简介

IotVideoSDK是P2P模式下的智能家居平台工具集，PC版IotVideoSDK集有监控、录像、消息通讯等功能。

快速开始

- 平台支持

平台	SDK描述	Demon版本	SDK及Demon源码
PC	win64，win7，win10	https://github.com/GWTimes/IotVideo-PC-Demon	https://github.com/GWTimes/IoTVideo-PC

demon是基于QT5.9版本进行开发。

第一步：接入准备

获取accessId与accessToken，用户自有账号体系可以采用云云对接的方式实现账户体系相关业务。根据APP的《厂商云对接IotVideo平台接口》

SDK初始化

1、实现SDK类的事例化

```
 m_pIotVideoSdk = IoTVideoSdk::getInstance();
```

2、注册消息回调函数

```
 m_pIotVideoSdk->addModelDataListener(std::bind(&Widget::ModelDataResp,

this,std::placeholders::_1,std::placeholders::_2));
```

3、用户账号接入服务器

```
 m_pIotVideoSdk->Register(accessId,accessToken);
```

监控播放器创建

注明：如果同时监控多个设备，则实例化多个监控播放器类对象。

1、监控播放器类对象实例化

```
m_pIotVideoPlayer = new IoTVideoPlayer;
```

2、注册监控播放器对应的回调函数或类对象

- 实现渲染功能对象，注册对象实例

```
m_pVideoRender = new VideoRenderCase(hShowWind);
m_pIotVideoPlayer->setVideoRender(m_pVideoRender);
```

- 实现声音播放功能对象，注册对象实例

```
m_pAudioRneder = new AudioRenderCase;
m_pIotVideoPlayer->setAudioRender(m_pAudioRneder);
```

- 注册监控播放器的状态回调与错误回调函数

```
m_pIotVideoPlayer-
>setStatusListener(std::bind(&Widget::playerStatusListener,this,std::placeholder
s::_1));
m_pIotVideoPlayer-
>setErrorListener(std::bind(&Widget::playerErrorListener,this,std::placeholders:
:_1));
```

3、开始监控设备

```
m_pIotVideoPlayer->setDataResource(u64DevID, 1,VIDEO_DEFINITION_HD);
m_pIotVideoPlayer->play();
```

4、停止监控

```
m_pIotVideoPlayer->stop();
```

物模型消息通信

1、获取物模型

```
IoTRequest getModelReq(sDevID);
    getModelReq.IoTGetModelData(sGetModeType)
            .IoTOnTimeout([qDevID,this](std::string msg){

                emit this->getModelDataTimeout_sig(qDevID);
             })
            .IoTError([qDevID,this](std::string msg,int err){
                QString qsMsg = QString::fromStdString(msg);
                emit this->getModelDataError_sig(qDevID,qsMsg,err);
            })
            .IoTSuccess([qDevID,this](std::string msg){
                printf("sDevID=%s
msg=%s\n",qDevID.toStdString().c_str(),msg.c_str());
                QString qsMsg = QString::fromStdString(msg);
                emit this->getModelDataSuccess_sig(qDevID,qsMsg);
            });
```

2、设置物模型

```
IoTRequest setModelReq(sDevID);
    std::string sJdata = qsSetModelText.toStdString();
```

```
    std::string sLeaf(sSetModeType);
    setModelReq.IoTSendModelData(sLeaf,sJdata)
            .IoTOnTimeout([qDevID,this](std::string msg){
                emit this->setModelDataTimeout_sig(qDevID);
            })
            .IoTError([qDevID,this](std::string msg, int err){
                QString qsMsg = QString::fromStdString(msg);
                emit this->setModelDataError_sig(qDevID,qsMsg,err);
            })
            .IoTSuccess([qDevID,this](std::string msg){
                QString qsMsg = QString::fromStdString(msg);
                emit this->setModelDataSuccess_sig(qDevID,qsMsg);
            })
            ;
```

注明：所有接口的详细使用说明，请查看，SDK类接口：windows SDK接入、消息接口,多媒体接口：多媒体