Daoke Cloud

目录

- Daoke Cloud
  - 目录
  - 1.获取配置
    - 1.1接口说明
    - 1.2 URL
    - 1.3 http请求方式
    - 1.4 返回支持格式
    - 1.5 访问限制
    - 1.6 注意事项
    - 1.7 请求参数
    - 1.8 返回参数
    - 1.9 返回参数说明
  - 2.上传并获取语音信息
    - 2.1接口说明
    - 2.2 URL
    - 2.3 http请求方式
    - 2.4 返回支持格式
    - 2.5 访问限制
    - 2.6 注意事项
    - 2.7 请求参数
    - 2.8 返回参数
    - 2.9 返回参数说明
  - 3.交互接口
    - 3.1接口说明
    - 3.2 URL
    - 3.3 http请求方式
    - 3.4 返回支持格式
    - 3.5 访问限制
    - 3.6 注意事项
    - 3.7 请求参数
    - 3.8 返回参数
  - 4 附录
    - 4.1 WEME终端按键对照表
    - 4.2 “上传并获取信息”接口weme终端的实现参考

1.获取配置

1.1接口说明

> 每次终端开机都必须通过该接口拿到配置授权信息，否则无法调用其他的接口。

1.2 URL

```
http://gcconfig.mirrtalk.com/config
```

1.3 http请求方式

```
GET
```

1.4 返回支持格式

```
JSON
```

1.5 访问限制

```
访问级别：普通接口
频次限制：30次/天
请求超时：30s
```

1.6 注意事项

```
服务器返回的数据通过response对象的HEAD中的RESULT获取。
```

1.7 请求参数

参数名	长度	类型	示例	说明
mod	5	大写字母，数字	SG900	机器型号，需要申请
imei	15	字符串纯数字	850212090004492	机器识别码，需语镜公司授权
imsi	15	字符串纯数字	460023027904550	sim卡号

1.8 返回参数

```
{
    "call1":"10010",
    "call2":"10086",
    "domain":"test.mirrtalk.com",
    "port":8080,
    "tokencode":"61HlullElt",
    "customargs":{
        "voiceCommand":true,
```

```
        "groupVoice":true,
        "autoSend":false
    }
}
```

1.9 返回参数说明

返回参数	参数类型	参数说明	值范围、示例
call1	string	直拨电话号码1	95511
call2	string	直拨电话号码2	10010
domain	string	上传获取服务、反馈接口域名	proddata.mirrtalk.com
port	int	上传获取服务、反馈接口端口	80
tokencode	string	授权码，每次开机都会分配	61HlullElt
customargs	object	开发者自定义配置参数	必须是json格式
voiceCommand	bool	是否开启语音指令功能	true
groupVoice	bool	是否开启群组语音功能	true
autoSend	bool	是否10s自动发送	false

- **tokencode** 授权码用于调用其他业务接口，且每次开机都不同，一个tokencode的有效期为48小时。
- **customargs** 在DaokeIO服务管理界面设置。一个型号的机器只能设置一组。格式为json格式。用于满足不同机型的参数配置。
- 系统默认参数名全部为小写。

2.上传并获取语音信息

2.1接口说明

```
终端开机后，请每5秒上传一次数据并获取服务器的信息服务。
```

2.2 URL

```
http://domain:port/newstatus
```

2.3 http请求方式

```
POST
```

2.4 返回支持格式

```
Binary
```

2.5 访问限制

```
访问级别：普通接口
频次限制：无限制
请求超时：15s
```

2.6 注意事项

```
domain和port为config接口中获取的系统参数。
请求必须用POST FORM方式提交。
服务器返回的数据通过response对象的HEAD中的RESULT获取。
服务器返回的文件流response对象中的Body获取。
```

2.7 请求参数

参数名	长度	类型	示例	说明
imei	15	string	355688000000158	终端设备号
imsi	15	string	460013002605754	sim卡号
mod	5	string	SG900	终端型号
tokencode	10	大小写字母，数字	61HlullElt	config接口获取的参数
gps	不定长	字符串	250314092030,12122.11260E,3113.16226N,170,87,99999	gps数据
other	不定长	字符串		其他传感器数据
bizid	34	大小写字母，	a123123456789a987654321B123456789a	已读信息唯一ID

数字

- **gps参数说明**
  包含5组连续的gps点数据，每组数据必须必须包含采集时间、经度、纬度、角度、速度、海拔。每组数据用“;”分隔。

  【示例数据】

```
gps=210314112755,11738.55807E,3058.35950N,-1,33,20;210314112754,11738.55810E,3058.35948N,-1,55,20;210314112752,11738.55797E,3058.35939N,-1,23,20;210314112750,11738.55784E,3058.35960N,-1,45,20;210314112749,11738.55784E,3058.35957N,-1,46,20
```

参数名	长度	说明	单位	值范围	示例
采集时间	12	采用GMT时间，格式日月年时分秒			250314092030
经度	12	采集经度+半球符(E/W)		0-180	12122.11260E
纬度	12	采集纬度+半球符(N/S)		0-90	3113.16226N
角度	6	保留整数	度	0-360	235
速度	6	保留整数	km/h	0-999	83
海拔	5	保留整数	米	-9999至9999	120

- **other参数说明**
  其他需要上传的数据，允许自定义格式，最大长度不能超过1024个字节。

  - *自定义gsensor数据示例*
    25组连续的gsensor数据，每组数据必须必须包含gx,gy,gz。每组数据用“;”分隔。
    第一组数据和最后一组数据，需要加上GMT时间，每组数据间隔200ms。

  【示例数据】

```
other=250314092030235,-3.8185,-1.5312,8.5845;-3.9908,-1.4929,8.5367;-4.0195,-1.5216,8.6611;-3.9238,-1.5503,8.7664;-3.9238,-1.5408,8.6037;-3.8855,-1.5216,8.6994;-5.1488,1.9906,10.6421;-3.8281,2.4404,9.9052;-7.2447,2.6222,5.9335;-1.2345,3.5410,2.7849;-4.6416,7.3978,8.5367;-7.0628,5.6273,10.1732;-5.4550,0.9283,5.9240;-3.7132,2.7466,8.2400;-4.2109,-3.1582,13.0730;-5.7804,-0.2488,9.0726;-2.5744,1.9044,9.3693;-5.8666,1.9810,9.9339;-3.4931,2.0576,0.2201;-7.7041,0.4976,5.4933;-7.8380,0.4880,5.8857;-7.7710,0.6986,5.5507;-7.6849,0.7177,5.4837;250314092035235,-7.8476,0.4593,5.9527
```

参数名	长度	说明	单位	值范围	示例
gx	10	X轴加速度	米/秒平方	-30–+30	-3.8185
gy	10	y轴加速度	米/秒平方	-30–+30	-1.5312
gz	10	z轴加速度	米/秒平方	-30–+30	8.5845

-

- **bizid参数说明**
  将已播放或已读的信息的id提交给服务器，多个用“,”分隔。无数据用0表示。

  【示例数据】

  ```
  bizid=a3f1bcf0aeb31e11e38a7300266cf02f24,a3427664d8b31c11e3b45b00266cf02
  f24
  ```

2.8 返回参数

response对象中HEAD返回

语音信息返回结果

```
{
    "bizid": "a123123456789a987654321B123456789a",
    "level": 23,
    "type": 2,
    "longitude": 121.21237,
    "latitude": 31.23232,
    "distance": 500,
    "direction": [138,10],
    "speed": [50,80],
    "content": {
        "mmfiletype":"amr",
        "mmfilelength":15424
    }
}
```

response对象中BODY返回

```
Binary文件流
```

2.9 返回参数说明

返回参数	返回类型	参数说明	示例
bizid	string	信息唯一ID	a123123456789a987654321B123456789a
level	int	信息等级	23
longitude	double	最佳播放经度	121.21232
latitude	double	最佳播放纬度	31.23232
direction	array	最佳播放角度	138:10
distance	int	最佳播放距	500

		离	
speed	array	最佳播放速度	50:80
type	int	信息类型	1 语音 2 回复 3 群组
content	object	信息结构体	
mmfiletype	string	音频格式	amr
mmfilelength	int	媒体文件长度[单位字节]	15424
mmfile	binary	媒体文件二进制流	

3.交互接口

3.1接口说明

> 用户在终端的操作可以通过该接口提交到服务器。

3.2 URL

```
http://domain:port/feedback
```

3.3 http请求方式

```
POST
```

3.4 返回支持格式

```
http状态码
```

3.5 访问限制

```
访问级别：普通接口
频次限制：无限制
请求超时：30s
```

3.6 注意事项

> 请求必须用POST方式提交，并且注意采用multipart/form-data编码方式；

3.7 请求参数

参数名	长度	类型	说明	示例
imei	15	string	终端设备号	355688000000158
imsi	15	string	sim卡号	460013002605754
mod	5	string	终端型号	SG900
tokencode	10	string	config接口获取的参数	61HlullElt
gps		string	一组gps数据	250314092030,12122.11260E,3113.16226N,180,56,99999
type	2	string	交互类型	1
bizid	34	string	信息唯一id	a123123456789a987654321B123456789a
mmfiletype	5	string	音频文件格式（目前仅支持amr格式）	amr
mmfilekbps	3	int	音频码率，目前不大于8k	8
mmfilemins	3	int	音频时长 最长支持15s	9
mmfilelength		int	音频文件长度	15424

mmfile	binary	音频文件二进制流

- **gps参数说明**

  最新一组gps点数据，数据必须必须包含采集时间、经度、纬度、角度、速度、海拔。

  【示例数据】

  ```
  210314112755,11738.55807E,3058.35950N,-1,23,20;
  ```

- **码率、时长、文件长度说明**

  码率目前仅支持不大于8kHz
  音频时长最长15s，含bizid回复的音频最长8s，不足1s的标记为1s。
  文件流长度最大20480字节，即20KB。

- **交互类型说明**

操作	操作类型	操作说明	对应WEME终端操作
拨打电话1	1	拨打电话1前调用该接口	长按吐槽键
拨打电话2	2	拨打电话2前调用该接口	不实现
发送语音	3	发送一段语音	按一下吐槽键，再次按下发送或10秒后自动发送
语音命令	4	发送一段语音命令给服务器	空闲状态单击+
发送群组语音	5	标记该录音为群组录音	空闲状态单击++
+回复	6	快捷1回复当前播放的语音	按+键回复
++回复	7	快捷2回复当前播放的语音	按++键回复
关机	8	终端关机流程执行前	usb断电
语音回复	10	直接回复当前播放的信息或语音	听到tts或语音时直接回复，时间8s，需返回bizid

3.8 返回参数

```
服务端成功接收后返回HTTP状态码200
```

4 附录

4.1 WEME终端按键对照表

> 本对照表仅展示weme终端的交互实现方式。

按键	触发方式	空闲状态	录音状态	播放状态	被叫状态	通话状态	工程模式
+键	单击	语音指令	发送	快捷回复1	无	无	
	长按	音量+	音量+	音量+	接听	音量+	
吐槽键	单击	开始录音	发送	语音回复开始	接听	无	
	长按	拨打call1	无	无	无	无	
++键	单击	群组语音	发送	快捷回复2	挂断	挂断	
	长按	音量-	音量-	音量-	无	音量-	

音量调节：长按1s激活音量调节，每500ms变化一个音量级

4.2 “上传并获取信息”接口weme终端的实现参考

- 上传频率控制

时间轴	0s	5s	7s	12s	15s	20s	…
采集gps线程	开始采集	已采集五个点	已采集2个点	已采集7个点	已采集3个点	已采集8个点	
上传线程	第一次上传获取，无gps数据	第二次上传，5个gps点，清空已采集队列	上传完毕	第三次上传，7个gps点，清空已采集队列	上传完毕	第四次上传	

> 以上表格展示了在同一个时间轴上采集线程和上传线程的工作顺序。
> 采集线程config成功后，将不停的采集数据放入一个采集队列。
> 上传线程config成功后，将不停的以5s的间隔上传数据。5s间隔从上传并获取完信息后开始计算。
> 每次上传已采集队列中的gps，最多为10个点，如果超出10个将上传最新采集的是个点，其他的舍去。

- 信息播放控制

> 终端最多保存10条待播放的信息（tts和语音），队列按照获取的时间排序，采用先进先出的原则。
> 获取信息和播放信息独立进行，即不管是否已播放，获取是不间断进行的。如获取比播放速度快，仅保留最后获取的10条在队列中。
> 播放前需要进行等级排序和播放条件判断。保证等级高的先播放（数字越小等级越高），播放条件判断有利于更精准的信息服务。

- 信息播放条件判断逻辑

  逻辑

  > 1、判断分为4个指标判断
  > a、距离的判断
  > b、方向角的判断
  > c、越界判断
  > d、速度判断
  > 四个判断指标任何一个为false都将返回false，即不播放。
  > 2、距离判断依据（使用参数tts经纬度，当前经纬度）
  > tts经纬度和当前经纬度不为空，且两点间距离 tts经纬度为999或tts距离为-1，该指标为true；
  > 3、方向角的判断依据（使用参数tts方向角，tts误差角，当前方向角）
  > tts方向角-tts误差角<=当前方向角<=tts方向角+tts误差角，时该指标为true；
  > tts方向角或tts误差角任意一个为999，该指标为true；
  > tts方向角和tts误差角都不为999时，当前方向角为-1时，该指标为false；
  > 4、判断越界依据（使用参数tts经纬度，tts方向角，tts误差角，当前经纬度）
  > tts经纬度、tts方向角、tts误差角有一个为999，该指标为true；
  > tts经纬度、tts方向角、tts误差角都不为999，表示进入计算。
  > a、当前经纬度为999，该指标为false
  > b、tts方向角-90度<=当前点指向tts经纬度的角度<=tts方向角+90度，该指标为true，否则为false
  > 5、速度判断依据（使用参数tts速度，当前速度）
  > tts速度为空时，或tts速度高值和低值任意一个为空，该指标为true；
  > tts速度不为空，当前速度为空，该指标为false；
  > tts速度低值<=当前速度<=tts速度高值，时该指标为true；
  >
  > 以上四个指标顺序判断，四个值全部为true时播放tts，返回1，否则返回0，以上任意一个指标为false即可直接返回0。

  示例代码

  传入参数:
  tts纬度 获取TTS接口返回参数latitude 示例：31.232342 999表示空值
  tts经度 获取TTS接口返回参数longitude 示例：121.378892 999表示空值
  tts距离 获取TTS接口返回参数distance 示例：748 -1表示空值
  tts角度 获取TTS接口返回参数direction 示例：278:10 999表示空值
  tts速度 获取TTS接口返回参数speed 示例：78:90 999表示空值
  当前纬度 最新的gps数据中的纬度 没有值，传入999
  当前经度 最新的gps数据中的经度 没有值，传入999
  当前角度 最新的gps数据中的角度 没有值，传入999
  当前速度 最新的gps数据中的速度 没有值，传入999

  返回值:
  返回1表示需要播放，返回0表示不需要播放。

```c
#include<stdio.h>


#include<math.h>



typedef struct
{
    double dBPOI;
    double dLPOI;
    double dDisPOI;
    double dAnglePOI;
    double dErrAnglePOI;
    double dTTSMinSpeed;
    double dTTSMaxSpeed;

    double dBCurr;
    double dLCurr;
    double dAngleCurr;
    double dTTSCurrSpeed;
}StrPOIAndCurr;


void BL2XY(double dB, double dL, double *dX, double *dY)
{
        double m_PI = 3.1415926;
        double dBeltNumber = floor((dL + 1.5) / 3.0);
        double dL0 = dBeltNumber * 3.0;

        double a = 6378137;
        double b = 6356752.3142;
        double k0 = 1;
        double FE = 500000;

        double e1 = sqrt(1 - pow(b / a, 2));
        double e2 = sqrt(pow(a / b, 2) - 1);

        double T = 0.0;
        double C = 0.0;
        double A = 0.0;
        double M = 0.0;
        double N = 0.0;
        double mgs1 = 0.0;
        double mgs2 = 0.0;
        double mgs3 = 0.0;

        T = pow(tan(dB), 2);
        dB = dB * m_PI / 180.0;

        C = e2 * e2 * pow(cos(dB), 2);

        dL = dL * m_PI / 180.0;
        dL0 = dL0 * m_PI / 180.0;
        A = (dL - dL0) * cos(dB);

        M = (1 - pow(e1, 2) / 4.0 - 3.0 * pow(e1, 4) / 64.0 - 5.0 * pow(
e1, 6) / 256.0) * dB;
        M = M - (3.0 * pow(e1, 2) / 8.0 + 3.0 * pow(e1, 4) / 32.0 + 45.0
* pow(e1, 6) / 1024.0) * sin(dB * 2);
        M = M + (15.0 * pow(e1, 4) / 256.0 + 45.0 * pow(e1, 6) / 1024.0)
```

```
* sin(dB * 4);
        M = M - (35.0 * pow(e1, 6) / 3072.0) * sin(dB * 6);
        M = a * M;

        N = a / pow(1.0 - pow(e1, 2) * pow(sin(dB), 2), 0.5);

        mgs1 = pow(A, 2) / 2.0;
        mgs2 = pow(A, 4) / 24.0 * (5.0 - T + 9.0 * C + 4.0 * pow(C, 2));
        mgs3 = pow(A, 6) / 720.0 * (61.0 - 58.0 * T + pow(T, 2) + 270 *
C - 330.0 * T * C);
        *dY = M + N * tan(dB) * (mgs1 + mgs2) + mgs3;
        *dY = *dY * k0;

        mgs1 = A + (1.0 - T + C) * pow(A, 3) / 6.0;
        mgs2 = (5.0 - 18.0 * T + pow(T, 2) + 14.0 * C - 58.0 * T * C) *
pow(A, 5) / 120.0;
        *dX = (mgs1 + mgs2) * N * k0 + FE;
}

double GetCentralMeridian(double dL)
{
    double dBeltNumber = floor((dL + 1.5) / 3.0);
    double dL0 = dBeltNumber * 3.0;

    return dL0;
}

double CalculateTwoPointsDistance(double dLongitude1, double dX1, double
dY1,
                                  double dLongitude2, double dX2, double
dY2)
{
    double dL01 = GetCentralMeridian(dLongitude1);
    double dL02 = GetCentralMeridian(dLongitude2);
    double dCentralDelta = fabs(dL02 - dL01);
    double dDeltaX = 0.0;
    double dDistance = 0.0;

    int m_GPSEnlargeScale = 100;

    if (dLongitude2 > dLongitude1)
        dDeltaX = dX2 - dX1 + dCentralDelta * m_GPSEnlargeScale * 954;
    else
        dDeltaX = dX1 - dX2 + dCentralDelta * m_GPSEnlargeScale * 954;

    dDistance = sqrt(pow(dDeltaX, 2) + pow(dY2 - dY1, 2));

    return dDistance;
}

double CalculateTwoPointsAngle(double dX1, double dY1, double dX2, doubl
e dY2)
{
    double dTolerance = 0.00001;
    double m_PI = 3.1415926;

    if (fabs(dX1 - dX2) < dTolerance)
    {
        if (dY2 > dY1)
            return 0.0;
        else
            return 180.0;
    }
```

```
    else if (fabs(dY1 - dY2) < dTolerance)
    {
        if (dX2 > dX1)
            return 90.0;
        else
            return 270.0;
    }
    else
    {
        double dResult = (atan((dX2 - dX1) / (dY2 - dY1))) * 180.0 / m_P
I;
        if ((dX1 < dX2) && (dY1 > dY2))
            return dResult + 180.0;
        else if ((dX1 > dX2) && (dY1 > dY2))
            return dResult + 180.0;
        else if ((dX1 > dX2) && (dY1 < dY2))
            return dResult + 360.0;
        else
            return dResult;
    }
}

int CheckDis( StrPOIAndCurr strPOICurr)
{
    int iResult = 0;
    double dis;
    double dXPOI,dYPOI,dXCurr,dYCurr;

    if ( strPOICurr.dDisPOI != -1 )
    {
        BL2XY(strPOICurr.dBPOI, strPOICurr.dLPOI, &dXPOI, &dYPOI);
        BL2XY(strPOICurr.dBCurr, strPOICurr.dLCurr, &dXCurr, &dYCurr);

        dis = CalculateTwoPointsDistance(strPOICurr.dLCurr, dXCurr, dYCu
rr,strPOICurr.dLPOI, dXPOI, dYPOI);
        printf("CalculateTwoPointsDistance: %lf\n", dis);

        if ( dis < strPOICurr.dDisPOI )
            iResult = 1;
    }
    else
        iResult = 1;

    return iResult;
}

int CheckAngle( StrPOIAndCurr strPOICurr)
{
    int iResult = 0;
    double angle,dAngleSub,dAngleAdd;
    double dXPOI,dYPOI,dXCurr,dYCurr;

    if ( strPOICurr.dAnglePOI != 999  && strPOICurr.dErrAnglePOI != 999
)
    {
            if ( strPOICurr.dAngleCurr != 999)
            {
                BL2XY(strPOICurr.dBPOI, strPOICurr.dLPOI, &dXPOI, &dYPOI
);
                BL2XY(strPOICurr.dBCurr, strPOICurr.dLCurr, &dXCurr, &dY
Curr);

                angle = CalculateTwoPointsAngle(dXPOI, dYPOI,dXCurr, dYC
```

```
urr);
                printf("CalculateTwoPointsAngle: %lf\n", angle);

                dAngleSub = strPOICurr.dAnglePOI  - strPOICurr.dErrAngle
POI;

                if ( dAngleSub >= 0 )
                {
                    dAngleAdd = strPOICurr.dAnglePOI  + strPOICurr.dErrA
nglePOI;

                    if ( dAngleAdd <= 360)
                    {
                        if (strPOICurr.dAngleCurr >= dAngleSub && strPOI
Curr.dAngleCurr <= dAngleAdd )
                        {
                            double angle1 = strPOICurr.dAnglePOI + 90;
                            double angle2 = strPOICurr.dAnglePOI - 90;

                            if ( angle1 <= 360 && angle2 >= 0 )
                            {
                                if ( angle >= angle2 && angle <= angle1
)
                                    iResult = 0;
                                else
                                    iResult = 1;
                            }

                            if ( angle1 > 360 )
                            {
                                if ( ( angle >= angle2 && angle <= 360 )
|| ( angle <= ( angle1 - 360 ) && angle >= 0 ) )
                                    iResult = 0;
                                else
                                    iResult = 1;
                            }

                            if ( angle2 < 0 )
                            {
                                if ( ( angle >= ( angle2 + 360 ) && angl
e <= 0 ) || ( angle <= angle1 && angle >= 0 ) )
                                    iResult = 0;
                                else
                                    iResult = 1;
                            }
                        }
                    }
                    else
                    {
                        if ( ( strPOICurr.dAngleCurr >= dAngleSub && str
POICurr.dAngleCurr <= 360) || ( strPOICurr.dAngleCurr >= 0 && strPOICurr
.dAngleCurr <= ( dAngleAdd - 360) ) )
                        {
                            double angle1 = strPOICurr.dAnglePOI + 90;
                            double angle2 = strPOICurr.dAnglePOI - 90;

                            if ( angle1 < 360 && angle2 > 0 )
                            {
                                if ( angle >= angle2 && angle <= angle1
)
                                    iResult = 0;
                                else
                                    iResult = 1;
```

```
                            }

                            if ( angle1 > 360 )
                            {
                                if ( ( angle >= angle2 && angle <= 360 )
|| ( angle <= ( angle1 - 360 ) && angle >= 0 ) )
                                    iResult = 0;
                                else
                                    iResult = 1;
                            }

                            if ( angle2 < 0 )
                            {
                                if ( ( angle >= ( angle2 + 360 ) && angl
e <= 0 ) || ( angle <= angle1 && angle >= 0 ) )
                                    iResult = 0;
                                else
                                    iResult = 1;
                            }
                        }
                    }
                }
                else // -
                {
                    dAngleAdd = strPOICurr.dAnglePOI  + strPOICurr.dErrA
nglePOI;

                    if ( dAngleAdd <= 360)
                    {
                        if ( ( strPOICurr.dAngleCurr >= ( 360 + dAngleSu
b ) && strPOICurr.dAngleCurr <= 360 ) || ( strPOICurr.dAngleCurr >= 0 &&
strPOICurr.dAngleCurr <= dAngleAdd ) )
                        {
                            double angle1 = strPOICurr.dAnglePOI + 90;
                            double angle2 = strPOICurr.dAnglePOI - 90;

                            if ( angle1 < 360 && angle2 > 0 )
                            {
                                if ( angle >= angle2 && angle <= angle1
)
                                    iResult = 0;
                                else
                                    iResult = 1;
                            }

                            if ( angle1 > 360 )
                            {
                                if ( ( angle >= angle2 && angle <= 360 )
|| ( angle <= ( angle1 - 360 ) && angle >= 0 ) )
                                    iResult = 0;
                                else
                                    iResult = 1;
                            }

                            if ( angle2 < 0 )
                            {
                                if ( ( angle >= ( angle2 + 360 ) && angl
e <= 0 ) || ( angle <= angle1 && angle >= 0 ) )
                                    iResult = 0;
                                else
                                    iResult = 1;
                            }
                        }
```

```
                    }
                }
            }
            else
                iResult = 0;
    }
    else
        iResult = 1;

    return iResult;
}

int CheckSpeed( StrPOIAndCurr strPOICurr)
{
    int iResult = 0;
    double dis;
    double dXPOI,dYPOI,dXCurr,dYCurr;

    if ( strPOICurr.dTTSMinSpeed == 999 || strPOICurr.dTTSMaxSpeed == 99
9 )
        iResult = 1;
    else
    {
        if ( strPOICurr.dTTSCurrSpeed == 999 )
            iResult = 0;
        else if ( strPOICurr.dTTSCurrSpeed <= strPOICurr.dTTSMaxSpeed &&
strPOICurr.dTTSCurrSpeed >= strPOICurr.dTTSMinSpeed )
            iResult = 1;
        else
            iResult = 0;
    }

    return iResult;
}

int IsPlayTTS( StrPOIAndCurr strPOICurr)
{
    int iResult = 0;

    if ( strPOICurr.dBPOI != 999 && strPOICurr.dLPOI != 999 )
    {
        if( strPOICurr.dBCurr == 999 || strPOICurr.dLCurr == 999 )
            iResult = 0;
        else
            iResult = CheckDis(strPOICurr) && CheckAngle(strPOICurr)&& C
heckSpeed(strPOICurr);
    }
    else
    {
        //if ()
        iResult = CheckAngle(strPOICurr)&& CheckSpeed(strPOICurr);//1;
    }

    return iResult;
}
```