CMS
客户端开发包接口使用手册

版本：V2.2

深圳天勺电力软件有限公司

中国. 深圳

1.开发包简介

国产自主可控新一代变电站通信协议（简称：CMS）是最新的 IEC61850 通信

协议，替代了原来采用 MMS 通信的标准。CMS 协议涉及的命令数量比 MMS 多很多，采用 PER 编码规则进行报文的传输。

为了方便国内客户端研发人员快速的实现基于 CMS 标准的通信协议，我司将 CMS 协议客户端的开发包采用 API 接口的方式进行了封装。极大的节约了客户从零研发的周期和成本。

本开发包的 API 接口，覆盖了 CMS 标准所有的命令，严格按照最新 CMS 标准进行封装。本开发包现场应用成熟稳定，方便易用。

提示：具体的参数说明，可参考 CMS 通信报文规范说明：《自主可控新一代变电站二次系统技术规范　通用类系列规范 4　DLT 860 通信报文（V1.1）》

2. 接口描述

2.1 建立 TCP 连接

函数声明	bool　connectServer(const char *svrName, const char *IPAddr, int ApduSize,int AsduSize,const char * local_ip, unsigned int ver,unsigned int port = 8102,unsigned int timeOut = 10);
功能	与服务器建立连接
参数	const char *svrName：连接服务器的名字，作为当前连接标识使用； const char *IPAddr:服务器 IP int ApduSize:APDU （取值范围 7~65531） int AsduSize:ASDU （取值范围 7~262144） const char * local_ip:本地 Ip unsigned int ver:协议版本号（一般给值 513） unsigned int port:服务器端口号（默认端口 8102） unsigned int timeOut:超时时间
返回值	发送成功返回 true，否则为 false;

示例	bool ret = connectServer ("test","192.168.3.147",65530, 13002,"192.168.3.147",513,8102);

Tip：当连接成功后，后续命令才能发送成功

2.2 建立 TCP 连接（密文传输模式）

函数声明	bool connectServerSSL(const char *svrName, const char *IPAddr, int ApduSize,int AsduSize,const char * local_ip, unsigned int ver,const char * derpath, const char * cerpath,const char * p12path, const char * pwd,unsigned int port = 9102, unsigned int timeOut = 10);
功能	与服务器建立连接
参数	const char *svrName: 连接服务器的名字，作为当前连接标识使用； const char *IPAddr: 服务器IP int ApduSize: APDU （取值范围7~65531） int AsduSize, （取值范围7~262144） const char * local_ip: 本地Ip unsigned int ver: 协议版本号（一般给值513） const char * derpath:der证书（xxder.cer） const char * cerpath:cer证书（xx.cer） const char * p12path:p12证书（xx.p12） const char * pwd:证书密码 unsigned int port: 服务器端口号（默认端口9102） unsigned int timeOut: 超时时间
返回值	发送成功返回 true，否则为 false;
示例	bool ret = connectServerSSL("test","192.168.3.130",65000, 130032,"192.168.3.130", 513, "D:\\simu\\ICD\\doc\\ca-der.cer","D:\\simu\\ICD\\doc\\ca.cer", "D:\\simu\\ICD\\doc\\platform-new.p12","test",9102,20);

2.3 AssociateNegotiate 服务函数

函数声明	int sendAssociateNegotiate(const char * svrName, unsigned int apduSzie, unsigned int asduSize);
功能	AssociateNegotiate 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 unsigned int apduSzie: APDU（取值范围7~65531） unsigned int asduSize: ASDU（取值范围 7~262144）

返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	int ret = sendAssociateNegotiate ("test",65530, 13002);

2.4 注册 AssociateNegotiate 正响应回调函数

函数声明	void Reg_Svr_AssociateNegotiate_P(Svr_AssociateNegotiate_P callBack);
功能	获取 AssociateNegotiate 响应信息
参数	Svr_AssociateNegotiate_P callback:回调函数
返回值	void
示例	void Associate_P_CallBack(const char *server_name,SZTS_AssociateNegotiateResp resp){} Reg_Svr_AssociateNegotiate_P(Associate_P_CallBack);

2.5 注册 AssociateNegotiate 负响应回调函数

函数声明	void Reg_Svr_AssociateNegotiate_N(Svr_AssociateNegotiate_N callBack);
功能	获取 AssociateNegotiate 响应信息
参数	Svr_AssociateNegotiate_N callBac:回调函数
返回值	发送成功返回 true，否则为 false;
示例	void Associate_N_CallBack (const char *server_name,int reqId,ServiceError err){} Reg_Svr_AssociateNegotiate_N (Associate_N_CallBack);

2.6 Associate 服务函数

函数声明	int sendAssociate(const char * svrName, const char * ServerAccessPointReference, AuthenticationParameter AuthenticationParameter);

功能	Associate 请求
参数	const char *svrName: 连接服务器的名字，作为当前连接标识使用； const char *ServerAccessPointReference: 访问点的引用（一般默认控制即可） AuthenticationParameter AuthenticationParameter:安全认证参数（详情看协议8.2.1）
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	char* str1 = GetSignatureCertificate("D:\\simu\\ICD\\doc\\platform-new.p12","test");
	char* str2 = GetSignedValue("D:\\simu\\ICD\\doc\\platform-new.p12","test","2023-02-24 12:46:55.000");
	AuthenticationParameter auth;
	auth.setSignatureCertificate(str1);
	auth.setSignedValue(str2);
	sendAssociateNegotiate("test",65000, 130032);
	Sleep(1000);
	sendAssociate("test", "", auth);
	if(str1){
	delete str1;
	str1 = NULL;
	}
	if(str2){
	delete str2;
	str2 = NULL;
	}

2.7 注册 Associate 正响应回调函数

函数声明	void Reg_Svr_Associate_P(Svr_Associate_P callBack);
功能	Associate 正响应回调注册
参数	Svr_Associate_P callback:回调函数
返回值	无
示例	void Associate_P_CallBack(const char *server_name,SZTS_AssociateResp resp){} Reg_Svr_Associate_P(Associate_P_CallBack);

2.8 注册 Associate 负响应回调函数

函数声明	void Reg_Svr_Associate_N(Svr_Associate_N callBack);
功能	Associate 负响应回调注册
参数	Svr_Associate_N callBack callback:回调函数
返回值	无
示例	//回调声明实现 void Associate_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_Associate_N(Associate_N_CallBack);

2.9 Abort 服务函数

函数声明	int sendAbort(const char * svrName, const char * associationId, AbortReason reason);
功能	Abort 请求
参数	const char * svrName:连接服务器的名字，作为当前连接标识使用 const char * associationId:关联的 ID（Associate 正响应中返回） AbortReason reason:原因（详情看协议 8.2.3 异常中止（Abort））
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	//关联响应中获取到 这里直接绑定
	const char * associationId = "0000000000000002";
	sendAbort(serName, associationId, other);

2.10 Release 服务函数

函数声明	int sendReleaseReq(const char * svrName, const char * associationId);
功能	Release 请求
参数	const char * svrName:连接服务器的名字，作为当前连接标识使用 const char * associationId:关联的 ID（Associate 正响应中返回）
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	//关联响应中获取到 这里直接绑定
	const char * associationId = "0000000000000002";
	sendReleaseReq(serName, associationId);

2.11 注册 Release 正响应回调函数

函数声明	void Reg_Svr_Release_P(Svr_Release_P callBack);
功能	Release 正响应回调注册
参数	Svr_Release_P callBack:回调函数
返回值	无
示例	//声明定义 void Release_P_CallBack(const char *server_name,SZTS_ReleaseResp resp){} Reg_Svr_Release_P(Release_P_CallBack);

Tips:响应参数看协议

2.12 注册 Release 负响应回调函数

函数声明	void Reg_Svr_Release_N(Svr_Release_N callBack);
功能	Release 负响应回调注册
参数	Svr_Release_N callBack:回调函数
返回值	无
示例	void Release_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_Release_N(Release_N_CallBack);

Tips:响应参数看协议

2.13 GetServerDirectory 服务函数

函数声明	int SendGetServerDirectory(string svrName,int objClass,string refAft);
功能	GetServerDirectory 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 int objClass:0-保留 1-逻辑设备 2-文件 见协议 表 8- 1ObjectClass string refAft: 参数为可选项 当需要多次获取时，取响应最后一个值
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetServerDirectory(svrName,1, "");

2.14 注册 GetServerDirectory 正响应回调函数

函数声明	Void Reg_Svr_GetServerDirectory_P(GetServerDirectory_P callback);
功能	获取 GetServerDirectory 正响应
参数	GetServerDirectory_P callback:回调函数
返回值	空
示例	/声明定义 void GetServerDirectory_P_CallBack(const char *server_name,SZTS_GetServerDirectoryResp resp){} Reg_Svr_GetServerDirectory_P(GetServerDirectory_P_CallBack);

Tips:响应参数看协议

2.15 注册 GetServerDirectory 负响应回调函数

函数声明	**Void Reg_Svr_GetServerDirectory_N(GetServerDirectory_N callback);**
功能	获取 GetServerDirectory 负响应
参数	**GetServerDirectory_N callback**:回调函数
返回值	空
示例	void GetServerDirectory_N_CallBack(const char *server_name,int reqId,ServiceError err){} **Reg_Svr_GetServerDirectory_N(**GetServerDirectory_N_CallBack);

Tips:响应参数看协议

2.16 GetLogicDeviceDirectory 服务函数

函数声明	int SendGetLogicalDeviceDirectory(string svrName,string ldName ,string continueAfter);
功能	GetLogicDeviceDirectory 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string ldName:逻辑设备 string continueAfter: 参数为可选项 当需要多次获取时，取响应最后一个值
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetServerDirectory(svrName,"TempLD0","");

2.17 注册 GetLogicDeviceDirectory 正响应回调函数

函数声明	void Reg_Svr_GetLogicalDeviceDirectory_P(GetLogicalDeviceDirectory_P callback);
功能	获取 GetLogicDeviceDirectory 正响应
参数	GetLogicalDeviceDirectory_P callback：回调函数
返回值	空
示例	void GetLogicalDeviceDirectory_P_CallBack (const char *server_name, SZTS_GetLogicalDeviceDirectoryResp resp){} Reg_Svr_GetLogicalDeviceDirectory_P(GetLogicalDeviceDirectory_P_CallBack);

2.18 注册 GetLogicDeviceDirectory 负响应回调函数

函数声明	void Reg_Svr_GetLogicalDeviceDirectory_N(GetLogicalDeviceDirectory_N callback);
功能	获取 GetLogicDeviceDirectory 负响应
参数	GetLogicalDeviceDirectory_N callback:回调函数
返回值	空
示例	void GetLogicalDeviceDirectory_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetLogicalDeviceDirectory_N(GetLogicalDeviceDirectory_N_CallBack);

2.19 GetLogicNodeDirectory 服务函数

函数声明	int SendGetLogicalNodeDirectory(string svrName,string

	ldName,string lnReference, int acsiClass,string continueAfter);
功能	GetLogicNodeDirectory 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string lnReference:ln 索引，如 LD/LN int acsiClass:对应协议表 8-2ACSIClass 值 string continueAfter: 参数为可选项 当需要多次获取时，取响应最后一个值
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetLogicalNodeDirectory (serName, "LD/LN",2, "");

2.20 注册 GetLogicNodeDirectory 正响应函数

函数声明	void Reg_Svr_GetLogicalNodeDirectory_P(GetLogicalNodeDirectory_P callback);
功能	获取 GetLogicNodeDirectory 正响应
参数	GetLogicalNodeDirectory_P callback:回调函数
返回值	空
示例	Void GetLogicalNodeDirectory_P_CallBack(const char *server_name,SZTS_GetLogicalNodeDirectoryResp resp){} Reg_Svr_GetLogicalNodeDirectory_P (GetLogicalNodeDirectory_P_CallBack);

2.21 注册 GetLogicNodeDirectory 负响应函数

函数声明	void Reg_Svr_GetLogicalNodeDirectory_N(GetLogicalNodeDirectory_N callback);
功能	获取 GetLogicNodeDirectory 负响应
参数	GetLogicalNodeDirectory_N callback:回调函数

返回值	空
示例	void GetLogicalNodeDirectory_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetLogicalNodeDirectory_N(GetLogicalNodeDirectory_N_CallBack);

2.22 GetAllDataValues 服务函数

函数声明	int SendGetAllDataValues(string svrName,string ldName,string lnReference, string fc, string continueAfter);
功能	GetAllDataValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string ldName: 指定逻辑设备 (ldName 或者 lnReference 二选一) string lnReference: 指定逻辑节点下 string fc: 功能约束值 (空值不指定 FC) string continueAfter: 参数为可选项 当需要多次获取时，取响应最后一个值
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetAllDataValues(serName, "", "LD/LN", "", "");

2.23 注册 GetAllDataValues 正响应回调函数

函数声明	void Reg_Svr_GetAllDataValues_P(GetAllDataValues_P callback);
功能	获取GetAllDataValues正响应回调
参数	GetAllDataValues_P callback:回调函数
返回值	空
示例	void GetAllDataValues_P_CallBack(const char *server_name,SZTS_GetAllDataValuesResp resp){} Reg_Svr_GetAllDataValues_P(GetAllDataValues_P_CallBack);

2.24 注册 GetAllDataValues 负响应回调函数

函数声明	void Reg_Svr_GetAllDataValues_N(GetAllDataValues_N callback);

功能	获取 GetAllDataValues 负响应回调
参数	GetAllDataValues_N callback:回调函数
返回值	空
示例	void GetAllDataValues_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetAllDataValues_N(GetAllDataValues_N_CallBack);

2.25 GetAllDataDefinition 服务函数

函数声明	int SendGetAllDataDefinition(string svrName,string ldName,string lnReference, string fc, string continueAfter);
功能	GetAllDataDefinition 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string ldName: 指定逻辑设备 (ldName 或者 lnReference 二选一) string lnReference: 指定逻辑节点下 string fc: 功能约束值 (空值不指定 FC) string continueAfter: 参数为可选项 当需要多次获取时，取响应最后一个值
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetAllDataDefinition(serName,"E1Q1SB22PROT","","","");

2.26 注册 GetAllDataDefinition 正响应回调函数

函数声明	void Reg_Svr_GetAllDataDefinition_P(GetAllDataDefinition_P callback);
功能	获取 GetAllDataDefinition 正响应
参数	GetAllDataDefinition_P callback:回调函数
返回值	空
示例	void GetAllDataDefinition_P(const char *server_name,SZTS_GetAllDataDefinitionResp resp){} Reg_Svr_GetAllDataDefinition_P(GetAllDataDefinition_P);

2.27 注册 GetAllDataDefinition 负响应回调函数

函数声	void Reg_Svr_GetAllDataDefinition_N(GetAllDataDefinition_N callback);

明	
功能	获取 GetAllDataDefinition_N callback 负响应
参数	GetAllDataDefinition_N callback:回调函数
返回值	空
示例	void GetAllDataDefinition_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetAllDataDefinition_N(GetAllDataDefinition_N_CallBack);

2.28 GetAllCBValues 服务函数

函数声明	int SendGetAllCBValues(string svrName,string ldName,string lnReference,ACSIClass acsiClass,string continueAfter);
功能	GetAllCBValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string ldName: 指定逻辑设备 (ldName 或者 lnReference 二选一) string lnReference: 指定逻辑节点下 ACSIClass acsiClass: 控制块类型 详情看协议表 8-2ACSIClass 值 string continueAfter: 参数为可选项 当需要多次获取时，取响应最后一个值
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetAllCBValues (serName, "E1Q1SB22PROT", "", brcb, "");

2.29 注册 GetAllCBValues 正响应回调函数

函数声明	void Reg_Svr_GetAllCBValues_P(GetAllCBValues_P callback);
功能	获取 GetAllCBValues 正响应
参数	GetAllCBValues_P callback:回到函数
返回值	空
示例	void GetAllCBValues_P_CallBack(const char *server_name,SZTS_GetAllCBValuesResp resp){} Reg_Svr_GetAllCBValues_P(GetAllCBValues_P_CallBack);

2.30 注册 GetAllCBValues 负响应回调函数

函数声明	void Reg_Svr_GetAllCBValues_N(GetAllCBValues_N callback);
功能	获取 GetAllCBValues 负响应
参数	GetAllCBValues_N callback:回到函数
返回值	空
示例	void GetAllCBValues_N_CallBack(const char *server_name,int reqId,SZTS_GetAllCBValuesResp resp){} Reg_Svr_GetAllCBValues_N (GetAllCBValues_N_CallBack);

2.31 GetDataValues 服务函数

函数声明	int sendGetDataValueReq(string svrName, deque<GdData> *datalist);
功能	GetDataValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<GdData> *datalist:请求数据列表 *getData.reference-索引* getData.fc-是否指定FC
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	*deque*<GdData> getDatalist; GdData *getData*; *getData.reference* = "TEMPLATELD0/LLN0.LEDRs.subEna"; getDatalist.*push_back*(*getData*); sendGetDataValueReq (svrName, &getDatalist);

2.32 注册 GetDataValues 正响应回调函数

函数声明	void Reg_Svr_GetDataValue_P(GetDataValue_P callBack);
功能	获取 GetDataValues 正响应
参数	GetDataValue_P callback:回调函数
返回值	无
示例	void GetDataValue_P_CallBack(const char *server_name,SZTS_GetDataValuesResp resp){} Reg_Svr_GetDataValue_P(GetDataValue_P_CallBack);

2.33 注册 GetDataValues 负响应回调函数

函数声明	void Reg_Svr_GetDataValue_N(GetDataValue_N callBack);
功能	获取 GetDataValues 负响应
参数	GetDataValue_N callBack:回调函数
返回值	无
示例	void GetDataValue_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetDataValue_N(GetDataValue_N_CallBack);

2.34 SetDataValues 服务函数

函数声明	int sendSetDataValueReq(string svrName, deque<Data> *datalist);
功能	SetDataValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<Data> *datalist:设置参数列表
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	*deque*<Data> datalist;
	Data setData;
	//引用
	setData.reference = "A_V0003BMONT/SMEM1.MemRat.subMag.f";
	//setData.dataType-数据类型
	setData.dataType = szts_FLOAT32;
	//数据类型对应的值
	//数据赋值时，需要注意类型 如:float32为string
	setData.float32 = "21.1";
	datalist.*push_back*(setData);
	sendSetDataValueReq (svrName,&datalist);

2.35 注册 SetDataValues 正响应回调函数

函数声明	void Reg_Svr_SetDataValue_P(SetDataValue_P callBack);
功能	获取 SetDataValues 正响应回调
参数	SetDataValue_P callBack：回调函数
返回值	空
示例	void SetDataValue_P_CallBack(const char *server_name,SZTS_SetDataValuesResp){} Reg_Svr_SetDataValue_P(SetDataValue_P_CallBack);

2.36 注册 SetDataValues 负响应回调函数

函数声明	Void Reg_Svr_SetDataValue_N(SetDataValue_N callBack);
功能	获取 SetDataValues 负响应回调
参数	SetDataValue_N callback:回调函数
返回值	空
示例	void SetDataValue_N_CallBack(const char *server_name,int reqId,SZTS_SetDataValuesErr err){} Reg_Svr_SetDataValue_N(SetDataValue_N_CallBack);

2.37 GetDataDirectory 服务函数

函数声明	int SendGetDataDirectory(string svrName,string dataReference, string referenceAfter);
功能	GetDataDirectory 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string dataReference: 数据对象 string referenceAfter: 参数为可选项 当需要多次获取时，取响应最后一个值
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetDataDirectory (serName, "TEMPLATEPROT/LLN0.Mod",");

2.38 注册 GetDataDirectory 正响应回调函数

函数声明	void Reg_Svr_GetDataDirectory_P(GetDataDirectory_P callback);
功能	获取 GetDataDirectory 正响应
参数	GetDataDirectory_P callback:回调函数
返回值	无
示例	void GetDataDirectory_P_Callback(const char *server_name,SZTS_GetDataDirectoryResp resp){} Reg_Svr_GetDataDirectory_P(GetDataDirectory_P_Callback);

2.39 注册 GetDataDirectory 负响应回调函数

函数声明	void Reg_Svr_GetDataDirectory_N(GetDataDirectory_N callback);
功能	获取 GetDataDirectory 负响应
参数	GetDataDirectory_N callback:回调函数
返回值	空
示例	void GetDataDirectory_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetDataDirectory_N(GetDataDirectory_N_CallBack);

2.40 GetDataDefinition 服务函数

函数声明	int sendGetDataDefinitionReq(string svrName, deque<GdData> *deflist);
功能	GetDataDefinition 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<GdData> *deflist: 数据对象或数据属性列表 GdData.reference - 索引 GdData.fc - 控制值（可选项）

返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; *deque*<GdData> deflist;; GdData dgdata; dgdata.reference = "TEMPLATEPROT/LLN0.Mod"; deflist.*push_back*(dgdata); sendGetDataDefinitionReq (serName, deflist);

2.41 注册 GetDataDefinition 正响应回调函数

函数声明	void Reg_Svr_GetDataDefinition_P(GetDataDefinition_P callBack);
功能	获取 **GetDataDefinition** 正响应
参数	GetDataDefinition_P callback:回调函数
返回值	无
示例	void GetDataDefinition_P_CallBack(const char *server_name,SZTS_GetDataDefinitionResp resp){} Reg_Svr_GetDataDefinition_P(GetDataDefinition_P_CallBack);

2.42 注册 GetDataDefinition 负响应回调函数

函数声明	void Reg_Svr_GetDataDefinition_N(GetDataDefinition_N callBack);
功能	获取 **GetDataDefinition** 负响应
参数	GetDataDefinition_N callBack:回调函数
返回值	无
示	void GetDataDefinition_N_CallBack(const char *server_name,int

例	reqId,ServiceError err){} Reg_Svr_GetDataDefinition_N (GetDataDefinition_N_CallBack);

2.43 CreateDataSet 服务函数

函数声明	int sendCreateDataSet(string svrName, string datasetReference, string referenceAfter, deque<MemberData> *memberDataList);
功能	CreateDataSet 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string datasetReference:数据集索引（以@开头为非永久性数据集） string referenceAfter: 未指定 referenceAfter,表示创建一个新的数据集；未指定 referenceAfter ，referenceAfter 应为现有数据集的最后一个成员 deque<MemberData> *memberDataList: 数据集成员 MemberData.objectReference – 成员索引 MemberData.fc –控制值
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque<MemberData> memberDataList;
	MemberData memData;
	memData.objectReference = "TEMPLATEPROT/LLN0.Mod";
	string dataSetName = "TEMPLATEPROT/LLN0.dataSetName1";
	memData.fc = "ST";
	memberDataList.push_back(memData);
	sendCreateDataSet(svrName,dataSetName,"",&memberDataList);

2.44 注册 CreateDataSet 正响应回调函数

函数声明	void Reg_Svr_CreateDataSet_P(CreateDataSet_P callBack);
功能	获取 CreateDataSet 正响应
参数	CreateDataSet_P callBack:回调函数
返回值	无

示例	void CreateDataSet_P_CallBack(const char *server_name,int reqId){} Reg_Svr_CreateDataSet_P (CreateDataSet_P_CallBack);

2.45 注册 CreateDataSet 负响应回调函数

函数声明	void Reg_Svr_CreateDataSet_N(CreateDataSet_N callBack);
功能	获取 CreateDataSet 负响应
参数	CreateDataSet_P callBack:回调函数
返回值	无
示例	void CreateDataSet_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_CreateDataSet_N(CreateDataSet_N_CallBack);

2.46 DeleteDataSet 服务函数

函数声明	int sendDelectDataSet(string svrName, string ref);
功能	DeleteDataSet 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string datasetReference:数据集索引（非永久性数据集以@开头）
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; string dataSetName = "TEMPLATEPROT/LLNO.dataSetName1"; sendDelectDataSet(svrName,dataSetName);

2.47 注册 DelectDataSet 正响应回调函数

函数	void Reg_Svr_DeleteDataSet_P(DeleteDataSet_P callBack);

声明	
功能	获取 DelectDataSet 正响应
参数	DeleteDataSet_P callBack:回调函数
返回值	无
示例	void DeleteDataSet_P_CallBack(const char *server_name,int reqId){} Reg_Svr_DeleteDataSet_P (DeleteDataSet_P_CallBack);

2.48 注册 DelectDataSet 负响应回调函数

函数声明	void Reg_Svr_DeleteDataSet_N(DeleteDataSet_N callBack);
功能	获取 DelectDataSet 负响应
参数	DeleteDataSet_N callBack:回调函数
返回值	无
示例	void DeleteDataSet_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_DeleteDataSet_N(DeleteDataSet_N_CallBack);

2.49 GetDataSetDirectory 服务函数

函数声明	int SendGetDataSetDirectory(string svrName,string dataReference, string referenceAfter);
功能	GetDataSetDirectory 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string dataReference:数据集索引（非永久性数据集以@开头） string referenceAfter:指定成员获取位置
返回	发送成功返回请求序号（ReqID），否则为-1;

值	
示例	const char *serName = "test"; SendGetDataSetDirectory(serName,"TEMPLATELDO/LLNO.dsWarning","");

2.50 注册 GetDataSetDirectory 正响应回调函数

函数声明	void Reg_Svr_GetDataSetDirectory_P(GetDataSetDirectory_P callback);
功能	获取 GetDataSetDirectory 正响应
参数	GetDataSetDirectory_P callback:回调函数
返回值	空
示例	void GetDataSetDirectory_P_CallBack(const char *server_name,SZTS_GetDataSetDirectoryResp resp){} Reg_Svr_GetDataSetDirectory_P(GetDataSetDirectory_P_CallBack);

2.51 注册 GetDataSetDirectory 负响应回调函数

函数声明	void Reg_Svr_GetDataSetDirectory_N(GetDataSetDirectory_N callback);
功能	获取 GetDataSetDirectory 负响应
参数	GetDataSetDirectory_N callback:回调函数
返回值	空
示例	void GetDataSetDirectory_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetDataSetDirectory_N(GetDataSetDirectory_N_CallBack);

2.52 GetDataSetValues 服务函数

函数声明	int sendGetDataSetValue(string svrName, string reference);
功能	GetDataSetValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string reference:数据集索引（非永久性数据集以@开头）
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; sendGetDataSetValue (serName,"TEMPLATELD0/LLN0.dsWarning");

2.53 注册 GetDataSetValues 正响应回调函数

函数声明	void Reg_Svr_GetDataSetValue_P(GetDataSetValue_P callBack);
功能	获取 GetDataSetValues 正响应
参数	GetDataSetValue_P callBack:回调函数
返回值	空
示例	void GetDataSetValue_P_Call(const char *server_name,SZTS_GetDataSetValuesResp resp){} Reg_Svr_GetDataSetValue_P (GetDataSetValue_P_Call);

2.54 注册 GetDataSetDirectory 负响应回调函数

函数声明	void Reg_Svr_GetDataSetValue_N(GetDataSetValue_N callBack);

功能	获取 GetDataSetValues 负响应
参数	GetDataSetValue_N callBack:回调函数
返回值	空
示例	void GetDataSetValue_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetDataSetValue_N (GetDataSetValue_N_CallBack);

2.55 SetDataSetValues 服务函数

函数声明	int SendSetDataSetValue(string svrName,string datasetReference,deque<Data> value, string referenceAfter);
功能	SetDataSetValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string datasetReference:数据集索引（非永久性数据集以@开头） deque<Data> value:设置值的参数列表 string referenceAfter:从某个成员开始，一般为空即可
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque<Data> value;
	Data data;
	data.dataType = szts_Structure;
	Data data1;
	data1.dataType = szts_BOOLEAN;
	data1.boolean = true;
	data.structure.push_back(&data1);
	value.push_back(data);
	string datasetReference = "TEMPLATELDO/LLNO.dsWarning";
	SendSetDataSetValue (serName,datasetReference,value,"");

2.56 注册 SetDataSetValues 正响应回调函数

函数声明	void Reg_Svr_SetDataSetValue_P(SetDataSetValue_P callBack);
功能	获取 SetDataSetValuess 正响应
参数	SetDataSetValue_P callBack:回调函数
返回值	空
示例	void SetDataSetValue_P_Call(const char *server_name,SZTS_SetDataSetValuesResp resp){} Reg_Svr_SetDataSetValue_P (SetDataSetValue_P_Call);

2.57 注册 SetDataSetValues 负响应回调函数

函数声明	void Reg_Svr_SetDataSetValue_N(SetDataSetValue_N callBack);
功能	获取 SetDataSetValues 负响应
参数	SetDataSetValue_N callBack:回调函数
返回值	空
示例	void GetDataSetValue_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetDataSetDirectory_N(GetDataSetValue_N_CallBack);

2.58 Select 服务函数

函数声明	int sendSelectReq(string svrName, string reference);

功能	Select 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string reference:控制对象索引
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; string reference = "TEMPLATELDO/LLNO.LEDRs"; sendSelectReq (serName, reference);

2.59 注册 Select 正响应回调函数

函数声明	void Reg_Svr_Select_P(Svr_Select_P callback);
功能	获取 Select 正响应
参数	Svr_Select_P callback:回调函数
返回值	空
示例	void Svr_Select_P_CallBack(const char *server_name,SZTS_SelectResp resp){} Reg_Svr_Select_P (Svr_Select_P_CallBack);

2.60 注册 Select 负响应回调函数

函数声明	void Reg_Svr_Select_N(Svr_Select_N callback);
功能	获取 Select 负响应
参数	Svr_Select_N callback:回调函数
返回	空

值	
示例	void Svr_Select_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_Select_N (Svr_Select_N_CallBack);

2.61 SelectWithValue 服务函数

函数声明	int sendSelectWithValueReq(string svrName, string reference, Data ctlVal, string operTm, Origin origin, int ctlNum, string t, bool test, string check);
功能	SelectWithValue 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string reference:控制对象索引 Data ctlVal:控制值参数 string operTm:执行时间 （如 2022-02-01 12:00:00.000） Origin origin: 控制操作的发出者 详情看协议 7.5.2 int ctlNum:序列号 string t:时间戳 （如 2022-02-01 12:00:00.000） bool test:是否检修 string check:互锁检查 （如 01）
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	string ref = "TEMPLATELD0/LLN0.LEDRs";
	Origin origin;
	origin.orCat=*bayControl*;
	origin.orIdent = "sztsdl";
	string t = "2022-02-01 12:00:00.000";
	string operTm = "";
	Data data;
	data.dataType = szts_BOOLEAN;
	data.boolean = true;
	sendSelectWithValueReq(serName ,ref,data,operTm,origin,0,t,*false*,"01");

2.62 注册 SelectWithValue 正响应回调函数

函	void Reg_Svr_SelectWithValue_P(Svr_SelectWithValue_P callBack);

数声明	
功能	获取 SelectWithValue 正响应
参数	Svr_SelectWithValue_P callBack:回调函数
返回值	空
示例	void Svr_SelectWithValue_CallBack(const char *server_name,SZTS_SelectWithValueResp *resp){} Reg_Svr_SelectWithValue_P(Svr_SelectWithValue_CallBack);

2.63 注册 SelectWithValue 负响应回调函数

函数声明	void Reg_Svr_SelectWithValue_N(Svr_SelectWithValue_N callBack);
功能	获取 SelectWithValue 负响应
参数	Svr_SelectWithValue_N callBack:回调函数
返回值	空
示例	void Svr_SelectWithValue_N_CallBack(const char *server_name,int reqId,SZTS_SelectWithValueError *error){} Reg_Svr_SelectWithValue_N (Svr_SelectWithValue_N_CallBack);

2.64 Cancel 服务函数

函数声明	int sendCancelReq (string svrName, string reference, Data ctlVal, string operTm, Origin origin, int ctlNum, string t, bool test, string check);

功能	Cancel 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string reference:控制对象索引 Data ctlVal:控制值参数 string operTm:执行时间 （如 2022-02-01 12:00:00.000） Origin origin: 控制操作的发出者 详情看协议 7.5.2 int ctlNum:序列号 string t:时间戳 （如 2022-02-01 12:00:00.000） bool test:是否检修 string check:互锁检查 （如 01）
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	string ref = "TEMPLATELD0/LLN0.LEDRs";
	Origin origin;
	origin.orCat=*bayControl*;
	origin.orIdent = "sztsdl";
	string t = "2022-02-01 12:00:00.000";
	string operTm = "";
	Data data;
	data.dataType = szts_BOOLEAN;
	data.boolean = true;
	sendCancelReq (serName ,ref,data,operTm,origin,0,t,*false*,"01");

2.65 注册 Cancel 正响应回调函数

函数声明	void Reg_Svr_Cancel_P(Svr_Cancel_P callBack);
功能	获取 Cancel 正响应
参数	Svr_Cancel_P callBack:回调函数
返回值	空
示例	void Svr_Cancel_P_CallBack(const char *server_name,SZTS_CancelResp *resp){} Reg_Svr_Cancel_P (Svr_SelectWithValue_CallBack);

2.66 注册 Cancel 负响应回调函数

函数声明	void Reg_Svr_Cancel_N(Svr_Cancel_N callBack);
功能	获取 Cancel 负响应
参数	Svr_Cancel_N callBack:回调函数
返回值	空
示例	void Svr_Cancel_N_CallBack(const char *server_name,SZTS_CancelError *resp){} Reg_Svr_Cancel_N(Svr_Cancel_N_CallBack);

2.67 Operate 服务函数

函数声明	int sendOperateReq (string svrName, string reference, Data ctlVal, string operTm, Origin origin, int ctlNum, string t, bool test, string check);
功能	Operate 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string reference:控制对象索引 Data ctlVal:控制值参数 string operTm:执行时间 （如 2022-02-01 12:00:00.000） Origin origin: 控制操作的发出者 详情看协议 7.5.2 int ctlNum:序列号 string t:时间戳 （如 2022-02-01 12:00:00.000） bool test:是否检修 string check:互锁检查 （如 01） 详情看 7.5.3 控制操作的检测（ Check）
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; string ref = "TEMPLATELD0/LLN0.LEDRs";

	Origin origin;
	origin.orCat=*bayControl*;
	origin.orIdent = "sztsdl";
	string t = "2022-02-01 12:00:00.000";
	string operTm = "";
	Data data;
	data.dataType = szts_BOOLEAN;
	data.boolean = true;
	sendOperateReq (serName ,ref,data,operTm,origin,0,t,*false*,"01");

orCat 的值

值	解释
not-supported	orCat 不支持
bay-control	操作员通过间隔层客户发出的控制操作
station-control	操作员通过变电站层客户发出的控制操作
remote-control	远程操作员在变电站外（如网络控制中心）
automatic-bay	间隔层自动功能发出的控制操作
automatic-station	变电站层自动功能发出的控制操作
automatic-remote	变电站外的自动功能发出的控制操作
maintenance	维修/服务工具发出的控制操作
process	无控制行为而出现的状态变位（如断路器外部跳闸或内部故障） failure inside the breaker)

orIdent:原发者标识标识引起值的变化的原发者的地址；

2.68 注册 Operate 正响应回调函数

函数声明	void Reg_Svr_Operate_P(Svr_Operate_P callBack);
功能	获取 Operate 正响应
参数	Svr_Operate_P callBack:回调函数
返回值	空
示例	void Svr_Operate_P_CallBack(const char *server_name,SZTS_OperateResp *resp){} Reg_Svr_Operate_P (Svr_Operate_P_CallBack);

2.69 注册 Operate 负响应回调函数

函数声明	void Reg_Svr_Operate_N(Svr_Operate_N callBack);
功能	获取 Operate 负响应
参数	Svr_Operate_N callBack:回调函数
返回值	空
示例	void Svr_Operate_N_CallBack(const char *server_name,SZTS_OperateError *resp){} Reg_Svr_Operate_N(Svr_Operate_N_CallBack);

2.70 注册 CommandTermination 请求回调函数

函数声明	void Reg_Svr_CommandTermination(CommandTermination callBack);
功能	获取 CommandTermination 请求
参数	CommandTermination callBack:回调函数
返回值	空
示例	void CommandTermination_CallBack(const char *server_name,SZTS_CommandTerminationReq req){} Reg_Svr_CommandTermination (CommandTermination_CallBack);

2.71 TimeActivatedOperate 服务函数

函数声明	int sendTimeActivatedOperateReq(string svrName, string reference, Data ctlVal, string operTm, Origin origin, int ctlNum,string t, bool test, string check);

功能	TimeActivatedOperate 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string reference:控制对象索引 Data ctlVal:控制值参数 string operTm:执行时间 （如 2022-02-01 12:00:00.000） Origin origin: 控制操作的发出者 详情看协议 7.5.2 int ctlNum:序列号 string t:时间戳 （如 2022-02-01 12:00:00.000） bool test:是否检修 string check:互锁检查 （如 01）
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	string ref = "TEMPLATELD0/LLN0.LEDRs";
	Origin origin;
	origin.orCat=*bayControl*;
	origin.orIdent = "sztsdl";
	string t = "2022-02-01 12:00:00.000";
	string operTm = "2022-02-01 12:30:00.000";
	Data data;
	data.dataType = szts_BOOLEAN;
	data.boolean = true;
	sendTimeActivatedOperateReq (serName ,ref,data,operTm,origin,0,t,*false*,"01");

2.72 注册 TimeActivatedOperate 正响应回调函数

函数声明	void Reg_Svr_TimeActivatedOperate_P(Svr_TimeActivatedOperate_P callback);
功能	获取 TimeActivatedOperate 正响应
参数	Svr_TimeActivatedOperate_P callback:回调函数
返回值	空
示例	void Svr_TimeActivatedOperate_P_CallBack(const char *server_name,SZTS_TimeActivatedOperateResp resp){} Reg_Svr_TimeActivatedOperate_P

	(Svr_TimeActivatedOperate_P_CallBack);

2.73 注册 TimeActivatedOperate 负响应回调函数

函数声明	void Reg_Svr_TimeActivatedOperate_N(Svr_TimeActivatedOperate_N callback);
功能	获取 TimeActivatedOperate 负响应
参数	Svr_TimeActivatedOperate_N callback:回调函数
返回值	空
示例	void Svr_TimeActivatedOperate_N_CallBack(const char *server_name,SZTS_TimeActivatedOperateError err){} Reg_Svr_TimeActivatedOperate_N (Svr_TimeActivatedOperate_N_CallBack);

2.74 注册 TimeActivatedOperateTermination 正响应回调函数

函数声明	void Reg_Svr_TimeActivatedOperateTermination_P(Svr_TimeActivatedOperateTermination_P callback);
功能	获取 TimeActivatedOperateTermination 正响应
参数	Svr_TimeActivatedOperateTermination_P callback:回调函数
返回值	空
示例	void Svr_TimeActivatedOperateTermination_P_CallBack(const char *server_name,SZTS_TimeActivatedOperateTerminationReq resp){} Reg_Svr_TimeActivatedOperateTermination_P(Svr_TimeActivatedOperateTermination_P_CallBack);

2.75 注册 TimeActivatedOperateTermination 负响应回调函数

函数声明	void Reg_Svr_TimeActivatedOperateTermination_N(Svr_TimeActivatedOperateTermination_N callback);
功能	获取 TimeActivatedOperateTermination 负响应
参数	Svr_TimeActivatedOperateTermination_N callback:回调函数
返回值	空
示例	void Svr_TimeActivatedOperateTermination_N_CallBack(const char *server_name,SZTS_TimeActivatedOperateError err){} Reg_Svr_TimeActivatedOperateTermination_N (Svr_TimeActivatedOperateTermination_N_CallBack);

2.76 SelectActiveSG 服务函数

函数声明	int sendSelectActiveSGReq(string svrName, string sgcbReference, int settingGroupNumber);
功能	SelectActiveSG 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string sgcbReference:定值组索引 int settingGroupNumber:激活定值组
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; string ref = "TEMPLATELD0/LLN0.SGCB"; sendSelectActiveSGReq (serName ,ref,2);

2.77 注册 SelectActiveSG 正响应回调函数

函	void Reg_Svr_SelectActiveSG_P(SelectActiveSG_P callBack);

数声明	
功能	获取 SelectActiveSG 正响应
参数	SelectActiveSG_P callBack:回调函数
返回值	空
示例	void SelectActiveSG_P_CallBack(const char *server_name,int reqId){} Reg_Svr_SelectActiveSG_P (SelectActiveSG_P_CallBack);

2.78 注册 SelectActiveSG 负响应回调函数

函数声明	void Reg_Svr_SelectActiveSG_N(SelectActiveSG_N callBack);
功能	SelectActiveSG 负响应
参数	SelectActiveSG_N callBack:回调函数
返回值	空
示例	void SelectActiveSG_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_SelectActiveSG_N (SelectActiveSG_N_CallBack);

2.79 SelectActiveSG 服务函数

函数声明	int sendSelectActiveSGReq(string svrName, string sgcbReference, int settingGroupNumber);
功能	SelectActiveSG 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string sgcbReference:定值组索引

	int settingGroupNumber:激活定值组
返回值	发送成功返回 true，否则为 false;
示例	const char *serName = "test"; string ref = "TEMPLATELD0/LLN0. SGCB"; sendSelectActiveSGReq (serName ,ref,2);

2.80 注册 SelectActiveSG 正响应回调函数

函数声明	void Reg_Svr_SelectActiveSG_P(SelectActiveSG_P callBack);
功能	获取 SelectActiveSG 正响应
参数	SelectActiveSG_P callBack:回调函数
返回值	空
示例	void SelectActiveSG_P_CallBack(const char *server_name,int reqId){} Reg_Svr_SelectActiveSG_P (SelectActiveSG_P_CallBack);

2.81 注册 SelectActiveSG 负响应回调函数

函数声明	void Reg_Svr_SelectActiveSG_N(SelectActiveSG_N callBack);
功能	SelectActiveSG 负响应
参数	SelectActiveSG_N callBack:回调函数
返回值	空
示例	void SelectActiveSG_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_SelectActiveSG_N (SelectActiveSG_N_CallBack);

2.82 SelectEditSG 服务函数

函数声明	int sendSelectEditSGReq(string svrName, string sgcbReference, int settingGroupNumber);
功能	SelectEditSG 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string sgcbReference:定值组索引 int settingGroupNumber:编辑定值组号
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = ″test″; string ref = “TEMPLATELD0/LLN0. SGCB”; sendSelectEditSGReq (serName ,ref,2);

2.83 注册 SelectEditSG 正响应回调函数

函数声明	void Reg_Svr_SelectEditSG_P(SelectEditSG_P callBack);
功能	获取 SelectEditSG 正响应
参数	SelectEditSG_P callBack:回调函数
返回值	空
示例	void SelectEditSG_P_CallBack(const char *server_name,int reqId){} Reg_Svr_SelectEditSG_P (SelectEditSG_P_CallBack);

2.84 注册 SelectEditSG 负响应回调函数

函数	void Reg_Svr_SelectEditSG_N(SelectEditSG_N callBack);

声明	
功能	SelectEditSG 负响应
参数	SelectEditSG_N callBack:回调函数
返回值	空
示例	void SelectEditSG_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_SelectEditSG_N (SelectEditSG_N_CallBack);

2.85 SetEditSGValue 服务函数

函数声明	int sendSetEditSGValueReq(string svrName, deque<Data> dataList);
功能	SetEditSGValue 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<Data> dataList:设置定值数据列表
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque<Data> value;
	Data data;
	data.dataType = szts_Structure;
	Data data1;
	data1.reference = "TEMPLATELD0/LLN0. Mod.stVal";
	data1.fc = "SE";
	data1.dataType = szts_BOOLEAN;
	data1.boolean = true;
	data.structure.push_back(&data1);
	value.push_back(data);
	sendSetEditSGValueReq (serName, value);

2.86 注册 SetEditSGValue 正响应回调函数

函数声明	void Reg_Svr_SetEditSGValue_P(SetEditSGValue_P callBack);
功能	获取 SetEditSGValue 正响应
参数	SetEditSGValue_P callBack:回调函数
返回值	空
示例	void SetEditSGValue_P_CallBack(const char *server_name,SZTS_SetEditSGValueResp resp){} Reg_Svr_SetEditSGValue_P(SetEditSGValue_P_CallBack);

2.87 注册 SetEditSGValue 负响应回调函数

函数声明	void Reg_Svr_SetEditSGValue_N(SetEditSGValue_N callBack);
功能	SetEditSGValue 负响应
参数	SetEditSGValue_N callBack:回调函数
返回值	空
示例	void SetEditSGValue_N_CallBack(const char *server_name,SZTS_SetEditSGValueErr err){} Reg_Svr_SetEditSGValue_N (SetEditSGValue_N_CallBack);

2.84 ConfirmEditSGValues 服务函数

函数声明	int sendConfirmEditSGValuesReq(string svrName, string sgcbReference);

功能	ConfirmEditSGValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 sgcbReference:定值控制块索引
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = ″test″; string ref = “TEMPLATELD0/LLN0. SGCB”; sendConfirmEditSGValuesReq (serName , ref);

2.88 注册 ConfirmEditSGValues 正响应回调函数

函数声明	void Reg_Svr_ConfirmEditSGValues_P(ConfirmEditSGValues_P callBack);
功能	获取 ConfirmEditSGValues 正响应
参数	ConfirmEditSGValues_P callBack:回调函数
返回值	空
示例	void ConfirmEditSGValues_P_CallBack(const char *server_name,int reqId){} Reg_Svr_ConfirmEditSGValues_P (ConfirmEditSGValues_P_CallBack);

2.89 注册 ConfirmEditSGValues 负响应回调函数

函数声明	void Reg_Svr_ConfirmEditSGValues_N(ConfirmEditSGValues_N callBack);
功能	ConfirmEditSGValues 负响应
参数	ConfirmEditSGValues_N callBack:回调函数
返回	空

值	
示例	void ConfirmEditSGValues_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_ConfirmEditSGValues_N (ConfirmEditSGValues_N_CallBack);

2.90 GetEditSGValue 服务函数

函数声明	int sendGetEditSGValueReq(string svrName, deque<GdData> dataList);
功能	GetEditSGValue 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<GdData> dataList:读编辑定制区值
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; *deque*<GdData> getDatalist; GdData *getData*; *getData.reference* = "TEMPLATELD0/LLN0.LEDRs.subEna"; *getData.fc* ="SE"; getDatalist.*push_back*(*getData*); sendGetEditSGValueReq (serName, etDatalist);

2.91 注册 GetEditSGValue 正响应回调函数

函数声明	void Reg_Svr_GetEditSGValue_P(GetEditSGValue_P callBack);
功能	获取 GetEditSGValue 正响应
参数	GetEditSGValue_P callBack:回调函数
返回值	空

示例	void GetEditSGValue_P_CallBack(const char *server_name,SZTS_GetEditSGValueResp resp){} Reg_Svr_GetEditSGValue_P (GetEditSGValue_P_CallBack);

2.92 注册 GetEditSGValue 负响应回调函数

函数声明	void Reg_Svr_GetEditSGValue_N(GetEditSGValue_N callBack);
功能	ConfirmEditSGValues 负响应
参数	GetEditSGValue_N callBack:回调函数
返回值	空
示例	void GetEditSGValue_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetEditSGValue_N (GetEditSGValue_N_CallBack);

2.93 GetSGCBValues 服务函数

函数声明	int sendGetSGCBValuesReq(string svrName, deque<string> sgcbReference);
功能	GetSGCBValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<string> sgcbReference:定制控制块列表
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; deque<string> sgcbReference; sgcbReference.push_back("TEMPLATELD0/LLN0.SGCB"); sendGetEditSGValueReq (serName, sgcbReference);

2.94 注册 GetSGCBValues 正响应回调函数

函数声明	void Reg_Svr_GetSGCBValues_P(GetSGCBValues_P callBack);
功能	获取 GetSGCBValues 正响应
参数	GetSGCBValues_P callBack:回调函数
返回值	空
示例	void GetSGCBValues_P_CallBack(const char *server_name,SZTS_GetSGCBValuesResp resp){} GetSGCBValues_P callBack (GetSGCBValues_P_CallBack);

2.95 注册 GetSGCBValues 负响应回调函数

函数声明	void Reg_Svr_GetSGCBValues_N(GetSGCBValues_N callBack);
功能	GetSGCBValues 负响应
参数	GetSGCBValues_N callBack:回调函数
返回值	空
示例	void GetSGCBValues_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetSGCBValues_N (GetSGCBValues_N_CallBack);

2.96 GetBRCBValues 服务函数

函数	int sendGetBRCBValuesReq(const char * svrName, *deque*<const char *> referenceList

声明	);
功能	GetBRCBValues 请求
参数	const char * svrName: 连接服务器的名字，作为当前连接标识使用 deque<const char *> referenceList:缓存控制块列表
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; deque<string> referenceList; referenceList.push_back("TEMPLATELD /LLN0.brcbRelayDin01"); sendGetBRCBValuesReq (serName, referenceList);

2.97 注册 GetBRCBValues 正响应回调函数

函数声明	void Reg_Svr_GetBrcbValue(Svr_GetBRCBValue callBack);
功能	获取 GetBRCBValues 正响应
参数	Svr_GetBRCBValue callBack:回调函数
返回值	空
示例	void Svr_GetBRCBValue_P_CallBack(const char *server_name,SZTS_GetBRCBValuesResp* resp){} Reg_Svr_GetBrcbValue (Svr_GetBRCBValue_P_CallBack);

2.98 注册 GetBRCBValues 负响应回调函数

函数声明	void Reg_Svr_GetBrcbValueErr(Svr_GetBRCBValue_N callBack);
功能	GetBRCBValues 负响应

参数	Svr_GetBRCBValue_N callBack:回调函数
返回值	空
示例	void Svr_GetBRCBValue_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetBrcbValueErr (Svr_GetBRCBValue_N_CallBack);

2.99 SetBRCBValues 服务函数

函数声明	int sendSetBRCBValuesReq(const char * svrName, deque<SetBrcbReq_Brcb> brcbList);
功能	SetBRCBValues 请求
参数	const char * svrName: 连接服务器的名字，作为当前连接标识使用 deque<SetBrcbReq_Brcb> brcbList:缓存控制块属性列表
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque<SetBrcbReq_Brcb> brcbList;
	SetBrcbReq_Brcb brcb;
	brcb.setReference("TEMPLATELD /LLN0.brcbRelayDin01");
	brcb.setRptEna(true); //当此参数不再设置时 需重置 brcb.hasRptEna = //false
	brcbList.*push_back*(brcb);
	sendSetBRCBValuesReq (serName, brcbList);

2.100 注册 SetBRCBValues 正响应回调函数

函数声明	void Reg_Svr_SetBrcbValue(Svr_SetBRCBValue callBack);
功能	获取 SetBRCBValues 正响应

参数	Svr_SetBRCBValue callBack:回调函数
返回值	空
示例	void Svr_ Svr_SetBRCBValue_P_CallBack(const char *server_name,int reqId,const char * ref){} Reg_Svr_SetBrcbValue (Svr_ Svr_SetBRCBValue_P_CallBack);

2.101 注册 SetBRCBValues 负响应回调函数

函数声明	void Reg_Svr_SetBrcbValueErr(Svr_SetBRCBValueErr callBack);
功能	SetBRCBValuess 负响应
参数	Svr_SetBRCBValueErr callBack:回调函数
返回值	空
示例	void Svr_SetBRCBValueErr_CallBack(const char *server_name,int reqId,SZTS_SetBRCBValuesErr *err){} Reg_Svr_SetBrcbValueErr (Svr_SetBRCBValueErr_CallBack);

2.102 GetURCBValues 服务函数

函数声明	int sendGetURCBValuesReq(const char * svrName, *deque*<const char *> referenceList);
功能	GetURCBValues 请求
参数	const char * svrName: 连接服务器的名字，作为当前连接标识使用 deque<const char *> referenceList:非缓存控制块列表
返回	发送成功返回请求序号（ReqID），否则为-1;

值	
示例	const char *serName = "test"; deque<const char *> referenceList; referenceList.push_back("TEMPLATELD /LLNO.urcbRelayDin01"); sendGetURCBValuesReq (serName, referenceList);

2.103 注册 GetURCBValues 正响应回调函数

函数声明	void Reg_Svr_GetUrcbValue(Svr_GetURCBValue callBack);
功能	获取 GetURCBValues 正响应
参数	Svr_GetURCBValue callBack:回调函数
返回值	空
示例	void Svr_Svr_GetURCBValue_P_CallBack(const char *server_name,SZTS_GetURCBValuesResp * resp){} Reg_Svr_GetUrcbValue (Svr_Svr_GetURCBValue_P_CallBack);

2.104 注册 GetURCBValues 负响应回调函数

函数声明	void Reg_Svr_GetUrcbValueErr(Svr_GetURCBValue_N callBack);
功能	GetBRCBValues 负响应
参数	Svr_GetURCBValue_N callBack:回调函数
返回值	空
示例	void Svr_ Svr_GetBRCBValue_N_CallBack(const char *server_name,int reqId,ServiceError err){}

	Reg_Svr_GetUrcbValueErr (Svr_ Svr_GetBRCBValue_N_CallBack);

2.105 SetURCBValues 服务函数

函数声明	int sendSetURCBValuesReq(const char * svrName, deque<SetUrcbReq_Urcb> urcbList);
功能	SetURCBValues 请求
参数	const char * svrName: 连接服务器的名字，作为当前连接标识使用 deque<SetUrcbReq_Urcb> urcbList:非缓存控制块属性列表
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque<SetUrcbReq_Urcb> urcbList;
	SetUrcbReq_Urcb urcb;
	urcb.setReference ("TEMPLATELD /LLN0.urcbRelayDin01");
	urcb.setRptEna(true); //当此参数不再设置时 需重置 brcb.hasRptEna =
	//false
	urcbList.*push_back*(brcb);
	sendSetURCBValuesReq (serName, urcbList);

2.106 注册 SetURCBValues 正响应回调函数

函数声明	void Reg_Svr_SetUrcbValue(Svr_SetURCBValue callBack);
功能	获取 SetURCBValues 正响应
参数	Svr_SetURCBValue callBack:回调函数
返回值	空
示例	void Svr_SetURCBValue_P_CallBack(const char *server_name,int reqId,const char * ref){} Reg_Svr_SetUrcbValue (Svr_SetURCBValue_P_CallBack);

2.107 注册 SetURCBValues 负响应回调函数

函数声明	void Reg_Svr_SetUrcbValueErr(Svr_SetURCBValueErr callBack);
功能	获取 SetURCBValues 负响应
参数	Svr_SetURCBValueErr callBack:回调函数
返回值	空
示例	void Svr_SetURCBValueErr_CallBack(const char *server_name,int reqId,SZTS_SetBRCBValuesErr *err){} Reg_Svr_SetUrcbValueErr(Svr_SetURCBValueErr);

2.108 GetLCBValues 服务函数

函数声明	int sendGetLCBValuesReq(string svrName, deque<string> reference);
功能	GetLCBValues 请求
参数	const char * svrName: 连接服务器的名字，作为当前连接标识使用 deque<string> referenceList:日志控制块列表
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque<string> referenceList;
	referenceList.push_back("TEMPLATELD /LLN0.log");
	sendGetLCBValuesReq(serName, referenceList);

2.109 注册 GetLCBValues 正响应回调函数

函数	void Reg_Svr_GetLCBValues_P(GetLCBValues_P callBack);

声明	
功能	获取 GetLCBValues 正响应
参数	GetLCBValues_P callBack:回调函数
返回值	空
示例	void GetLCBValues_P_CallBack(const char *server_name,SZTS_GetLCBValuesResp resp){} Reg_Svr_GetLCBValues_P (GetLCBValues_P_CallBack);

2.110 注册 GetLCBValues 负响应回调函数

函数声明	void Reg_Svr_GetLCBValues_N(GetLCBValues_N callBack);
功能	获取 GetLCBValues 负响应
参数	GetLCBValues_N callBack:回调函数
返回值	空
示例	void GetLCBValues_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetLCBValues_N (GetLCBValues_N_CallBack);

2.111 SetLCBValues 服务函数

函数声明	int sendSetLCBValuesReq(string svrName, deque<SetLcbReq_Lcb> lcbList);
功能	SetLCBValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<SetLcbReq_Lcb> lcbList:日志控制块属性列表

返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque< SetLcbReq_Lcb > lcbList;
	SetLcbReq_Lcb lcb;
	lcb.setReference ("TEMPLATELD /LLN0.log");
	l.setRptEna(true); //当此参数不再设置时 需重置 brcb.hasRptEna = //false
	lcbList.*push_back*(lcb);
	sendSetLCBValuesReq(serName, lcbList);

2.112 注册 SetLCBValues 正响应回调函数

函数声明	void Reg_Svr_SetLCBValues_P(SetLCBValues_P callBack);
功能	获取 SetLCBValues 正响应
参数	SetLCBValues_P callBack:回调函数
返回值	空
示例	void SetLCBValues_P _CallBack(const char *server_name,int reqId){} Reg_Svr_SetLCBValues_P (SetLCBValues_P_CallBack);

2.113 注册 SetLCBValues 负响应回调函数

函数声明	void Reg_Svr_SetLCBValues_N(SetLCBValues_N callBack);
功能	获取 SetLCBValues 负响应
参数	SetLCBValues_N callBack:回调函数
返	空

回值	
示例	void SetLCBValues_N_CallBack(const char *server_name,int reqId,SZTS_SetLCBValuesErr err){} Reg_Svr_SetLCBValues_N (SetLCBValues_N_CallBack);

2.114 QueryLogByTime 服务函数

函数声明	int sendQueryLogByTimeReq(string svrName, string logReference, string startTime, string stopTime , string entryAfer));
功能	QueryLogByTime 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string logReference:日志控制块 string startTime:开始时间(格式: ”2023-02-10 20:00:00.000”) string stopTime:结束时间(格式: ”2023-02-10 20:00:00.000”) string entryAfer:日志条目标识（如 0000000000000000）
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = ″test″;sendQueryLogByTimeReq (serName, “TEMPLATELD /LLN0.log”,”2023-02-10 20:00:00.000” ,”2025-02-10 20:00:00.000”, “0000000000000000”);

2.115 注册 QueryLogByTime 正响应回调函数

函数声明	void Reg_Svr_QueryLogByTime_P(QueryLogByTime_P callBack);
功能	获取 QueryLogByTime 正响应
参数	QueryLogByTime_P callBack:回调函数
返回值	空
示	void QueryLogByTime_P_CallBack(const char

例	*server_name,SZTS_QueryLogByTimeResp resp){} Reg_Svr_QueryLogByTime_P(QueryLogByTime_P_CallBack);

2.116 注册 QueryLogByTime 负响应回调函数

函数声明	void Reg_Svr_QueryLogByTime_N(QueryLogByTime_N callBack);
功能	获取 QueryLogByTime 负响应
参数	QueryLogByTime_N callBack:回调函数
返回值	空
示例	void QueryLogByTime_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_QueryLogByTime_N (QueryLogByTime_N_CallBack);

2.117 QueryLogAfter 服务函数

函数声明	int sendQueryLogAfterReq(string svrName, string logReference, string startTime, string entryId);
功能	QueryLogAfter 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string logReference:日志控制块索引 string startTime:开始时间 (格式: ”2023-02-10 20:00:00.000”) string entryId:日志条目标识 （如 0000000000000000）
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = ″test″; string logReference = “TEMPLATELD /LLN0.log”; sendQueryLogAfterReq (serName, logReference, ”2023-02-10 20:00:00.000”,”0000000000000000”,);

2.118 注册 QueryLogAfter 正响应回调函数

函数声明	void Reg_Svr_QueryLogAfter_P(QueryLogAfter_P callBack);
功能	获取 QueryLogAfter 正响应
参数	QueryLogAfter_P callBack:回调函数
返回值	空
示例	void QueryLogAfter_P_CallBack(const char *server_name,SZTS_QueryLogAfterResp resp){} Reg_Svr_QueryLogAfter_P (QueryLogAfter_P_CallBack);

2.119 注册 QueryLogAfter 负响应回调函数

函数声明	void Reg_Svr_QueryLogAfter_N(QueryLogAfter_N callBack);
功能	获取 QueryLogAfter 负响应
参数	QueryLogAfter_N callBack:回调函数
返回值	空
示例	void QueryLogAfter_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_QueryLogAfter_N(QueryLogAfter_N_CallBack);

2.120 GetLogStatusValues 服务函数

函数声明	int sendGetLogStatusValuesReq(string svrName, deque<string> logReferenceList);

功能	GetLogStatusValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<string> logReferenceList:日志控制块索引列表
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; string logReference = "TEMPLATELD /LLN0.log"; deque<string> logReferenceList; logReferenceList.push_back(logReference); sendGetLogStatusValuesReq (serName, logReferenceList);

2.121 注册 GetLogStatusValues 正响应回调函数

函数声明	void Reg_Svr_GetLogStatusValues_P(GetLogStatusValues_P callBack);;
功能	获取 GetLogStatusValues 正响应
参数	GetLogStatusValues_P callBack:回调函数
返回值	空
示例	void GetLogStatusValues_P_CallBack(const char *server_name,SZTS_GetLogStatusValuesResp resp){} Reg_Svr_GetLogStatusValues_P (GetLogStatusValues_P_CallBack);

2.122 注册 GetLogStatusValues 负响应回调函数

函数声明	void Reg_Svr_GetLogStatusValues_N(GetLogStatusValues_N callBack);
功能	获取 GetLogStatusValues 负响应

参数	GetLogStatusValues_N callBack:回调函数
返回值	空
示例	void GetLogStatusValues_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetLogStatusValues_N (GetLogStatusValues_N_CallBack);

2.123 GetFile 服务函数

函数声明	int sendGetFileReq(const char *svrName, const char * fileName, int startPostion);
功能	GetFile 请求
参数	const char * svrName: 连接服务器的名字，作为当前连接标识使用 const char * filename:文件名 如/COMTRADE/aaa.txt int startPostion:开始位置
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; sendGetFileReq (serName, "/COMTRADE/aaa.txt", 1);

2.124 注册 GetFile 正响应回调函数

函数声明	void Reg_Svr_GetFile_P(GetFile_P callBack);
功能	获取 GetFile 正响应
参数	GetFile_P callBack:回调函数
返回	空

值	
示例	void GetFile_P_CallBack(const char *server_name,SZTS_GetFileResp resp){} Reg_Svr_GetFile_P (GetFile_P_CallBack);

2.125 注册 GetFile 负响应回调函数

函数声明	void Reg_Svr_GetFile_N(GetFile_N callBack);
功能	获取 GetFile 负响应
参数	GetFile_N callBack:回调函数
返回值	空
示例	void GetFile_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetFile_N (GetFile_N_CallBack);

2.126 SetFile 服务函数

函数声明	int sendSetFileReq(const char* svrName, const char* fileName, int startPostion, const char* fileData,int fileSize, int endOfFile);
功能	SetFile 请求
参数	const char* svrName: 连接服务器的名字，作为当前连接标识使用 const char*filename:文件名 如/COMTRADE/aaa.txt int startPostion:开始位置 const char* fileData:写文件内容(16 进制) int fileSize:需要写文件的大小 bool endOfFile: 默认给 1 即可 API 内部处理，直至写文件结束
返回	发送成功返回请求序号（ReqID），否则为-1;

值	

示例	const char *serName = "test";
	//服务文件名称
	const char* file = "/COMTRADE/aaa.txt";
	//写文件大小
	int fileSize = 2;
	//16进制文件内容
	char *fileData = (char*)*malloc*(fileSize + 1);
	for(int i = 0; i < fileSize; i++){fileData[i] = 0x15;}
	fileData[FileSize] = '\0' ;
	//如需获取写文件进度 注册 Reg_Svr_SetFile_Continue
	//开始写文件
	sendSetFileReq (serName, file,1,filedata,false);
	//写文件结束后 释放
	if(fileData){
	free(fileData); fileData = NULL;
	}

2.127 注册 SetFile 正响应回调函数

函数声明	void Reg_Svr_SetFile_P(SetFile_P callBack);
功能	获取 SetFile 正响应
参数	SetFile_P callBack:回调函数
返回值	空
示例	void SetFile_P_CallBack(){const char *server_name,int reqId} Reg_Svr_SetFile_P(SetFile_P_CallBack);

2.128 注册 SetFile 负响应回调函数

函数声	void Reg_Svr_SetFile_N(SetFile_N callBack);

明	
功能	获取 SetFile 负响应
参数	SetFile_N callBack:回调函数
返回值	空
示例	void SetFile_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_SetFile_N (SetFile_N_CallBack);

2.129 DeleteFile 服务函数

函数声明	int sendDeleteFileReq(string svrName, string fileName);
功能	DeleteFile 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string filename:文件名 如/COMTRADE/aaa.txt
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; tring file = "/COMTRADE/aaa.txt"; sendDeleteFileReq (serName, file);

2.130 注册 DeleteFile 正响应回调函数

函数声明	void Reg_Svr_DeleteFile_P(DeleteFile_P callBack);
功能	获取 DeleteFile 正响应
参数	DeleteFile_P callBack:回调函数

返回值	空
示例	void DeleteFile_P_CallBack(){const char *server_name,int reqId} Reg_Svr_SetFile_P(DeleteFile_P_CallBack);

2.131 注册 DeleteFile 负响应回调函数

函数声明	void Reg_Svr_DeleteFile_N(DeleteFile_N callBack);
功能	获取 DeleteFile 负响应
参数	DeleteFile_N callBack:回调函数
返回值	空
示例	void DeleteFile_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_DeleteFile_N (SetFile_N_CallBack);

2.132 GetFileAttributeValues 服务函数

函数声明	int sendGetFileAttributeValuesReq(string svrName, string fileName);
功能	GetFileAttributeValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string filename:文件名 如/COMTRADE/aaa.txt
返回值	发送成功返回 true，否则为 false;
示例	const char *serName = "test"; tring file = "/COMTRADE/aaa.txt"; sendGetFileAttributeValuesReq (serName, file);

2.133 注册 GetFileAttributeValues 正响应回调函数

函数声明	void Reg_Svr_GetFileAttributeValues_P(GetFileAttributeValues_P callBack);
功能	获取 GetFileAttributeValues 正响应
参数	GetFileAttributeValues_P callBack:回调函数
返回值	空
示例	void GetFileAttributeValues_P_CallBack(SZTS_GetFileAttributeValuesResp resp){} Reg_Svr_GetFileAttributeValues_P(GetFileAttributeValues_P_CallBack);

2.134 注册 GetFileAttributeValues 负响应回调函数

函数声明	void Reg_Svr_GetFileAttributeValues_N(GetFileAttributeValues_N callBack);
功能	获取 GetFileAttributeValues 负响应
参数	GetFileAttributeValues_N callBack:回调函数
返回值	空
示例	void GetFileAttributeValues_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetFileAttributeValues_N (GetFileAttributeValues_N_CallBack);

2.135 GetFileDirectory 服务函数

函数	int sendGetFileDirectoryReq(string svrName, string pathName, string startTime, string stopTime, string continueAfter);

声明	
功能	GetFileDirectory 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string pathname: 采用完整路径名，格式为“/XXXXXX” string startTime: 文件目录的起始时间（可选） string stopTime: 文件目录的截止时间 （可选） string continueAfter:从某个文件开始
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = ″test″; tring file = ″/COMTRADE″; sendGetFileAttributeValuesReq (serName, file,″″, ″″,″″);

2.136 注册 GetFileDirectory 正响应回调函数

函数声明	void Reg_Svr_GetFileDirectory_P(GetFileDirectory_P callBack);
功能	获取 GetFileDirectory 正响应
参数	GetFileDirectory_P callBack:回调函数
返回值	空
示例	void GetFileDirectory_P_CallBack(const char *server_name,SZTS_GetFileDirectoryResp resp){} Reg_Svr_GetFileDirectory_P (GetFileDirectory_P_CallBack);

2.137 注册 GetFileDirectory 负响应回调函数

函数声明	void Reg_Svr_GetFileDirectory_N(GetFileDirectory_N callBack);

功能	获取 GetFileDirectory 负响应
参数	GetFileDirectory_N callBack:回调函数
返回值	空
示例	void GetFileDirectory_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetFileDirectory_N (GetFileDirectory_N_CallBack);

2.138 Test 服务函数

函数声明	int sendTestReq(string svrName);
功能	Test 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; sendTestReq (serName);

2.139 注册 Test 超时回调函数

函数声明	void Reg_Svr_TestTimeOut(Svr_testTimeOut callBack);
功能	获取 Test 超时回调
参数	Svr_testTimeOut callBack:回调函数
返回	空

值	
示例	void Svr_testTimeOut_CallBack(const char *server_name){} Reg_Svr_TestTimeOut (Svr_testTimeOut_CallBack);

2.138 GetRpcInterfaceDirectory 服务函数

函数声明	int SendGetRpcInterfaceDirectory(string svrName, string referenceAfter);
功能	GetRpcInterfaceDirectory 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string referenceAfter:从某个索引开始
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetRpcInterfaceDirectory((serName, "");

2.139 注册 GetRpcInterfaceDirectory 正响应回调函数

函数声明	void Reg_Svr_GetRpcInterfaceDirectory_P(GetRpcInterfaceDirectory_P callBack);
功能	获取 GetRpcInterfaceDirectory 正响应
参数	GetRpcInterfaceDirectory_P callBack:回调函数
返回值	空
示例	Void GetRpcInterfaceDirectory_P_CallBack(const char *server_name,SZTS_GetRpcInterfaceDirectoryResp resp){} Reg_Svr_GetRpcInterfaceDirectory_P (GetRpcInterfaceDirectory_P_CallBack);

2.140 注册 GetRpcInterfaceDirectory 负响应回调函数

函数声明	void Reg_Svr_GetRpcInterfaceDirectory_N(GetRpcInterfaceDirectory_N callBack);
功能	获取 GetRpcInterfaceDirectory 负响应
参数	GetRpcInterfaceDirectory_N callBack:回调函数
返回值	空
示例	void GetRpcInterfaceDirectory_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetRpcInterfaceDirectory_N (GetRpcInterfaceDirectory_N_CallBack);

2.141 GetRpcMethodDIrectory 服务函数

函数声明	int SendGetRpcMethodDirectory(string svrName, string inter, string referenceAfter);
功能	GetRpcMethodDIrectory 请求
参数	String svrName: 连接服务器的名字，作为当前连接标识使用 string inter:指定的接口名称 string referenceAfter: 通知服务器只返回 referenceAfter 之后的方法
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetRpcMethodDirectory ((serName, "test2", "");

2.142 注册 GetRpcMethodDIrectory 正响应回调函数

函数声	void Reg_Svr_GetRpcMethodDirectory_P(GetRpcMethodDirectory_P callBack);

明	
功能	获取 GetRpcMethodDIrectory 正响应
参数	GetRpcMethodDirectory_P callBack:回调函数
返回值	空
示例	Void GetRpcMethodDirectory_P_CallBack(const char *server_name,SZTS_GetRpcMethodDirectoryResp resp){} Reg_Svr_GetRpcMethodDirectory_P (GetRpcMethodDirectory_P_CallBack);

2.143 注册 GetRpcMethodDIrectory 负响应回调函数

函数声明	void Reg_Svr_GetRpcMethodDirectory_N(GetRpcMethodDirectory_N callBack);
功能	获取 GetRpcMethodDIrectory 负响应
参数	GetRpcMethodDirectory_N callBack:回调函数
返回值	空
示例	void GetRpcMethodDirectory_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetRpcMethodDirectory_N (GetRpcMethodDirectory_N_CallBack);

2.144 GetRpcInterfaceDefinition 服务函数

函数声明	int SendGetRpcInterfaceDefinition(string svrName, string inter, string referenceAfter);
功能	GetRpcInterfaceDefinition 请求
参	string svrName: 连接服务器的名字，作为当前连接标识使用

数	string inter: 接口的名称 string referenceAfter: 通知服务器返回指定方法之后的结果
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; SendGetRpcInterfaceDefinition((serName, "test2", "");

2.145 注册 GetRpcInterfaceDefinition 正响应回调函数

函数声明	void Reg_Svr_GetRpcInterfaceDefinition_P(GetRpcInterfaceDefinition_P callBack);
功能	获取 GetRpcInterfaceDefinition 正响应
参数	GetRpcInterfaceDefinition_P callBack:回调函数
返回值	空
示例	Void GetRpcInterfaceDefinition_P_CallBack(const char *server_name,SZTS_GetRpcInterfaceDefinitionResp resp){} Reg_Svr_GetRpcInterfaceDefinition_P (GetRpcInterfaceDefinition_P_CallBack);

2.146 注册 GetRpcInterfaceDefinition 负响应回调函数

函数声明	void Reg_Svr_GetRpcInterfaceDefinition_N(GetRpcInterfaceDefinition_N callBack);
功能	获取 GetRpcInterfaceDefinition 负响应
参数	GetRpcInterfaceDefinition_N callBack:回调函数
返回值	空

示例	void GetRpcInterfaceDefinition_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetRpcInterfaceDefinition_N (GetRpcInterfaceDefinition_N_CallBack);

2.147 GetRpcMethodDefinition 服务函数

函数声明	int SendGetRpcMethodDefinition(string svrName, deque<string> referenceList);
功能	GetRpcMethodDefinition 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<string> referenceList: 方法的引用名列表
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; deque<string> referenceList; referenceList.push_back("test2"); SendGetRpcInterfaceDefinition((serName, referenceList);

2.148 注册 GetRpcMethodDefinition 正响应回调函数

函数声明	void Reg_Svr_GetRpcMethodDefinition_P(GetRpcMethodDefinition_P callBack);
功能	获取 GetRpcMethodDefinition 正响应
参数	GetRpcMethodDefinition_P callBack:回调函数
返回值	空
示例	Void GetRpcMethodDefinition_P_CallBack(const char *server_name,SZTS_GetRpcMethodDefinitionResp resp){} Reg_Svr_GetRpcMethodDefinition_P

	(GetRpcMethodDefinition_P_CallBack);

2.149 注册 GetRpcMethodDefinition 负响应回调函数

函数声明	void Reg_Svr_GetRpcMethodDefinition_N(GetRpcMethodDefinition_N callBack);
功能	获取 GetRpcInterfaceDefinition 负响应
参数	GetRpcMethodDefinition_N callBack:回调函数
返回值	空
示例	void GetRpcMethodDefinition_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetRpcMethodDefinition_N (GetRpcMethodDefinition_N_CallBack);

2.150 GetRpcMethodDefinition 服务函数

函数声明	int SendGetRpcMethodDefinition(string svrName, deque<string> referenceList);
功能	GetRpcMethodDefinition 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 deque<string> referenceList: 方法的引用名列表
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; deque<string> referenceList; referenceList.push_back("test2"); SendGetRpcInterfaceDefinition((serName, referenceList);

2.151 注册 GetRpcMethodDefinition 正响应回调函数

函数声明	void Reg_Svr_GetRpcMethodDefinition_P(GetRpcMethodDefinition_P callBack);
功能	获取 GetRpcMethodDefinition 正响应
参数	GetRpcMethodDefinition_P callBack:回调函数
返回值	空
示例	Void GetRpcMethodDefinition_P_CallBack(const char *server_name,SZTS_GetRpcMethodDefinitionResp resp){} Reg_Svr_GetRpcMethodDefinition_P (GetRpcMethodDefinition_P_CallBack);

2.152 注册 GetRpcMethodDefinition 负响应回调函数

函数声明	void Reg_Svr_GetRpcMethodDefinition_N(GetRpcMethodDefinition_N callBack);
功能	获取 GetRpcInterfaceDefinition 负响应
参数	GetRpcMethodDefinition_N callBack:回调函数
返回值	空
示例	void GetRpcMethodDefinition_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetRpcMethodDefinition_N (GetRpcMethodDefinition_N_CallBack);

2.153 RpcCall 服务函数

函数声明	int SendRpcCall(string svrName, string method, RpcCallReq_req req);
功能	RpcCall 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 string method: 方法的引用 RpcCallReq_req req: 调用方法的请求参数
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; RpcCallReq_req req; req.callID = "123"; SendGetRpcInterfaceDefinition((serName, "test", req);

2.154 注册 RpcCall 正响应回调函数

函数声明	void Reg_Svr_RpcCall_P(RpcCall_P callBack);
功能	获取 RpcCall 正响应
参数	RpcCall_P callBack:回调函数
返回值	空
示例	Void RpcCall_P_CallBack(const char *server_name,SZTS_RpcCallResp resp){} Reg_Svr_RpcCall_P (RpcCall_P_CallBack);

2.155 注册 RpcCall 负响应回调函数

函数	void Reg_Svr_RpcCall_N(RpcCall_N callBack);

声明	
功能	获取 RpcCall 负响应
参数	RpcCall_N callBack:回调函数
返回值	空
示例	void RpcCall_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_RpcCall_N (RpcCall_N_CallBack);

2.156 断开 TCP 连接函数

函数声明	bool ExitServer(const char *svrName, bool flag);
功能	断开 TCP 连接以及关闭数据相关线程
参数	const char * svrName: 连接服务器的名字，作为当前连接标识使用 bool flag:预留参数
返回值	成功返回 true，否则为 false;
示例	const char *serName = "test"; ExitServer(serName,false);

2.157 GetGoCBValues 服务函数

函数声明	int SendGetGoCBValue(string svrName,*deque*<string> referenceList);
功能	GetGoCBValues 请求
参	string svrName: 连接服务器的名字，作为当前连接标识使用

数	*deque*<string> referenceList: GO控制块集合
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; deque<string> referenceList; referenceList.push_back("TEMPLATELD0/LLN0.GOCB"); SendGetGoCBValue (serName, referenceList);

2.158 注册 GetGoCBValues 正响应回调函数

函数声明	void Reg_Svr_GetGoCBValues_P(GetGoCBValues_P callBack);
功能	获取 GetGoCBValues 正响应
参数	GetGoCBValues_P callBack:回调函数
返回值	空
示例	void GetGoCBValues_P_CallBack(const char *server_name,SZTS_GetGoCBValuesResp resp){} Reg_Svr_GetGoCBValues_P (GetGoCBValues_P_CallBack);

2.159 注册 GetGoCBValues 负响应回调函数

函数声明	void Reg_Svr_GetGoCBValues_N(GetGoCBValues_N callBack)
功能	获取 GetGoCBValues 负响应
参数	GetGoCBValues_N callBack:回调函数
返回值	空

示例	void GetGoCBValues_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetGoCBValues_N (GetGoCBValues_N_CallBack);

2.160 SetGoCBValues 服务函数

函数声明	int SendSetGoCBValue(string svrName, *deque*<SetGocbReq_Gocb> referenceList);
功能	SetGoCBValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 *deque*<SetGocbReq_Gocb> referenceList: GO控制块以及属性集合
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque<SetGocbReq_Gocb> referenceList;
	SetGocbReq_Gocb gocb;
	gocb.setReference("TEMPLATELD0/LLN0.GOCB");
	gocb.setGoEna(true);
	referenceList.push_back(gocb);
	SendGetGoCBValue (serName, referenceList);

2.161 注册 SetGoCBValues 正响应回调函数

函数声明	void Reg_Svr_SetGoCBValues_P(SetGoCBValues_P callBack); ;
功能	获取 SetGoCBValues 正响应
参数	SetGoCBValues_P callBack :回调函数
返回值	空
示	void SetGoCBValues_P_CallBack(const char *server_name,int reqId){}

例	Reg_Svr_SetGoCBValues_P (SetGoCBValues_P_CallBack);

2.162 注册 SetGoCBValues 负响应回调函数

函数声明	void Reg_Svr_SetGoCBValues_N(SetGoCBValues_N callBack)
功能	获取 SetGoCBValues 负响应
参数	SetGoCBValues_N callBack:回调函数
返回值	空
示例	void SetGoCBValues_N_CallBack(const char *server_name,int reqId,SZTS_SetGoCBValuesErr err){} Reg_Svr_SetGoCBValues_N (SetGoCBValues_N_CallBack);

2.163 GetMSVCBValues 服务函数

函数声明	int SendGetMsvCBValue(string svrName,*deque*<string> referenceList); ;
功能	GetMSVCBValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 *deque*<string> referenceList: MSVCB控制块集合
返回值	发送成功返回请求序号（ReqID），否则为-1;
示例	const char *serName = "test"; deque<string> referenceList; referenceList.push_back("TEMPLATECTRL/LLN0.svcb"); SendGetMsvCBValue (serName, referenceList);

2.164 注册 GetMSVCBValue 正响应回调函数

函数声明	void Reg_Svr_GetMSVCBValue_P(GetMSVCBValues_P callBack)
功能	获取 GetMSVCBValues 正响应
参数	GetMSVCBValues_P callBack:回调函数
返回值	空
示例	void GetMSVCBValues_P_CallBack(const char *server_name,SZTS_GetMSVCBValuesResp resp){} Reg_Svr_GetMSVCBValue_P (GetMSVCBValues_P_CallBack);

2.165 注册 GetMSVCBValue 负响应回调函数

函数声明	void Reg_Svr_GetMSVCBValue_N(GetMSVCBValues_N callBack)
功能	获取 GetMSVCBValues 负响应
参数	GetMSVCBValues_N callBack:回调函数
返回值	空
示例	void GetMSVCBValues_N_CallBack(const char *server_name,int reqId,ServiceError err){} Reg_Svr_GetMSVCBValue_N (GetMSVCBValues_N_CallBack);

2.166 SetMSVCBValues 服务函数

函数	int SendSetMsvCBValue(string svrName,*deque*<SetMsvcbReq_Msvcb> referenceList);

声明	
功能	SetMSVCBValues 请求
参数	string svrName: 连接服务器的名字，作为当前连接标识使用 *deque*<SetMsvcbReq_Msvcb> referenceList: MSVCB控制块以及属性集合
返回值	发送成功返回请求序号（ReqID），否则为-1;

示例	const char *serName = "test";
	deque< SetMsvcbReq_Msvcb> referenceList;
	SetMsvcbReq_Msvcb msvcb;
	msvcb.setReference("TEMPLATECTRL/LLN0.svcb");
	msvcb.setSvEn(true);
	referenceList.push_back(msvcb);
	SendSetMsvCBValue (serName, referenceList);

2.167 注册 SetMSVCBValues 正响应回调函数

函数声明	void Reg_Svr_SetMSVCBValues_P(SetMSVCBValues_P callBack) ;
功能	获取 SetMSVCBValues 正响应
参数	SetMSVCBValues_P callBack:回调函数
返回值	空
示例	void SetMSVCBValues_P_CallBack(const char *server_name,int reqId){} Reg_Svr_SetMSVCBValues_P(SetMSVCBValues_P_CallBack);

2.168 注册 SetMSVCBValues 负响应回调函数

函数声明	void Reg_Svr_SetMSVCBValues_N(SetMSVCBValues_N callBack)

功能	获取 SetMSVCBValues 负响应
参数	SetMSVCBValues_N callBack:回调函数
返回值	空
示例	void SetMSVCBValues_N_CallBack(const char *server_name,int reqId,SZTS_SetMSVCBValuesErr err){} Reg_Svr_SetMSVCBValues_N (SetMSVCBValues_N_CallBack);