [in] The session id returned by recog\_begin

\* @param const void\* waveData - [in] Binary data of waveform

\* @param unsigned int waveLen - [in] Waveform data size in bytes

\* @param int audioStatus - [in] Audio status, can be

\* @param int \*epStatus - [out] ISRepState

\* @param int \*recogStatus - [out] ISRrecRecognizerStatus, see isr\_rec.h

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_QISRAudioWrite)(const char\* sessionID, const void\* waveData, unsigned int waveLen, int audioStatus, int \*epStatus, int \*recogStatus);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QISRAudioWriteW(const wchar\_t\* sessionID, const void\* waveData, unsigned int waveLen, int audioStatus, int \*epStatus, int \*recogStatus);

typedef int (MSPAPI \*Proc\_QISRAudioWriteW)(const wchar\_t\* sessionID, const void\* waveData, unsigned int waveLen, int audioStatus, int \*epStatus, int \*recogStatus);

#endif

/\*\*

\* @fn QISRGetResult

\* @brief Get Recognize Result in Specified Format

\*

\* Get recognize result in Specified format.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* sessionID - [in] session id returned by session begin

\* @param int\* rsltStatus - [out] status of recognition result, 0: success, 1: no match, 2: incomplete, 5:speech complete

\* @param int \*errorCode - [out] return 0 on success, otherwise return error code.

\* @see

\*/

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

typedef const char \* (MSPAPI \*Proc\_QISRGetResult)(const char\* sessionID, int\* rsltStatus, int waitTime, int \*errorCode);

#ifdef MSP\_WCHAR\_SUPPORT

const wchar\_t\* MSPAPI QISRGetResultW(const wchar\_t\* sessionID, int\* rsltStatus, int waitTime, int \*errorCode);

typedef const wchar\_t\* (MSPAPI \*Proc\_QISRGetResultW)(const wchar\_t\* sessionID, int\* rsltStatus, int waitTime, int \*errorCode);

#endif

/\*\*

\* @fn QISRGetBinaryResult

\* @brief Get Recognize Result in Specified Format

\*

\* Get recognize result in Specified format.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* sessionID - [in] session id returned by session begin

\* @param int\* rsltStatus - [out] status of recognition result, 0: success, 1: no match, 2: incomplete, 5:speech complete

\* @param int \*errorCode - [out] return 0 on success, otherwise return error code.

\* @see

\*/

const char \* MSPAPI QISRGetBinaryResult(const char\* sessionID, unsigned int\* rsltLen,int\* rsltStatus, int waitTime, int \*errorCode);

typedef const char \* (MSPAPI \*Proc\_QISRGetBinaryResult)(const char\* sessionID, unsigned int\* rsltLen, int\* rsltStatus, int waitTime, int \*errorCode);

#ifdef MSP\_WCHAR\_SUPPORT

const wchar\_t\* MSPAPI QISRGetBinaryResultW(const wchar\_t\* sessionID, unsigned int\* rsltLen, int\* rsltStatus, int waitTime, int \*errorCode);

typedef const wchar\_t\* (MSPAPI \*Proc\_QISRGetBinaryResultW)(const wchar\_t\* sessionID, unsigned int\* rsltLen, int\* rsltStatus, int waitTime, int \*errorCode);

#endif

/\*\*

\* @fn QISRSessionEnd

\* @brief End a Recognizer Session

\*

\* End the recognizer session, release all resource.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* sessionID - [in] session id string to end

\* @param const char\* hints - [in] user hints to end session, hints will be logged to CallLog

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_QISRSessionEnd)(const char\* sessionID, const char\* hints);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QISRSessionEndW(const wchar\_t\* sessionID, const wchar\_t\* hints);

typedef int (MSPAPI \*Proc\_QISRSessionEndW)(const wchar\_t\* sessionID, const wchar\_t\* hints);

#endif

/\*\*

\* @fn QISRGetParam

\* @brief get params related with msc

\*

\* the params could be local or server param, we only support netflow params "upflow" & "downflow" now

\*

\* @return int - Return 0 if success, otherwise return errcode.

\* @param const char\* sessionID - [in] session id of related param, set NULL to got global param

\* @param const char\* paramName - [in] param name,could pass more than one param split by ','';'or'\n'

\* @param const char\* paramValue - [in] param value buffer, malloced by user

\* @param int \*valueLen - [in, out] pass in length of value buffer, and return length of value string

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_QISRGetParam)(const char\* sessionID, const char\* paramName, char\* paramValue, unsigned int\* valueLen);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QISRGetParamW(const wchar\_t\* sessionID, const wchar\_t\* paramName, wchar\_t\* paramValue, unsigned int\* valueLen);

typedef int (MSPAPI \*Proc\_QISRGetParamW)(const wchar\_t\* sessionID, const wchar\_t\* paramName, wchar\_t\* paramValue, unsigned int\* valueLen);

#endif

/\*\*

\* @fn QISRSetParam

\* @brief get params related with msc

\*

\* the params could be local or server param, we only support netflow params "upflow" & "downflow" now

\*

\* @return int - Return 0 if success, otherwise return errcode.

\* @param const char\* sessionID - [in] session id of related param, set NULL to got global param

\* @param const char\* paramName - [in] param name,could pass more than one param split by ','';'or'\n'

\* @param const char\* paramValue - [in] param value buffer, malloced by user

\* @param int \*valueLen - [in, out] pass in length of value buffer, and return length of value string

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_QISRSetParam)(const char\* sessionID, const char\* paramName, const char\* paramValue);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QISRSetParamW(const wchar\_t\* sessionID, const wchar\_t\* paramName, const wchar\_t\* paramValue);

typedef int (MSPAPI \*Proc\_QISRSetParamW)(const wchar\_t\* sessionID, const wchar\_t\* paramName, const wchar\_t\* paramValue);

#endif

typedef void ( \*recog\_result\_ntf\_handler)( const char \*sessionID, const char \*result, int resultLen, int resultStatus, void \*userData );

typedef void ( \*recog\_status\_ntf\_handler)( const char \*sessionID, int type, int status, int param1, const void \*param2, void \*userData);

typedef void ( \*recog\_error\_ntf\_handler)(const char \*sessionID, int errorCode, const char \*detail, void \*userData);

int MSPAPI QISRRegisterNotify(const char \*sessionID, recog\_result\_ntf\_handler rsltCb, recog\_status\_ntf\_handler statusCb, recog\_error\_ntf\_handler errCb, void \*userData);

typedef int ( \*UserCallBack)( int, const char\*, void\*);

typedef int ( \*GrammarCallBack)( int, const char\*, void\*);

typedef int ( \*LexiconCallBack)( int, const char\*, void\*);

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

typedef int (MSPAPI \*Proc\_QISRBuildGrammar)(const char \*grammarType, const char \*grammarContent, unsigned int grammarLength, const char \*params, GrammarCallBack callback, void \*userData);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QISRBuildGrammarW(const wchar\_t \*grmmarType, const wchar\_t \*grammarContent, unsigned int grammarLength, const wchar\_t \*params, GrammarCallBack callback, void \*userData);

typedef int (MSPAPI \*Proc\_QISRBuildGrammarW)(const wchar\_t \*grmmarType, const wchar\_t \*grammarContent, unsigned int grammarLength, const wchar\_t \*params, GrammarCallBack callback, void \*userData);

#endif

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

typedef int (MSPAPI \*Proc\_QISRUpdataLexicon)(const char \*lexiconName, const char \*lexiconContent, unsigned int lexiconLength, const char \*params, LexiconCallBack callback, void \*userData);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QISRUpdateLexiconW(const wchar\_t \*lexiconName, const wchar\_t \*lexiconContent, unsigned int lexiconLength, const wchar\_t \*params, LexiconCallBack callback, void \*userData);

typedef int (MSPAPI Proc\_QISRUpdateLexiconW)(const wchar\_t \*lexiconName, const wchar\_t \*lexiconContent, unsigned int lexiconLength, const wchar\_t \*params, LexiconCallBack callback, void \*userData);

#endif

#ifdef \_\_cplusplus

} /\* extern "C" \*/

#endif /\* C++ \*/

#endif /\* \_\_QISR\_H\_\_ \*/

/\*\*

\* @file qtts.h

\* @brief iFLY Speech Synthesizer Header File

\*

\* This file contains the quick application programming interface (API) declarations

\* of TTS. Developer can include this file in your project to build applications.

\* For more information, please read the developer guide.

\* Use of this software is subject to certain restrictions and limitations set

\* forth in a license agreement entered into between iFLYTEK, Co,LTD.

\* and the licensee of this software. Please refer to the license

\* agreement for license use rights and restrictions.

\*

\* Copyright (C) 1999 - 2009 by ANHUI USTC iFLYTEK, Co,LTD.

\* All rights reserved.

\*

\* @author Speech Dept.

\* @version 1.0

\* @date 2009/11/26

\*

\* @see

\*

\* <b>History:</b><br>

\* <table>

\* <tr> <th>Version <th>Date <th>Author <th>Notes</tr>

\* <tr> <td>1.0 <td>2009/11/26 <td>Speech <td>Create this file</tr>

\* </table>

\*

\*/

#ifndef \_\_QTTS\_H\_\_

#define \_\_QTTS\_H\_\_

#if !defined(MSPAPI)

#if defined(WIN32)

#define MSPAPI \_\_stdcall

#else

#define MSPAPI

#endif /\* WIN32 \*/

#endif /\* MSPAPI \*/

#ifdef \_\_cplusplus

extern "C" {

#endif /\* C++ \*/

#include "msp\_types.h"

/\*\*

\* @fn QTTSSessionBegin

\* @brief Begin a TTS Session

\*

\* Create a tts session to synthesize data.

\*

\* @return const char\* - Return the new session id in success, otherwise return NULL, error code.

\* @param const char\* params - [in] parameters when the session created.

\* @param const char\*\* sessionID - [out] return a string to this session.

\* @see

\*/

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

typedef const char\* (MSPAPI \*Proc\_QTTSSessionBegin)(const char\* params, int\* errorCode);

#ifdef MSP\_WCHAR\_SUPPORT

const wchar\_t\* MSPAPI QTTSSessionBeginW(const wchar\_t\* params, int\* errorCode);

typedef const wchar\_t\* (MSPAPI \*Proc\_QTTSSessionBeginW)(const wchar\_t\* params, int\* errorCode);

#endif

/\*\*

\* @fn QTTSTextPut

\* @brief Put Text Buffer to TTS Session

\*

\* Writing text string to synthesizer.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* sessionID - [in] The session id returned by sesson begin

\* @param const char\* textString - [in] text buffer

\* @param unsigned int textLen - [in] text size in bytes

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_QTTSTextPut)(const char\* sessionID, const char\* textString, unsigned int textLen, const char\* params);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QTTSTextPutW(const wchar\_t\* sessionID, const wchar\_t\* textString, unsigned int textLen, const wchar\_t\* params);

typedef int (MSPAPI \*Proc\_QTTSTextPutW)(const wchar\_t\* sessionID, const wchar\_t\* textString, unsigned int textLen, const wchar\_t\* params);

#endif

/\*\*

\* @fn QTTSAudioGet

\* @brief Synthesize text to audio

\*

\* Synthesize text to audio, and return audio information.

\*

\* @return const void\* - Return current synthesized audio data buffer, size returned by QTTSTextSynth.

\* @param const char\* sessionID - [in] session id returned by session begin

\* @param unsigned int\* audioLen - [out] synthesized audio size in bytes

\* @param int\* synthStatus - [out] synthesizing status

\* @param int\* errorCode - [out] error code if failed, 0 to success.

\* @see

\*/

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

typedef const void\* (MSPAPI \*Proc\_QTTSAudioGet)(const char\* sessionID, unsigned int\* audioLen, int\* synthStatus, int\* errorCode);

#ifdef MSP\_WCHAR\_SUPPORT

const void\* MSPAPI QTTSAudioGetW(const wchar\_t\* sessionID, unsigned int\* audioLen, int\* synthStatus, int\* errorCode);

typedef const void\* (MSPAPI \*Proc\_QTTSAudioGetW)(const wchar\_t\* sessionID, unsigned int\* audioLen, int\* synthStatus, int\* errorCode);

#endif

/\*\*

\* @fn QTTSAudioInfo

\* @brief Get Synthesized Audio information

\*

\* Get synthesized audio data information.

\*

\* @return const char \* - Return audio info string.

\* @param const char\* sessionID - [in] session id returned by session begin

\* @see

\*/

const char\* MSPAPI QTTSAudioInfo(const char\* sessionID);

typedef const char\* (MSPAPI \*Proc\_QTTSAudioInfo)(const char\* sessionID);

#ifdef MSP\_WCHAR\_SUPPORT

const wchar\_t\* MSPAPI QTTSAudioInfoW(const wchar\_t\* sessionID);

typedef const wchar\_t\* (MSPAPI \*Proc\_QTTSAudioInfoW)(const wchar\_t\* sessionID);

#endif

/\*\*

\* @fn QTTSSessionEnd

\* @brief End a Recognizer Session

\*

\* End the recognizer session, release all resource.

\*

\* @return int MSPAPI - Return 0 in success, otherwise return error code.

\* @param const char\* session\_id - [in] session id string to end

\* @param const char\* hints - [in] user hints to end session, hints will be logged to CallLog

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_QTTSSessionEnd)(const char\* sessionID, const char\* hints);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QTTSSessionEndW(const wchar\_t\* sessionID, const wchar\_t\* hints);

typedef int (MSPAPI \*Proc\_QTTSSessionEndW)(const wchar\_t\* sessionID, const wchar\_t\* hints);

#endif

/\*\*

\* @fn QTTSGetParam

\* @brief get params related with msc

\*

\* the params could be local or server param, we only support netflow params "upflow" & "downflow" now

\*

\* @return int - Return 0 if success, otherwise return errcode.

\* @param const char\* sessionID - [in] session id of related param, set NULL to got global param

\* @param const char\* paramName - [in] param name,could pass more than one param split by ','';'or'\n'

\* @param const char\* paramValue - [in] param value buffer, malloced by user

\* @param int \*valueLen - [in, out] pass in length of value buffer, and return length of value string

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_QTTSGetParam)(const char\* sessionID, const char\* paramName, char\* paramValue, unsigned int\* valueLen);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QTTSGetParamW(const wchar\_t\* sessionID, const wchar\_t\* paramName, wchar\_t\* paramValue, unsigned int\* valueLen);

typedef int (MSPAPI \*Proc\_QTTSGetParamW)(const wchar\_t\* sessionID, const wchar\_t\* paramName, wchar\_t\* paramValue, unsigned int\* valueLen);

#endif

/\*\*

\* @fn QTTSSetParam

\* @brief set params related with msc

\*

\* the params could be local or server param, we only support netflow params "upflow" & "downflow" now

\*

\* @return int - Return 0 if success, otherwise return errcode.

\* @param const char\* sessionID - [in] session id of related param, set NULL to got global param

\* @param const char\* paramName - [in] param name,could pass more than one param split by ','';'or'\n'

\* @param const char\* paramValue - [in] param value buffer, malloced by user

\* @see

\*/

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

typedef int (MSPAPI \*Proc\_QTTSSetParam)(const char\* sessionID, const char\* paramName, char\* paramValue);

#ifdef MSP\_WCHAR\_SUPPORT

int MSPAPI QTTSSetParamW(const wchar\_t\* sessionID, const wchar\_t\* paramName, wchar\_t\* paramValue);

typedef int (MSPAPI \*Proc\_QTTSSetParamW)(const wchar\_t\* sessionID, const wchar\_t\* paramName, wchar\_t\* paramValue);

#endif

typedef void ( \*tts\_result\_ntf\_handler)( const char \*sessionID, const char \*audio, int audioLen, int synthStatus, int ced, const char \*audioInfo, int audioInfoLen, void \*userData );

typedef void ( \*tts\_status\_ntf\_handler)( const char \*sessionID, int type, int status, int param1, const void \*param2, void \*userData);

typedef void ( \*tts\_error\_ntf\_handler)(const char \*sessionID, int errorCode, const char \*detail, void \*userData);

int MSPAPI QTTSRegisterNotify(const char \*sessionID, tts\_result\_ntf\_handler rsltCb, tts\_status\_ntf\_handler statusCb, tts\_error\_ntf\_handler errCb, void \*userData);

#ifdef \_\_cplusplus

} /\* extern "C" \*/

#endif /\* C++ \*/

#endif /\* \_\_QTTS\_H\_\_ \*/