IEC102
主站开发包接口使用手册

版本：V2.0

深圳天勺电力软件有限公司

中国. 深圳

目录

1.开发包简介

本开发包适用于 IEC102 规约，基于主站应用程序程序开发。规约主要参考 IEC 61870-5 系列文件进行编写。

其中注册的函数，只需要注册一次即可。

节约了客户从零研发的周期和成本。

编译说明：

LINUX 系统：在 szts_IEC102_DataFile.h 中，注释 #define IEC102_IS_WINDOWS 宏定义,如果是 windows 则打开。

CMakeLists.txt 是生成动态库的文件，需要重新编译，可参考此文件。

2. 接口描述

2.1 打开串口连接

函数声明	int TS_DLL_EXPORT TS_IEC102_ConnectToCom(const char* svrName, IEC_BYTE comAddre = 0x01,const char* portNo = "COM2", IEC102_UINT baud = 9600, char parity = 'N', IEC_UINT databits = 8, IEC_UINT stopsbits = 1);
功能	打开串口，初始化通信参数
参数	const char *svrName：连接从站的名字，作为当前连接标识使用； portNo:串口号，默认为 COM2,如果是 linux 一般为/dev/pts/2 形式 baud:波特率，默认 9600 parity:校验位 databits:数据位，默认 8 位 stopsbits:停止位 comAddre:公共地址
返回值	连接成功返回 true，否则为 false;

示例	const char *svrName = "test";//从站标识 程序自定义
	//跟当前连接进行绑定 后续此字符串用处类似
	int comNo = 2; //串口号
	int comAddr = 1; //公共地址
	ret = TS_IEC102_ConnectToCom(svrName, comAddr, comNo);

Tip：当连接成功后，后续命令才能发送成功

2.2 关闭串口连接

函数声明	void TS_DLL_EXPORT TS_IEC102_DisConnectCom(const char* svrName);
功能	关闭串口，退出数据接收、解析线程
参数	const char *svrName: 连接从站的名字，作为当前连接标识使用
返回值	NULL

示例	const char* svrName = = "test";
	//关闭串口
	TS_IEC102_DisConnectCom(const char* svrName)

2.3 发送链路状态查询

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_Ask_Link_Status(const char* svrName);
功能	链路状态查询
参数	const char *svrName：连接从站的名字，作为当前连接标识使用
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char* svrName = = "test";
	//查询链路状态
	TS_IEC102_Send_Ask_Link_Status(svrName)

注意：一般通讯流程如下（实际可根据现场进行调整）：

Step1:查询链路状态；

Step2:链路复位；

Step3:初始化完成校验（实际为问一级数据），当从站回复初始化确认后，进行数据查询。

2.4 注册 查询链路状态响应 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_ASK_LINK_STATUS_Resp (Svr_IEC102_ASK_LINK_STATUS_Resp callback);
功能	获取链路状态查询的响应，只注册一次即可
参数	函数指针，具体参数看结构体定义
返回	void

值	

示例	//查询链路状态响应
	void Svr_IEC102_ASK_LINK_STATUS_Resp_CallBack(const char *server_name, struct IEC102_ASK_LINK_STATUS_Resp* resp){
	printf("Svr_IEC102_ASK_LINK_STATUS_Resp_CallBack\n");
	}
	//注册查询链路状态响应回调
	Reg_Svr_IEC102_ASK_LINK_STATUS_Resp(Svr_IEC102_ASK_LINK_STATUS_Resp_CallBack);

2.5 发送链路复位

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_RetSet_Link(const char* svrName);
功能	发送链路复位
参数	const char *svrName：连接从站的名字，作为当前连接标识使用
返回值	发送成功字节数，发送成功大于0，否则失败

示例	const char *svrName = "test";　　//从站标识 程序自定义
	//跟当前连接进行绑定
	//查询链路状态
	int ret = TS_IEC102_Send_Ask_Link_Status(svrName);

2.6 注册链路复位响应回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_RESET_LINK_Resp(Svr_IEC102_RESET_LINK_Resp callback);
功能	发送链路复位的回调相应
参数	函数指针
返回值	NULL

示例	//链路复位响应
	void Svr_IEC102_RESET_LINK_Resp_CallBack(const char *server_name, struct IEC102_RESET_LINK_Resp* resp)
	{

	printf("Svr_IEC102_RESET_LINK_Resp_CallBack\n");
	}
	//注册链路复位响应回调
	Reg_Svr_IEC102_RESET_LINK_Resp(Svr_IEC102_RESET_LINK_Resp_CallBack);

2.7 初始化校验

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_Check_Init(const char* svrName);
功能	校验初始化是否完成，实际发送为问一级数据帧
参数	const char *svrName：连接从站的名字，作为当前连接标识使用
返回值	发送成功字节数，发送成功大于 0，否则失败
示例	const char *svrName = "test";　　//从站标识 程序自定义跟当前连接进行绑定 int ret = TS_IEC102_Send_Check_Init(svrName);

2.8 注册 初始化完成响应 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Init_Resp (Svr_IEC102_Init_Resp callback);
功能	注册初始化完成响应回调，发送 2.8 后响应
参数	函数指针
返回值	无

示例	void Svr_IEC102_Init_Resp_CallBack(const char *server_name, IEC102_Init_Resp* resp)
	{
	printf("Svr_IEC102_Init_Resp_CallBack\n");
	printf("Svr_IEC102_Init_Resp_CallBack resp->byCtrlZoneHex:%d\n", resp->byCtrlZoneHex);
	printf("Svr_IEC102_Init_Resp_CallBack resp->cot:%d\n", resp->cot);
	printf("Svr_IEC102_Init_Resp_CallBack resp->funCode:%d\n", resp->funCode);
	printf("Svr_IEC102_Init_Resp_CallBack resp->isTest:%d\n", resp->isTest);
	printf("Svr_IEC102_Init_Resp_CallBack resp->POrN:%d\n", resp->POrN);
	if(resp->POrN == 0 && resp->cot == ENUM_102_REQ_OR_BEREQ){

	//回复正响应 并且传送原因为预期4
	}
	}
	//注册初始化完成响应回调
	Reg_Svr_IEC102_Init_Resp (Svr_IEC102_Init_Resp_CallBack);

2.9 读远方终端当前的系统时间

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_Get_RTU_Sys_Time(const char* svrName);
功能	发送读远方终端当前的系统时间
参数	const char *svrName：连接从站的名字，作为当前连接标识使用
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char *svrName = "test"; //从站标识 程序自定义
	//跟当前连接进行绑定
	int ret = TS_IEC102_Send_Get_RTU_Sys_Time(svrName);

具体流程:可参考 demo 中的 GetRTUSysTime(const char * svrName);

2.10 注册单字节帧 0xE5 回复 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_SINGLE_E5_Resp(Svr_IEC102_SINGLE_E5_Resp callback);
功能	获取单字节帧 0xE5 回复 回调
参数	函数指针
返回值	NULL

示例	//单字节帧 0xE5 数据回复
	void Svr_IEC102_SINGLE_E5_Resp_CallBack(const char *server_name, struct IEC102_SINGLE_E5_Resp *resp)
	{
	printf("Svr_IEC102_RESET_LINK_Resp_CallBack\n");
	//接受到单字符E5 //在此添加条件判断
	}
	//注册单字节帧 0xE5回复 回调
	Reg_Svr_IEC102_SINGLE_E5_Resp(Svr_IEC102_SINGLE_E5_Resp_CallBack);

2.11 问一级数据

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_Ask_Level_Data(const char* svrName);
功能	发送问一级数据
参数	const char *svrName：连接从站的名字，作为当前连接标识使用
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char *svrName = "test"; //从站标识 程序自定义 跟当前连接进行绑定
	int ret = TS_IEC102_Send_Ask_Level_Data(svrName);

注意:具体的调用时机，需要根据实际需求进行性调用；

2.12 注册 读电能累计量数据终端设备的当前系统时间 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type103_Resp(Svr_IEC102_Type103_Resp callback);
功能	注册读电能累计量数据终端设备的当前系统时间 回调
参数	函数指针
返回值	无

示例	// CON<103> := C_TI_NA_2 读电能累计量数据终端设备的当前系统时间
	void Svr_IEC102_Type103_Resp_CallBack(const char *server_name, struct IEC102_Type103_Resp *resp){
	printf("Svr_IEC102_Type103_Resp_CallBack byFunCode:%02X\n", resp->pCtrlZone->byFunCode);
	//可根据需求进行调整 一般为8
	if(resp->pCtrlZone->byFunCode == ENUM_102_CTRL_FUNCODE_DATARESP_REQ){
	//此处对功能码进行校验 序号8 以要求访问位响应(控制站) 用户数据（被控站）
	}else{
	//可自定义解析原因等
	}
	}
	//注册读电能累计量数据终端设备的当前系统时间 回调
	Reg_Svr_IEC102_Type103_Resp(Svr_IEC102_Type103_Resp_CallBack);

2.13 注册 电能累计量数据终端设备目前的系统时间 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type72_Resp(Svr_IEC102_Type72_Resp callback);

功能	电能累计量数据终端设备目前的系统时间 信息获取
参数	函数指针
返回值	NULL

示例	// 72:M TI TA 2 电能累计量数据终端设备目前的系统时间
	void Svr_IEC102_Type72_Resp_CallBack(const char *server_name, struct IEC102_Type72_Resp *resp)
	{
	printf("Svr_IEC102_Type72_Resp_CallBack byFunCode:%d\n", resp->pCtrlZone->byFunCode);
	printf("Svr_IEC102_Type72_Resp_CallBack str_time:%s\n", resp->pTimeB->str_time);
	}
	//注册电能累计量数据终端设备目前的系统时间 回调
	Reg_Svr_IEC102_Type72_Resp(Svr_IEC102_Type72_Resp_CallBack);

2.14 电能累计量数据终端系统时间(时间同步命令）

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_SysTime_Sync(const char* svrName);
功能	电能累计量数据终端系统时间（时间同步命令）
参数	const char *svrName：连接从站的名字，作为当前连接标识使用
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char *svrName = "test"; //从站标识 程序自定义 跟当前连接进行绑定
	int ret = TS_IEC102_Send_SysTime_Sync(svrName);

注意:具体流程，可参考 demo 的 SyncTime(const char * svrName)。

2.15 注册 电能累计量数据终端系统时间同步命令(发送对时命令) 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type128_Resp(Svr_IEC102_Type128_Resp callback);
功能	电能累计量数据终端系统时间同步命令 数据获取
参数	函数指针
返回	空

值	

示例	//128:C_SYN_TA_2 电能累计量数据终端系统时间同步命令(发送对时命令)
	void Svr_IEC102_Type128_Resp_CallBack(const char *server_name, struct IEC102_Type128_Resp *resp){
	printf("void Svr_IEC102_Type128_Resp_CallBack time:%s\n", resp->pTimeB->str_time);
	if(resp->pCot->cot == ENUM_102_DEAD_ACT){
	//g_typeId128_Dead_Act = 1;
	}
	if(resp->pCot->cot == ENUM_102_ACT_TERM){
	//g_typeId128_Dead_Act = 1;
	}
	}
	//128:C_SYN_TA_2 电能累计量数据终端系统时间同步命令(发送对时命令)
	Reg_Svr_IEC102_Type128_Resp(Svr_IEC102_Type128_Resp_CallBack);

2.16 读选定时间范围的带时标的单点信息记录

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_GetTimeRangeSPInfo(const char* svrName,IEC102_BYTE Minutes = 5,IEC102_BYTE recordAddr = 0x33);
功能	读选定时间范围的带时标的单点信息记录
参数	const char *svrName：连接从站的名字，作为当前连接标识使用 Minutes:分钟，开始时间会当前系统时间上减这个值，最大为 60 分钟 recordAddr: 记录地址
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char *svrName = "test"; //从站标识 程序自定义 跟当前连接进行绑定
	int ret = TS_IEC102_Send_GetTimeRangeSPInfo(svrName);

注意：具体流程，看参考 demo 中的 int GetSPInfo(const char * svrName)

2.17 注册 读选定时间范围的带时标的单点信息记录 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type102_Resp(Svr_IEC102_Type102_Resp callback);
功能	获取 读选定时间范围的带时标的单点信息记录

参数	函数指针
返回值	Void

示例	//102:= C_SP_NB_2读选定时间范围的带时标的单点信息记录 struct
	//IEC102_Type102_Resp
	void Svr_IEC102_Type102_Resp_CallBack(const char *server_name,
	struct IEC102_Type102_Resp *resp)
	{
	//
	printf("Svr_IEC102_Type102_Resp_CallBack
	resp->pCot->cot:%d\n", resp->pCot->cot);
	if(resp->pCot->cot == ENUM_102_ACT_CON){
	//激活确认
	//g_typeId102 = 1;
	}
	else if(resp->pCot->cot == ENUM_102_ACT_TERM){
	//激活终止
	//g_typeId102_Dead_Act = 1;
	}
	//SetRecvSatus(1);
	}
	// 注册读选定时间范围的带时标的单点信息记录 回调
	Reg_Svr_IEC102_Type102_Resp(Svr_IEC102_Type102_Resp_CallBack);

2.19 报文解析（接收）回调

函数声明	void TS_DLL_EXPORT Reg_Svr_parseReceMsg(Svr_parseReceMsg callBack);
功能	报文解析回调
参数	函数指针
返回值	Void

示例	//报文解析回调
	void Svr_parseReceMsg_CallBack(const char *server_name,
	struct MsgInfo *msgInfo){
	int len = msgInfo->fl;
	printf("recv [%d]:", len);
	for (int i = 0; i < len; i++)
	{
	//
	printf("%02X ", msgInfo->msg[i]);

	}
	printf("\n");
	}
	//报文解析回调
	Reg_Svr_parseReceMsg(Svr_parseReceMsg_CallBack);

2.20 报文发送回调

函数声明	void TS_DLL_EXPORT Reg_Svr_parseSendMsg(Svr_parseSendMsg callBack);
功能	报文发送回调
参数	函数指针
返回值	空

示例	void Svr_parseSendMsg_CallBack(const char *server_name, struct MsgInfo *msgInfo)
	{
	int len = msgInfo->fl;
	printf("send [%d]:", len);
	for (int i = 0; i < len; i++)
	{
	//
	printf("%02X ", msgInfo->msg[i]);
	}
	printf("\n");
	}
	//报文发送回调
	Reg_Svr_parseSendMsg(Svr_parseSendMsg_CallBack);

2.21 C_CI_NA_2、C_CI_NE_2、C_CI_NI_2、C_CI_NN_2 读累积量

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_GetElect104_108_112_116(const char* svrName,E_102_TYPE typeId);

功能	根据 typeId，获取各种累计量数据
参数	svrName：连接从站的名字，作为当前连接标识使用； typeId:类型标识(ENUM_102_TYPE_C_CI_NA_2、ENUM_102_TYPE_C_CI_NE_2、ENUM_102_TYPE_C_CI_NI_2、ENUM_102_TYPE_C_CI_NN_2)
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char *svrName = "test";//从站标识 程序自定义 跟当前连接进行绑定
	int ret = TS_IEC102_Send_GetElect104_108_112_116(svrName, ENUM_102_TYPE_C_CI_NA_2);

2.18 注册 带时标的单点信息 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type1_Resp(Svr_IEC102_Type1_Resp callback);
功能	带时标的单点信息 回调
参数	函数指针
返回值	空

示例	//<1>:=M_SP_TA_2 带时标的单点信息
	void Svr_IEC102_Type1_Resp_CallBack(const char *server_name, struct IEC102_Type1_Resp *resp)
	{
	printf("Svr_IEC102_Type1_Resp_CallBack:%d\n", resp->info_Num);
	for (int i = 0; i < resp->info_Num; i++)
	{
	printf("addr:%d\n", resp->info_addrList[i]->bySPA);
	printf("time:%s\n", resp->pTimeBList[i]->str_time);
	}
	//SetRecvSatus(1);
	}
	//注册 <1>:=M_SP_TA_2 带时标的单点信息
	Reg_Svr_IEC102_Type1_Resp(Svr_IEC102_Type1_Resp_CallBack);

2.22 注册 C_CI_NA_2、C_CI_NE_2、C_CI_NI_2、C_CI_NN_2 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type104_108_112_116_Resp(Svr_IEC102_Type104_108_112_116_Resp callback);
功能	C_CI_NA_2、C_CI_NE_2、C_CI_NI_2、C_CI_NN_2 信息 回调
参数	函数指针
返回值	空

示例	Svr_IEC102_Type104_108_112_116_Resp_CallBack(const char *server_name, struct IEC102_Type104_108_112_116_Resp *resp)
	{
	}
	Reg_Svr_IEC_102_Type104_108_112_116_Resp(Svr_IEC_102_Type104_108_112_116_Resp_CallBack);

2.23 C_CI_NC_2、C_CI_NG_2、C_CI_NL_2、C_CI_NP_2 读累积量

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_GetElect106_110_114_118(const char* svrName,E_102_TYPE typeId, IEC102_BYTE recordAddr, IEC_BYTE minute);
功能	根据 typeId，获取各种累计量数据
参数	svrName：连接从站的名字，作为当前连接标识使用； typeId:类型标识(ENUM_102_TYPE_C_CI_NC_2、ENUM_102_TYPE_C_CI_NG_2、ENUM_102_TYPE_C_CI_NL_2、ENUM_102_TYPE_C_CI_NP_2) recordAddr:记录地址 minute:分钟，获取 minute 前的时间
返回值	发送成功字节数，发送成功大于 0，否则失败
示例	const char *svrName = "test";//从站标识 程序自定义 跟当前连接进行绑定

	int minute = 5; //5分钟
	int recordAddr = 1; //记录地址
	int ret = TS_IEC102_Send_GetElect106_110_114_118(svrName, ENUM_102_TYPE_C_CI_NC_2,recordAddr, minute);

2.24 注册 C_CI_NC_2 、 C_CI_NG_2 、 C_CI_NL_2 、 C_CI_NP_2 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type106_110_114_118_Resp(Svr_IEC102_Type106_110_114_118_Resp callback);
功能	**C_CI_NC_2、C_CI_NG_2、C_CI_NL_2、C_CI_NP_2** 信息 回调
参数	函数指针
返回值	空

示例	Svr_IEC102_Type106_110_114_118_Resp_CallBack(const char *server_name, struct IEC_102_Type106_110_114_118Resp *resp)
	{
	}
	Reg_Svr_IEC102_Type106_110_114_118_Resp(Svr_IEC102_Type106_110_114_118_Resp_CallBack);

2.25 C_CI_ND_2、C_CI_NH_2、C_CI_NM_2、C_CI_NQ_2 读累积量

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_GetElect107_111_115_119(const char* svrName,E_102_TYPE typeId,IEC102_BYTE recordAddr, IEC102_BYTE startAddr,IEC102_BYTE endAddr, IEC102_BYTE minute);
功能	根据 typeId，获取各种累计量数据
参数	svrName：连接从站的名字，作为当前连接标识使用； typeId:类型标识(ENUM_102_TYPE_C_CI_ND_2、ENUM_102_TYPE_C_CI_NH_2、ENUM_102_TYPE_C_CI_NM_2、ENUM_102_TYPE_C_CI_NQ_2) recordAddr:记录地址 startAddr:开始地址

	endAddr:结束地址 minute:距离当前时间过去分钟数
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char *svrName = "test";//从站标识 程序自定义 跟当前连接进行绑定
	BYTE recordAddr = 1; //记录地址
	IEC102_BYTE startAddr = 1; //开始地址
	IEC102_BYTE endAddr = 2;//结束地址
	IEC102_BYTE minute = 5; //时间指定为5分钟前
	int ret = TS_IEC102_Send_GetElect107_111_115_119(svrName, ENUM_102_TYPE_C_CI_ND_2, recordAddr,startAddr,endAddr,minute);

2.26 注册 C_CI_ND_2、C_CI_NH_2、C_CI_NM_2、C_CI_NQ_2 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type107_111_115_119_Resp(Svr_IEC102_Type107_111_115_119_Resp callback);
功能	C_CI_ND_2、C_CI_NH_2、C_CI_NM_2、C_CI_NQ_2 信息 回调
参数	函数指针
返回值	空

示例	Svr_IEC102_Type107_111_115_119_Resp_CallBack(const char *server_name, struct IEC102_Type107_111_115_119_Resp *resp)
	{
	}
	Reg_Svr_IEC102_Type107_111_115_119_Resp(Svr_IEC102_Type107_111_115_119_Resp_CallBack);

2.27 C_CI_NB_2、C_CI_NF_2、C_CI_NK_2、C_CI_NO_2 读累积量

函数	int TS_DLL_EXPORT TS_IEC102_Send_GetElect105_109_113_117(const char* svrName,E_102_TYPE typeId,IEC102_BYTE recordAddr, IEC102_BYTE

声明	startAddr,IEC102_BYTE endAddr);
功能	根据 typeId，获取各种累计量数据
参数	svrName：连接从站的名字，作为当前连接标识使用； typeId:类型标识(ENUM_102_TYPE_C_CI_NB_2、ENUM_102_TYPE_C_CI_NF_2、ENUM_102_TYPE_C_CI_NK_2、ENUM_102_TYPE_C_CI_NO_2)
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char *svrName = "test";//从站标识 程序自定义 跟当前连接进行绑定
	IEC102_BYTE recordAddr = 1;
	IEC102_BYTE startAddr = 1;
	IEC102_BYTE endAddr = 2;
	int ret = TS_IEC102_Send_GetElect105_109_113_117(svrName, ENUM_102_TYPE_C_CI_NB_2,recordAddr,startAddr, endAddr);

2.28 注册 C_CI_NB_2、C_CI_NF_2、C_CI_NK_2、C_CI_NO_2 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type105_109_113_117_Resp(Svr_IEC102_Type104_108_112_116_Resp callback);
功能	**C_CI_NB_2、C_CI_NF_2、C_CI_NK_2、C_CI_NO_2** 信息 回调
参数	函数指针
返回值	空

示例	Svr_IEC102_Type105_109_113_117_Resp_CallBack(const char *server_name, struct IEC102_Type105_109_113_117_Resp *resp)
	{
	}
	Reg_Svr_IEC102_Type105_109_113_117_Resp(Svr_IEC102_Type105_109_113_117_Resp_CallBack);

2.29 C_CI_NR_2、C_CI_NS_2、C_CI_NT_2、C_CI_NU_2 读累积量

函数声明	int TS_DLL_EXPORT TS_IEC102_Send_GetElect120_121_122_123(const char* svrName,E_102_TYPE typeId, IEC102_BYTE recordAddr, IEC102_BYTE startAddr,IEC102_BYTE endAddr,IEC102_BYTE minute);
功能	根据 typeId，获取各种累计量数据
参数	svrName：连接从站的名字，作为当前连接标识使用； typeId:类型标识(ENUM_102_TYPE_C_CI_NR_2、ENUM_102_TYPE_C_CI_NS_2、ENUM_102_TYPE_C_CI_NT_2、ENUM_102_TYPE_C_CI_NU_2) recordAddr:记录地址 startAddr:开始地址 endAddr:结束地址 minute:当前时间前多少分钟
返回值	发送成功字节数，发送成功大于 0，否则失败

示例	const char *svrName = "test";//从站标识 程序自定义 跟当前连接进行绑定
	IEC102_BYTE recordAddr = 1; //记录地址
	IEC102_BYTE startAddr = 1; //开始地址
	IEC102_BYTE endAddr = 2; //结束地址
	IEC102_BYTE minute = 5; //当前时间多少分钟前
	int ret = TS_IEC102_Send_GetElect120_121_122_123(svrName, ENUM_102_TYPE_C_CI_NA_2, recordAddr,startAddr,endAddr, minute);

2.30 注册 C_CI_NR_2 、 C_CI_NS_2 、 C_CI_NT_2 、 C_CI_NU_2 回调

函数声明	void TS_DLL_EXPORT Reg_Svr_IEC102_Type120_121_122_123_Resp(Svr_IEC_102_Type120_121_122_123_Resp callback);
功能	C_CI_NR_2、C_CI_NS_2、C_CI_NT_2、C_CI_NU_2 信息 回调
参数	函数指针
返	空

回值	

示例	Svr_IEC102_Type120_121_122_123_Resp_CallBack(const char *server_name, struct IEC102_Type120_121_122_123_Resp *resp)
	{
	}
	Reg_Svr_IEC102_Type120_121_122_123_Resp(Svr_IEC_102_Type120_121_122_123_Resp_CallBack);