**总线客户端**

**开发简介**

## 一 概述

本文档介绍总线客户端的开发API，开发语言为C++。开发API主要包括数据包API和客户端API以及相关结构体定义。

## 二 数据包API

数据包由报文头和记录组成，报文头中需要设置功能号以区分不同业务，记录由多个字段组成，字段包括字段名和对应的取值。

数据的组包和解包使用统一的数据包API来完成，相关功能通过调用CPackage类中的方法来实现，具体可参考附录中的package.h文件。

### 组包

数据包的组包主要有：设置包的报文头和功能号，增加记录以及设置记录中不同字段的值。组包接口如下：

//设置报文头

void SetPackageHeader(PACKAGEHEADER \*pHeader) ;

//设置功能号

void SetFuncNo(int nFuncNo) ;

//增加一条记录(空)

int AppendBlankRecord(ERROR\_INFO &errInfo) ;

//设置字段的值(char \*类型)

bool SetFieldValue(int index,char \*fieldname,char \*value,ERROR\_INFO &errInfo);

//设置字段的值(string类型)

bool SetFieldValue(int index,char \*fieldname,string &value,ERROR\_INFO &errInfo);

//设置字段的值(int类型)

bool SetFieldValue(int index,char \*fieldname,int value,ERROR\_INFO &errInfo);

//设置字段的值(double类型)

bool SetFieldValue(int index,char \*fieldname,double value,ERROR\_INFO &errInfo);

//设置strvector类型的数据

bool SetFieldValue(int index,char \*fieldname,vector<string> &value,ERROR\_INFO &errInfo);

//设置intvector类型的数据

bool SetFieldValue(int index,char \*fieldname,vector<int> &value,ERROR\_INFO &errInfo);

//设置doublevector类型的数据

bool SetFieldValue(int index,char \*fieldname,vector<double> &value,ERROR\_INFO &errInfo);

### 解包

数据包的解包主要有获取报文头，获得记录的个数、记录中字段的个数、指定字段名对应的取值，另外也可获得记录中的字段信息。

//获得包头

PACKAGEHEADER \*GetPackageHeader();

//获得字段数量

int GetFieldCount();

//获得记录数

int GetRecordCount();

//获得字符串类型的数据

bool GetFieldValue(int recordindex,char \*fieldname,string &RetStr,ERROR\_INFO &errInfo);

//获得int类型的数据

bool GetFieldValue(int recordindex,char \*fieldname,int &Ret,ERROR\_INFO &errInfo);

//获得double类型的数据

bool GetFieldValue(int recordindex,char \*fieldname,double &Ret,ERROR\_INFO &errInfo);

//获得strvector类型的值

bool GetFieldValue(int recordindex,char \*fieldname,vector <string> &RetStr,ERROR\_INFO &errInfo);

//获得intvector类型的值

bool GetFieldValue(int recordindex,char \*fieldname,vector <int> &RetStr,ERROR\_INFO &errInfo);

//获得doublevector类型的值

bool GetFieldValue(int recordindex,char \*fieldname,vector <double> &RetStr,ERROR\_INFO &errInfo);

//获得字段信息

vector<FIELDINFO> \*GetFieldInfo();

## 三 客户端API

客户端API由通信接口（CClientApi类）和回凋接口（CClientApiSpi类）两部分。通信接口完成通信对象的创建和初始化，发送请求、数据包或文件到指定的服务节点。回凋接口用于向调用通信接口的对象发送状态等信息。

### 通信接口

通信接口CClientApi类中的方法具体可参考附录中clientapi.h文件。通信接口提供通信对象的创建和初始化，注册回凋接口，连接总线并登陆。

//创建api实例

static CClientApi \*CreateClientApi();

//初始化

int Init(ERROR\_INFO &ErrMsg);

//注册回调接口

void RegisterSpi(CClientApiSpi \*pGeneralSpi);

//连接到总线

bool Connect(char \*routerip, int port, ERROR\_INFO &ErrMsg);

//登陆

bool Login(char \*username, char \*password, ERROR\_INFO &ErrMsg);

通信对象可以创建数据包，在发送完数据包后需要删除不再使用的数据包。

//创建一个数据包

static CPackage \* CreatePackage();

//删除数据包

static void ReleasePackage(CPackage \* pPackage);

通信对象完成向服务节点发送文件、数据和请求的功能。

//发送文件到指定的总线节点

int SendFileToService(char \*filename,int seriveno,ERROR\_INFO &ErrMsg);

//发送数据包到指定的总线节点

int SendDataToService(int nSeriveNo,CPackage \*pData,ERROR\_INFO &ErrMsg);

//发送请求到总线节点

int SendRequestToService(int nSeriveNo,int nReqNo,CPackage \*Para,ERROR\_INFO &ErrMsg);

不再使用通信对象时，需要注销登陆，并释放资源。

//登出

bool Logout(ERROR\_INFO &ErrMsg) ;

//释放资源

void Release() ;

### 回调接口

回调接口CClientApiSpi类中的方法具体可参考附录中clientapi.h文件。回调接口中有三个方法需要重新实现，其中OnNtyStatus向调用通信接口的对象发送某个业务的状态信息，OnRcvRspData向调用通信接口的对象发送数据包，OnRcvFile在收到文件后向调用通信接口的对象发送文件接收完成消息。

//状态回调显示

virtual void OnNtyStatus(NTY\_STATUS\_INFO \*pStatus){};

//数据回调

virtual void OnRcvRspData(CPackage \*pPackage,int &nEventHandle){};

//收到文件后回调

virtual void OnRcvFile(char \*FileName){};

## 四 相关结构体

在数据包和客户端API中用到的数据结构如：报文头PACKAGEHEADER，错误信息提示ERROR\_INFO等结构体的定义可参考附录中BusStruct.h文件。

## 五 开发举例

### 实现回调接口，

class CApiRsp :public CClientApiSpi

{

public:

CApiRsp();

~CApiRsp();

void OnNtyStatus(NTY\_STATUS\_INFO \*pStatus)

{

//具体实现

}

void OnRcvRspData(CPackage \*pPackage,int &nEventHandle)

{

//以下为处理登陆的实现举例

PACKAGEHEADER \*pHeader = pPackage->GetPackageHeader();

if(pHeader->nFuncType == RSPTYPE )//如果是应答包

{

switch(pHeader->nFuncNo)

{

case -3000://登陆应答

{

ERROR\_INFO errInfo;

int nReturnCode ;

bool bRet = pPackage->GetFieldValue(0,"msgid",nReturnCode,errInfo);

if(bRet == true)

{

if(nReturnCode == 0)

{

MessageBox.Show("登陆成功!!!");

}

else

{

string msg;

bRet = pPackage->GetFieldValue(0,"msginfo",msg,errInfo);

MessageBox.Show("登陆失败:%s!!!",msg.c\_str());

}

}

}

break;

defaut:

break;

}//switch end

}//if end

}

void OnRcvFile(char \*FileName)

{

//具体实现

}

};

### 调用通信接口

1. **创建通信对象并初始化**

CClientApi pApi = CClientApi::CreateClientApi();

pApi->Init(ErrInfo);

1. **注册回调**

CApiRsp m\_pRsp = new CApiRsp();

pApi->RegisterSpi(m\_pRsp);

1. **连接并登陆**

pApi->Connect(IPString,PortInt,ErrInfo);

pApi->Login(UsernameString,PasswordString,ErrInfo);

1. **创建数据包，添加纪录，设置记录字段名和取值**

CPackage \*pPackage = pApi->CreatePackage();

int iRec = pPackage->AppendBlankRecord(ErrInfo);

pPackage->SetFieldValue(iRec,FieldNameString,fieldValue, ErrInfo);

1. **发送数据包**

int nRet = pApi->SendRequestToService(ServiceNo,RequestNo,pPackage, ErrInfo);

1. **释放数据包**

pApi->ReleasePackage(pPackage);

1. **注销登陆并释放资源**

pApi->Logout(ERROR\_INFO &ErrMsg) ;

pApi->Release() ;

## 六 附录

数据包CPackage类头文件：

客户端CClientApi和CClientApiSpi的头文件

相关结构体的定义文件