SystemCORP Pty Ltd

IEC 61850 Protocol API User Manual

Protocol Integration Stack

This PDF Document contains internal hyperlinks for ease of navigation.
For example, click on any item listed in the Table of Contents to go to that page.

Documentation Control

Author:	Balaji Sreenivasan / Blake Myers / Oscar Naval
Revision:	1.02
Revision History:	1.00 Initial Release 1.01 Minor changes: Sections 3.1, 4.0, general corrections 1.02 References to Appendix page 10, Misc. re-wording grammar/spelling
Creation Date:	23 September 2009
Last Revision Date:	12 April 2010
Product Reference:	161-0100-0108
Document Status:	RELEASE

Table of Contents

1 Introduction

A Substation Automation System (SAS) has depended upon the development and availability of microprocessor-based systems. Thus, the substation equipment evolved from simple electro-mechanical devices to robust digital devices. This in turn provided the possibility of implementing SAS using several intelligent electronic devices (IEDs) to perform the required functions (e.g. protection, local and remote monitoring and control). Consequently, the need arose for efficient communication protocols among the IEDs. Until recently, specific proprietary communication protocols developed by each manufacturer have been used requiring complicated and costly protocol converters when using IEDs from different vendors.

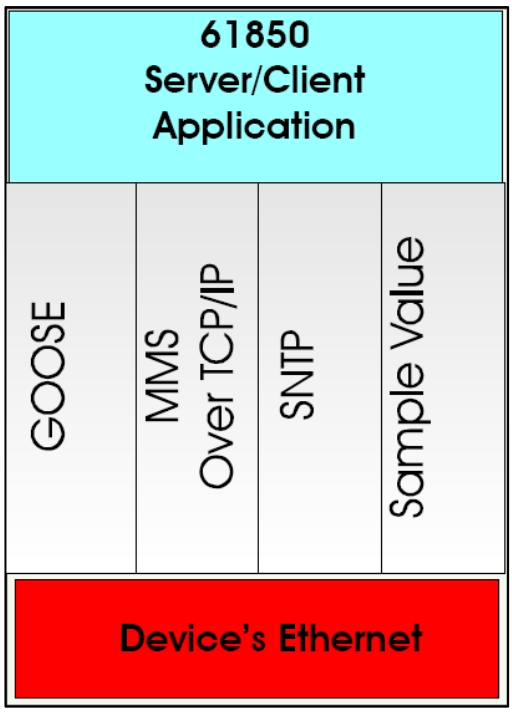
The industry's experiences have demonstrated the need for developing a standard communication protocol, which would support interoperability of IEDs from different manufacturers. Interoperability is the ability to operate on the same network or communication path sharing information and commands. Interoperability should not be confused with interchange-ability of IEDs (i.e. the ability to replace an IED supplied by one manufacturer with an IED supplied by another manufacturer without having to change other elements in the system). Interchange-ability is beyond the scope of a communication standard.

Interoperability is a common goal for electric utilities, equipment vendors and standardisation bodies. All communications must allow for the seamless integration of IEDs that allow devices from multiple vendors to be integrated together. A consensus must be found between IED manufacturers and users in a way that such devices can freely exchange information. As a result the International Electro-technical Commission (IEC) has published the IEC 61850 standard (in 10 parts, see Document Reference list).

SystemCORP Pty Ltd has a Protocol Integration Stack (PIS) that will allow you to build custom client/server applications via SystemCORP's API that meets the functional and performance communications requirements compliant with the IEC 61850 standards, therefore supporting current IEDs, future IEDs, and further substation technological developments regardless of the vendor.

1.1 IEC 61850

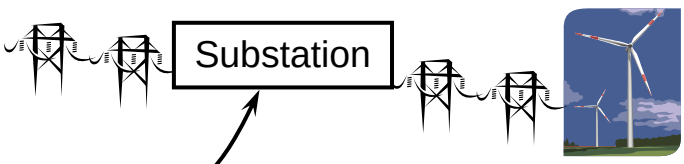
The IEC 61850 user-created application is the information exchange and service interface for substation events. This User Manual describes the Application Programming Interface (API) required to create this custom application.



Actual substation physical devices are accessed via Ethernet. From the IEC 61850 viewpoint the actual devices are a collection of logical devices made up of logical nodes.

These logical devices are mapped to the specific communication services – GOOSE, Sample Value, SNTP, or the abstract communication service interface MMS service over TCP/IP.

Generic Object Oriented Substation Events (GOOSE) bypass the TCP/IP protocol to present substation events in real-time. GOOSE is a generic substation event (GSE) that supports the exchange of a wide range of possible common data organized by a dataset.



The GOOSE messages contain information that allow the receiving device to know that a status has changed and the time of the last status change. The time of the last status change allows a receiving device to set local timers relating to a given event.

A newly activated device, upon power-up or reinstatement to service, will send current data (status) or values as the initial GOOSE message. Moreover, all devices sending GOOSE messages will continue to send the message with a long cycle time, even if no status/value change has occurred. This ensures that devices that have been activated recently will know the current status values of their peer devices. [IEC 61850-7-2]

MMS services and protocol are specified to operate over full OSI and TCP compliant communications profiles. The use of MMS allows provisions for supporting both centralized and distributed architectures. This standard includes the exchange of real-time data indications, control operations, report notification.

The Manufacturing Message Specification (MMS) protocol suite provides the information modelling methods and services required by the Abstract Communication Service Interface (ACSI). This mapping of ACSI to MMS defines how the concepts, objects, and services of the ACSI are to be implemented using MMS concepts, objects, and services. This allows interoperability across functions implemented by different IEDs regardless of the manufacturer. [IEC 61850-7-2, IEC 61850-8-1]

SNTP (Simple Network Time Protocol) is a time synchronization protocol providing time synchronization with other IEDs. SNTP protocol is widely used in synchronizing computer systems within a network. The SNTP-servers themselves are synchronized to timeservers traceable to international standards. UTC time accuracy from SNTP systems is usually in the millisecond range. SNTP provides the current time, the current number of leap seconds, and the warning flags marking the introduction of a leap second correction. [ICE 61850-8-1]

The transmission of Sampled Values (SV) requires special attention with regard to the time constraints. The model provides transmission of sampled values in an organized and time controlled way so that the combined jitter of sampling and transmission is minimized to a degree that an unambiguous allocation of the samples, times, and sequence is provided.

The SV model applies to the exchange of values of a dataset. The data of the dataset are of the common data class Sampled Analog Value (SV as defined in IEC 61850-7-3). A buffer structure is defined for the transmission of the sampled values.

The information exchange is based on a publisher/subscriber mechanism. The publisher writes the values in a local buffer at the sending side. The subscriber reads the values from a local buffer at the receiving side. A time stamp is added to the values, so that the subscriber can check the timeliness of the values. The communication system is responsible to update the local buffers of the subscribers. A sampled value control (SVC) in the publisher is used to control the communication procedure. [IEC 61850-7-2]

In general, the IEC 61850 approach is to blend the strengths of the following three methods: functional decomposition, data flow, and information modelling.

Functional decomposition is used to understand the logical relationship between components of a distributed function, and is presented in terms of logical nodes that describe the functions, sub-functions and functional interfaces.

Data flow is used to understand the communication interfaces that must support the exchange of information between distributed functional components and the functional performance requirements.

Information modelling is used to define the abstract syntax and semantics of the information exchanged, and is presented in terms of data object classes and types, attributes, abstract object methods (services), and their relationships.

1.2 Document Reference

- IEC 61850-1 Introduction and overview
- IEC 61850-2 Glossary
- IEC 61850-3 General requirements
- IEC 61850-4 System and project management
- IEC 61850-5 Communication requirements for functions and device models
- IEC 61850-6 Configuration description language for communication in electrical substations related to IED's
- IEC 61850-7-1 Basic communication structure for substation and feeder equipment – Principles and models
- IEC 61850-7-2 Basic communication structure for substation and feeder equipment – Abstract communication service interface (ACSI)
- IEC 61850-7-3 Basic communication structure for substation and feeder equipment – Common data classes
- IEC 61850-7-4 Basic communication structure for substation and feeder equipment – Compatible logical node classes and data classes
- IEC 61850-8-1 Specific communication service mapping (SCSM) – Mappings to MMS (ISO/IEC 9506-1 and ISO/IEC 9506-2) and to ISO/IEC 8802-3
- IEC 61850-9-1 Specific communication service mapping (SCSM) – Sampled values over serial unidirectional multi-drop point-to-point link
- IEC 61850-9-2 Specific communication service mapping (SCSM) – Sampled values over ISO/IEC 8802-3 2
- IEC 61850-10 Conformance testing
- IEC 61850-80-1 Ed. 1.0 Communication networks and systems for power utility automation – Part 80-1: Guideline to exchange information from a CDC based data model using IEC 60870-5-101/104

1.3 List of Abbreviations

ACSI = Abstract Communication Service Interface
API = Application Programmers Interface
ASDU = Application Service Data Unit
BRCB = Buffered Report Control Block
CDC = Common Data Class
CID = Configured IED Description
CT = Current Transducer
DA = Data Attribute
DAType = Data Attribute Type
DO = Data Object
DPS = Double Point Status information
DS = DATA-SET
DTD = Document Type Definition
DUT = Device Under Test
FAT = Factory Acceptance Test
FC = Functional Constraint
GI = General Interrogation
GoCB = GOOSE Control Block
GOOSE = Generic Object Oriented Substation Events
GSE = Generic Substation Event
GSSE = Generic Substation Status Event
GsCB = GSSE Control Block
HMI = Human Machine Interface
ICD = IED Capability Description
IEC = International Electro-technical Commission
IED = Intelligent Electronic Device
INS = Integer Status
IP = Internet Protocol
LCB = Log Control Block
LD = Logical Device
LN = Logical Node
MC = Multi-Cast
MCAA = Multicast Application Association
MICS = Model Implementation Conformance Statement
MMS = Manufacturing Message Specification (ISO 9506 series)
MSVCB = Multicast Sampled Value Control Block
OSI = Open System Interconnection
PC = Physical Connection

PD = Physical Device
PICS = Protocol Implementation Conformance Statement
PIS = Protocol Integration Stack
PIXIT = Protocol Implementation eXtra Information for Testing
RTOS = Real Time Operating System
RTU = Remote Terminal Unit
SAS = Substation Automation System
SAT = Site Acceptance Test
SAV = Sampled Analogue Value (IEC 61850-9 series)
SBO = Select Before Operate
SCADA = Supervisory Control And Data Acquisition
SCD = Substation Configuration Description.
SCL = Substation Configuration Language
SCSM = Specific Communication Service Mapping
SGCB = Setting Group Control Block
SoE = Sequence-of-Events
SPS = Single Point Status information
SSD = System Specification Description
SUT = System Under Test
SV = Sampled Values
SVC = Sampled Value Control
TCP = Transport Control Protocol
TPAA = Two Party Application Association
URCB = Unbuffered Report Control Block
USVCB = Unicast Sampled Value Control Block
UTC = Coordinated Universal Time
VT = Voltage Transducer
WDS = WebCAN Designer Studio
XML = eXtensible Markup Language

2 Overview

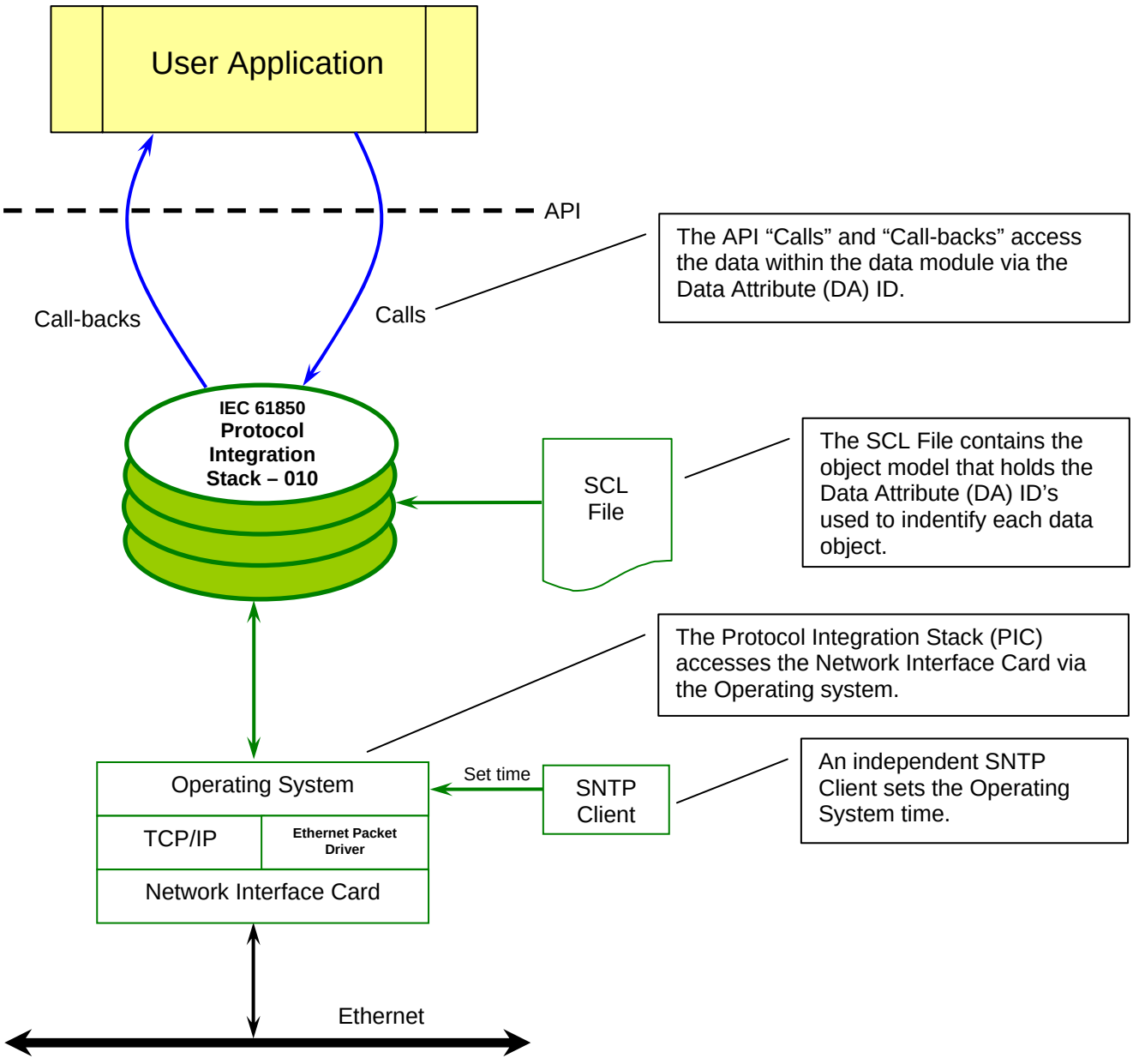
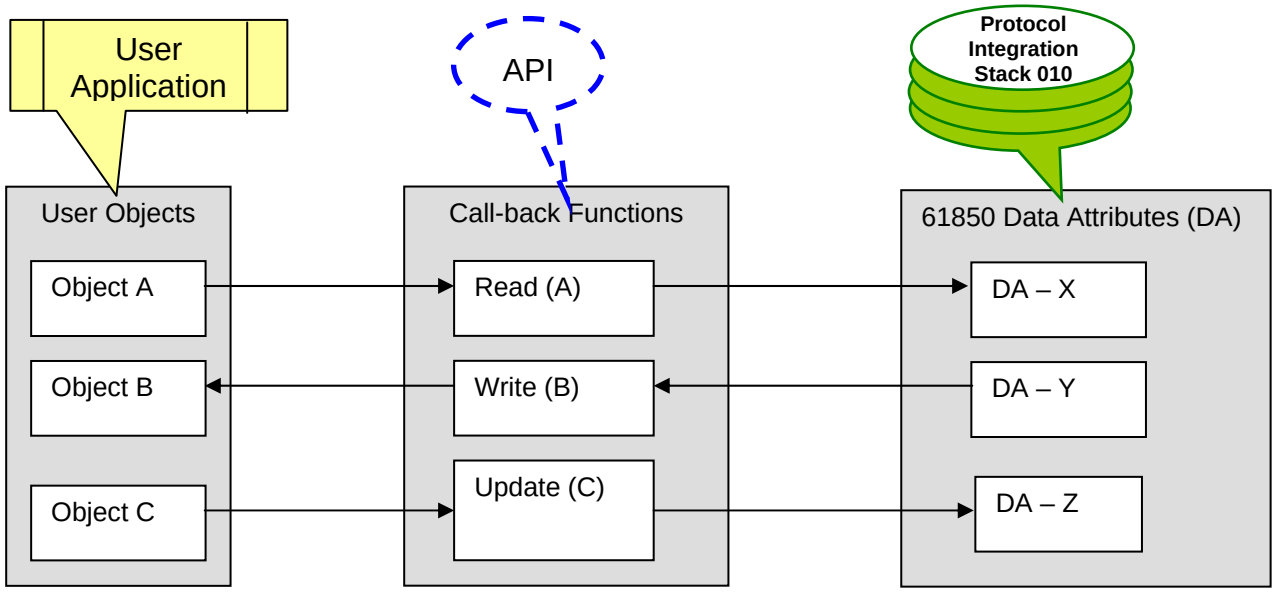


Figure 2-1 – Context Diagram

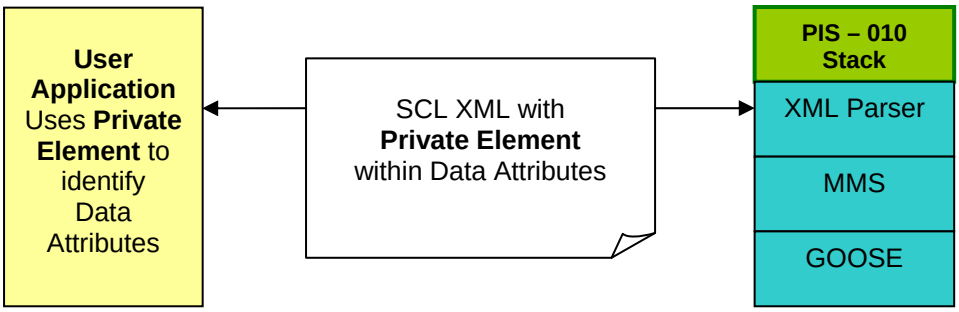
2.1 Protocol Integration Stack – API Overview

The PIS -010 implementation allows users to integrate a Server or Client applications easily. It uses the Substation Configuration Language (SCL) XML file format using “Private Elements” for mapping the 61850 objects to User Application objects using the Data Attribute ID. The call-back functions must be provided to the PIS-010 for converting the mapped objects to 61850 objects.



The user application objects are read, written or updated using the call-back functions with Data Attribute (DA) ID.

2.2 Protocol Integration Stack –Configuration Overview



An SCL configuration must be provided to the PIS-010 for it to configure an IEC 61850 data template.

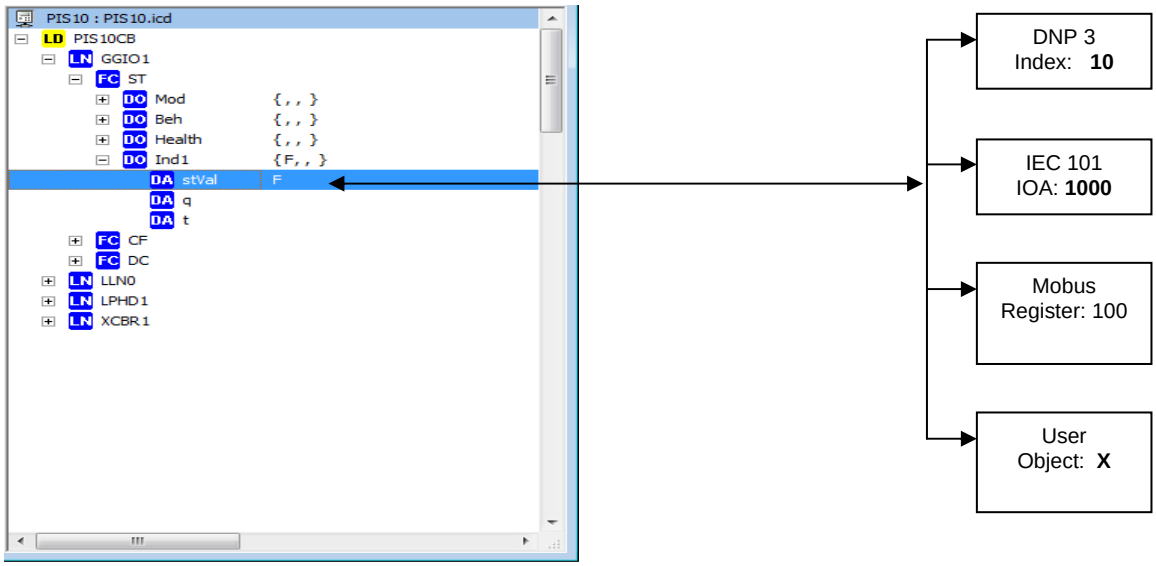
The SCL uses “Private Elements” as part of the Data Attribute Information (DAI). Private Elements describe and link user application objects, which are part of the user application, with IEC 61850 Data Attributes (DA). This can also be described as the mapping mechanism between user application and the IEC 61850 stack. Use either the SystemCORP WebCAN ICD Designer or any third-party tool that can read the XML schema to generate the SCL file. The WebCAN ICD Designer uses “SystemCorp_Generic” as Private Elements.

Example Server Program is in Appendix 7.3.1. Also, see “Appendix 7.3.2 – CID File for Server” for an example SCL (a CID – Configured IED Description) file created by the SystemCORP WebCAN ICD Designer.

Example Client Program is in Appendix 7.3.3.

3 User Data Attributes (DA)

The SCL (Substation Configuration Language) files include all the information needed to configure IEDs, communication networks and substation topologies. The SCL file is described in a XML format so it can be easily interpreted and transformed. The schema restricts the information allowed and it assures that its information can be processed by different tools. As an example, below is the Omicron IED Scout showing one possible mapping from a source.



Using “Private Elements”, the 61850 data attributes can than be mapped by any user configuration tool into objects such as a DNP 3 Index, IEC 101 Information Object Address, Modbus Register and any other user specific object as the Private Element as part of the configuration.

3.1 Configuration of User Data Attribute Information (DAI)

The Private section is included inside the “DAI” element. Below is an example of how it is used. The examples describe the use for a generic application when `Private type="SystemCORP_Generic"` is used. In the examples all the information has been included at the DAI level:

3.1.1 Single Point Status (SPS) common data class example

```
<DAI name="stVal">
        <Private type="SystemCorp_Generic">
                <SystemCorp_Generic:GenericPrivateObject
                        Field1="1" Field2="1" Field3="1" Field4="0" Field5="0"
                        xmlns:SystemCorp_Generic=
                        "http://www.systemcorp.com.au/61850/SCL/Generic"/>
        </Private>
</DAI>
```

The contents of the “Field[1-5]” attributes are populated by the specific user application requirements.

3.1.2 Single Point Command (SPC) common data class example

```
<DOI name="SPCSO1">
        <SDI name="Oper">
                <DAI name="ctlVal">
                        <Private type="SystemCorp_Generic">
                                <SystemCorp_Generic:GenericPrivateObject
                                        Field1="1" Field2="2" Field3="1" Field4="0" Field5="0"
                                        xmlns:SystemCorp_Generic=
                                        "http://www.systemcorp.com.au/61850/SCL/Generic"/>
```

```
                    </Private>
                </DAI>
            </SDI>
            <DAI name="ctlModel" valKind="Set">
                <Val>direct-with-normal-security</Val>
            </DAI>
        </DOI>
```

Note: Both the above examples shows a `xmlns:SystemCorp_Generic` http link that is not active. It is an XML name space (`xmlns`) used for validation only according to the IEC 61850 specification.

Field Attributes:
Both the SPS and SPC examples above use the `SystemCorp_Generic:GenericPrivateObject` Field attributes. “Field1” to “Field5” are used to identify the mapping between IEC 61850 data attributes and the user data objects. The meaning and contents of this field attributes has to be defined by the application programmer.

For example, using the DK61 Development Kit the Field meanings and contents are:

Field #	Meaning:	Content (value):
Field1	For DIP switches 1 through 8 =	1, 2, 3, 4, 5, 6, 7, or 8
	For the 8LED’s =	1, 2, 3, 4, 5, 6, 7, or 8
Field2	For DIP switches (Digital Inputs) =	1
	For LED’s (Digital Outputs) =	2
Field3	Value =	1
	Status =	2
	Time =	3
Field4	Unused	
Field5	Unused	

Appendix 7.4 contains the schema for the Private Elements.

3.2 User Data Attributes Access via API

Once loaded the user Data Attributes can be accessed via read, write, update functions by specifying the DA ID that is assigned to the required DA.

4 Protocol Stack

This section describes the interface provided between the PIS-010 and user application. The API is divided into two categories listed below.

- Client/Server Management
- Data Attributes Access

When integrating the IEC 61850 stack into a third party software application the programmer is provided with call-back functions described below. No programming inside the stack software is required. Integration work is required linking the stack to the Ethernet driver environment of the target system.

The tables below summaries all API functions needed for interfacing a user application to the PIS-10 IEC 61850 stack.

The Client/Server Management functions are

No	API	Purpose
1	**IEC61850_Create**	API to create a client or server object with call-backs for reading, writing and updating data objects
2	**IEC61850_LoadSCLFile**	API to read the SCL XML file to get the configuration of server or client
3	**IEC61850_Start**	API to start the server or client
4	**IEC61850_Stop**	API to stop the server or client
5	**IEC61850_Free**	API to delete a client or server object created

The Data Attributes Access functions are

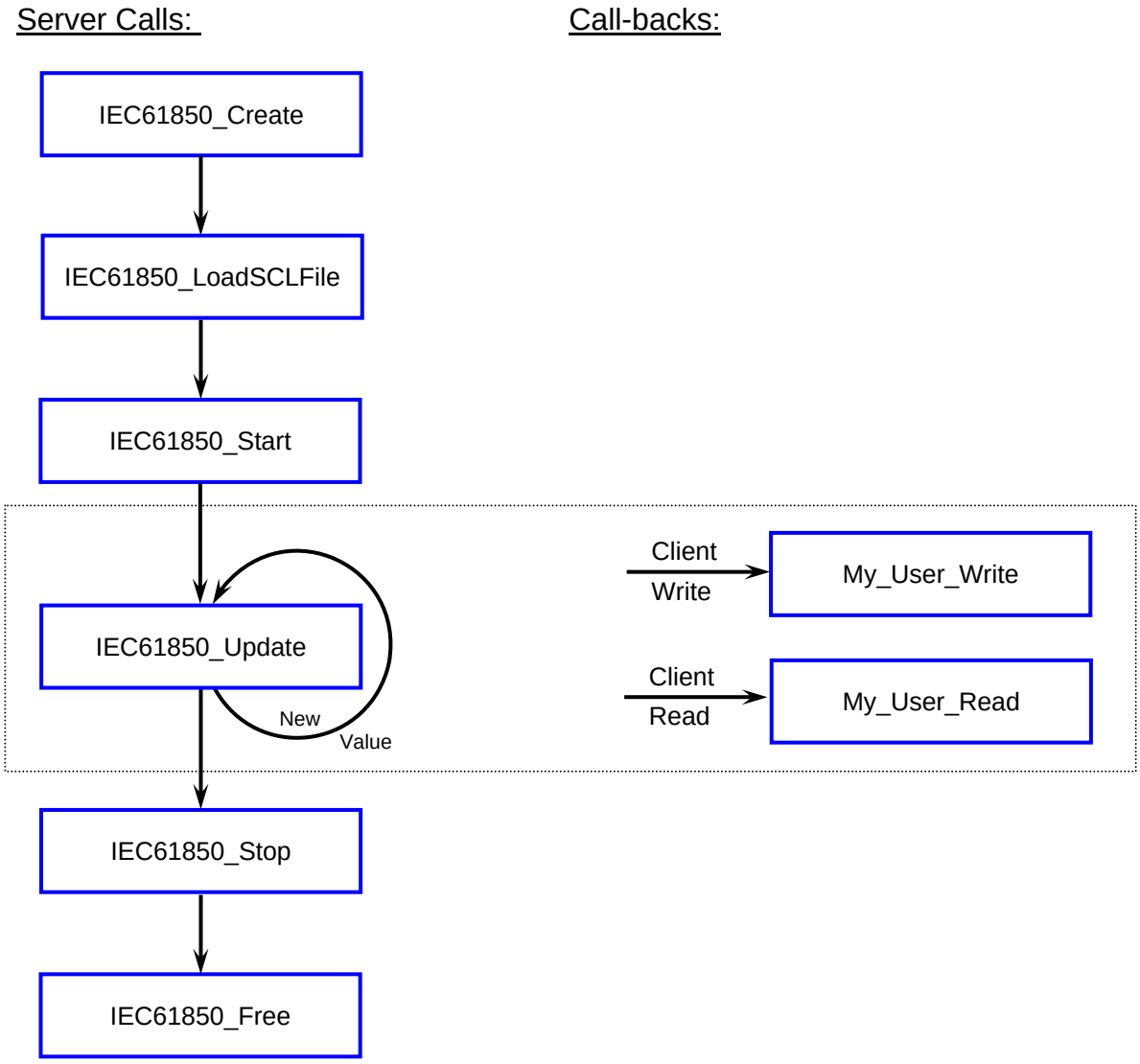
No	API	Purpose
1	**IEC61850_Read**	Read the value of a specified data attribute
2	**IEC61850_Write**	Write the value to a specified data attribute
3	**IEC61850_Update**	Update the value of a specified data attribute

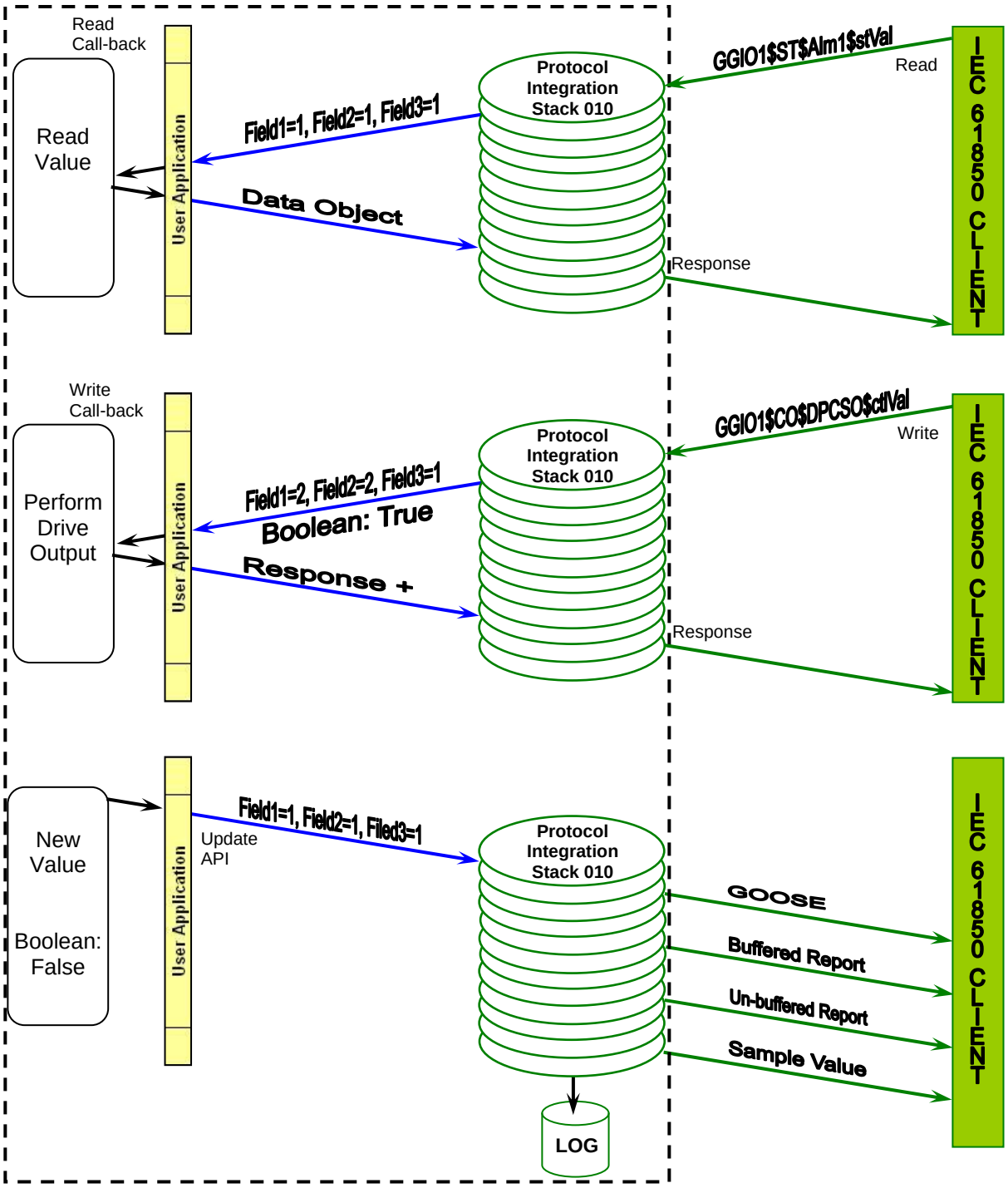
User Objects are managed by using call-back functions. In case of server the read and write call-back must be provided by the user application when using the ***"IEC61850_Create"*** function so that PIS-010 call these functions when it reads are writes. In case of client, the ***"IEC61850_Update"*** function call-back provided during the client creation is used to update the user objects.

The following sections provided data flow between PIS-010 and a user application.

4.1 Server

The section describes how the data is exchanged between PIS-010 and server user application.

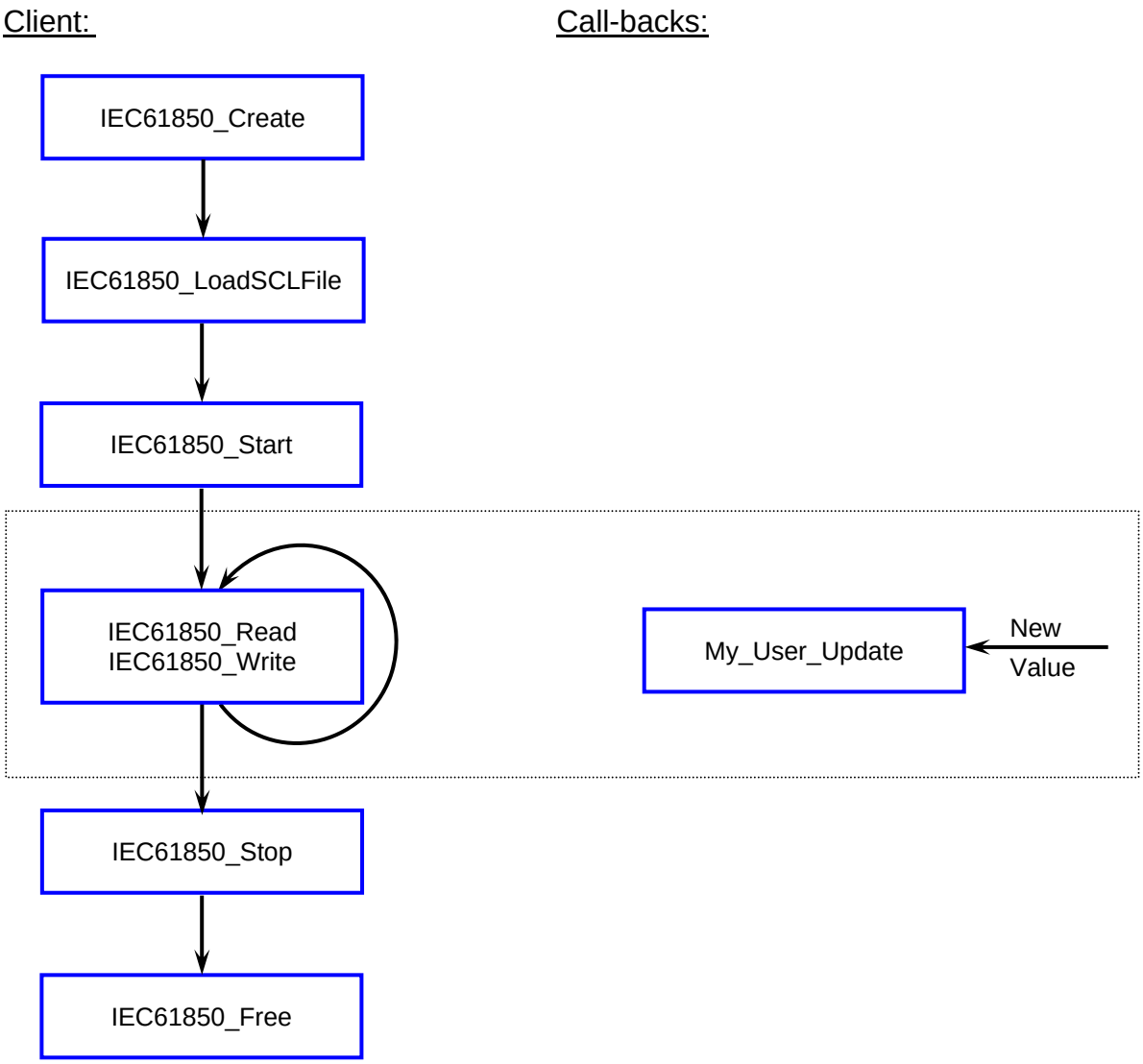


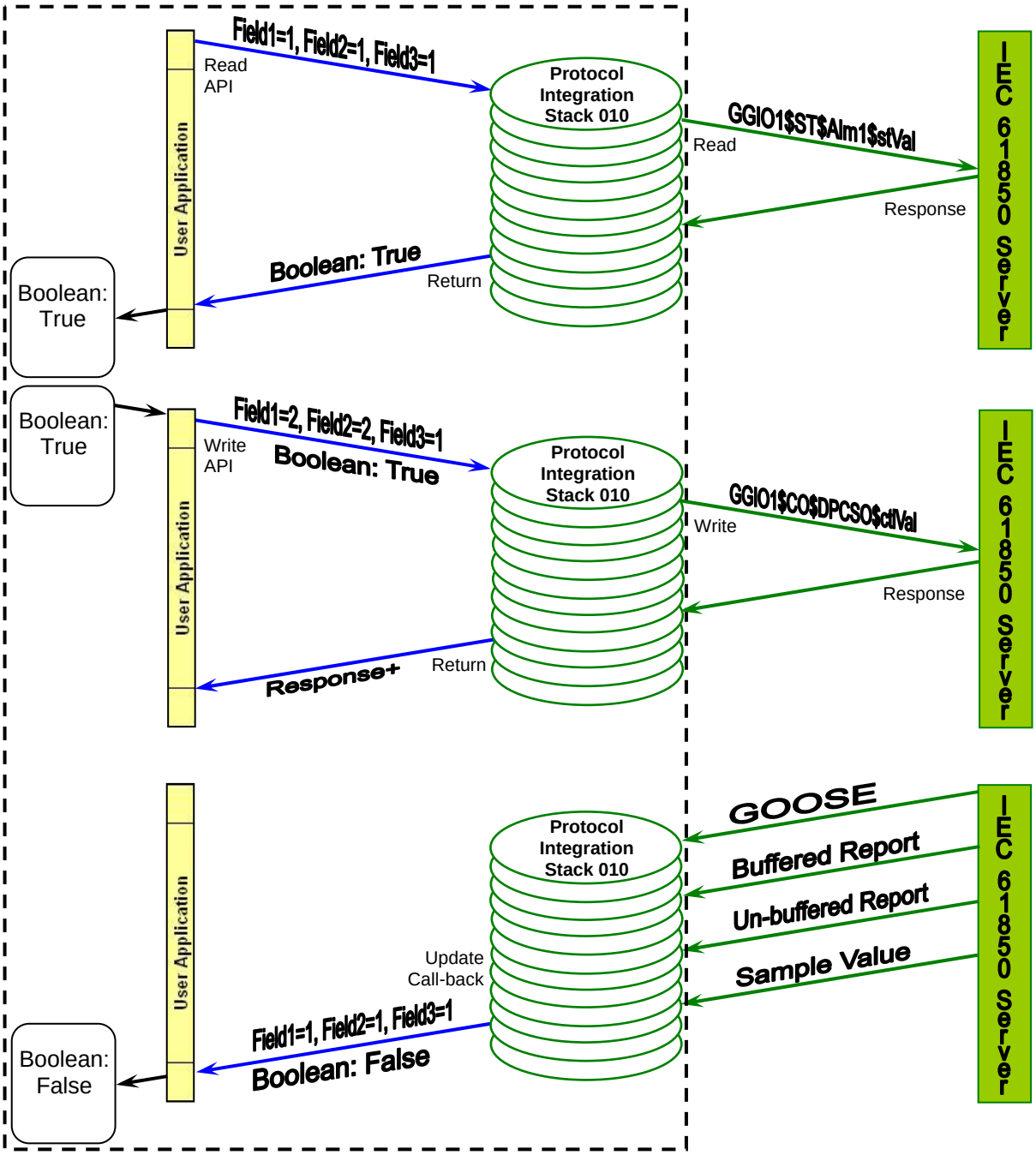


The SCD file defines all subscribers for GOOSE services. GOOSE messages are then automatically generated by the PIS-10 stack.

4.2 Client

The section describes how the data is exchanged between PIS-010 and client user application.





The SCD file defines all GOOSE services as subscriber. The PIS-10 stack then processes all incoming GOOSE messages accordingly.

5 API Module Reference and Usage

Error codes in Appendix 7.1

(See IEC 61850-7-2:2003 section 5.5.3.4 for Read Write and Update callbacks returned service error codes.).

5.1 Client/Server Management

Data Structures

struct IEC61850_Parameters
➔ *Create Server/client parameters structure*.

Typedefs

- `typedef struct IEC61850_Struct * IEC61850`
  ➔ *Pointer to a IEC 61850 object*.

- `typedef int(* IEC61850_ReadCallback )(IEC61850_ObjectID *ptObjectID, IEC61850_ObjectData *ptReturnedValue)`
  ➔ *Read Callback definition. This function should return a value from the available IEC61850_CallbackReturnServiceErrorCodes*.

- `typedef int(* IEC61850_WriteCallback )(IEC61850_ObjectID *ptObjectID, const IEC61850_ObjectData ptNewValue)`
  ➔ *Write callback definition. This function should return a value from the avaliable IEC61850_CallbackReturnServiceErrorCodes*.

- `typedef void( IEC61850_UpdateCallback )(IEC61850_ObjectID ptObjectID, const IEC61850_ObjectData ptNewValue)`
  ➔ *Update callback definition*.

Enumerations

- `enum IEC61850_ClientServerFlag {IEC61850_SERVER = 0, IEC61850_-CLIENT = 1}`
  ➔ *Server/client parameter*.

Functions

- `IEC61850 _IEC61850_Create (IEC61850_Parameters *ptParameters, int *errorCode)`
  ➔ *Create a New IEC61850 object*.

- `void _IEC61850_Free (IEC61850 clientServerObject)`
  ➔ Free memory used by IEC 61850 object.

- `int _IEC61850_LoadSCLFile (IEC61850 clientServer, char *SCLFileName)`
  ➔ *Load a SCL file into the IEC 61850*.

- `int _IEC61850_Start (IEC61850 clientServer)`
  ➔ *Start IEC61850 object communications*.

- `int _IEC61850_Stop (IEC61850 clientServer)`
  ➔ *Stop IEC61850 object communications*.

5.1.1 Enumeration Type Documentation

5.1.1.1 enum IEC61850_ClientServerFlag

Server/client parameter.

Enumerator:

IEC61850_SERVER → This IEC61850 Object is a Client
IEC61850_CLIENT → This IEC61850 Object is a Server

5.1.2 Function Documentation

5.1.2.1 IEC61850 _ IEC61850_Create (IEC61850_Parameters * ptParameters, int * errorCode)

Create a New IEC61850 object.

Parameters:

← ***ptParameters*** IEC 61850 Object Parameters.
→ ***errorCode*** pointer to return integer for error code if an error occurred.

Returns:

Pointer to a new IEC 61850 object
NULL if an error occurred (errorCode will contain an error code)

Server example usage:

```
// Create Server
IEC61850 myServer;
IEC61850_Parameters tServerParam;
int error;

tServerParam.ClientServerFlag = IEC61850_SERVER;
tServerParam.ptReadCallback = MyReadFunction;
tServerParam.ptWriteCallback = MyWriteFunction;
tServerParam.ptUpdateCallback = NULL;

myServer = IEC61850_Create(&tServerParam,&error);
//Create a server
if(myServer == NULL)
{
      printf("Server Failed to create: %i", error);
}
```

Client example usage:

```
// Create Client
IEC61850 myClient;
IEC61850_Parameters tClientParam;
int   error;

tClientParam.ClientServerFlag = IEC61850_CLIENT;
tClientParam.ptReadCallback = NULL;
tClientParam.ptWriteCallback = NULL;
tClientParam.ptUpdateCallback = MyUpdateFunction;
myClient = IEC61850_Create(&tClientParam,&error);
//Create a client
if(myClient == NULL)
{
      printf("Client Failed to create:%i",error);
}
```

5.1.2.2 void _IEC61850_Free (IEC61850 clientServerObject)

Free memory used by IEC 61850 object.

Parameters:
← ***clientServerObject*** Server/Client object to free

Example Usage:
```
IEC61850_Free(myServer);      //Frees myServer
```

5.1.2.3 int _IEC61850_LoadSCLFile (IEC61850 clientServer, char * SCLFileName)

Load a SCL file into the IEC 61850.

Parameters:
← ***clientServer*** client/Server object
← ***SCLFileName*** File name of the SCL file

Returns:
IEC61850_ERROR_NONE *on* success otherwise error code

Example Usage:
```
error = IEC61850_LoadSCLFile(myServer,"myIDE.icd");
// Load in my IDE.icd file
if(error != IEC61850_ERROR_NONE)
{
      printf("Loading SCL file has failed: %i",error);
}
```

5.1.2.4 int _IEC61850_Start (IEC61850 clientServer)

Start IEC61850 object communications.

Parameters:
clientServer client or Server object to start

Returns:
IEC61850_ERROR_NONE on success otherwise error code

Example Usage:
```
error = IEC61850_Start(myServer); // Starts myServer
if(error != IEC61850_ERROR_NONE)
{
      printf("Can't start server: %i",error);
}
```

5.1.2.5 int _IEC61850_Stop (IEC61850 clientServer)

Stop IEC61850 object communications.

Parameters:
clientServer client or Server object to stop

Returns:
IEC61850_ERROR_NONE on success otherwise error code

Example Usage:

```
error = IEC61850_Stop(myServer);    // Stops myServer
if(error != IEC61850_ERROR_NONE)
      {
            printf("Can't stop server: %i",error);
      }
```

5.2 Data IO

Data Structures

- struct IEC61850_ObjectID

  This structure holds the identification of a IEC61850 data object. This is an example structure that matches the PIS10 schema. This can be customised to suit your requirements.

- struct IEC61850_ObjectData

  A Data object structure used to exchange data objects between IEC61850 object and application.

Typedefs

- typedef unsigned int tIEC61850Quality

  IEC61850 Quality data type. (as specified in IEC61850_DATATYPE_TIMESTAMP).

IEC 61850 Data Type Enumerations in Appendix 7.2

Functions

- `int _IEC61850_Update (IEC61850 server, IEC61850_-ObjectID *ptObjectID, const IEC61850_ObjectData *ptNewValue)`

Update a given Object of tObjectID with value of ptNewValue.

- `int _IEC61850_Read (IEC61850 client, IEC61850_ObjectID *ptObjectID, IEC61850_ObjectData *ptReturnedValue)`

Read a value to a given Object ID via the client.

- `int _IEC61850_Write (IEC61850 client, IEC61850_-ObjectID ptObjectID, const IEC61850_ObjectData *ptNewValue)`

Write a value to a given Object ID via the client.

5.2.1 Function Documentation

5.2.1.1 *int _ IEC61850_Read (IEC61850 client, IEC61850_ObjectID * ptObjectID, IEC61850_ObjectData * ptReturnedValue)*

Read a value to a given Object ID via the client.

Parameters:

← client Client object to read from
← ptObjectID Pointer to Object ID that is to be read
→ ptReturnedValue Pointer to Object Data structure that hold the returned value of the tObjectID

Returns:

IEC61850_ERROR_NONE on success otherwise error code

Example Usage:

```
IEC61850_ObjectData Value;
IEC61850_ObjectID Object;
Unsigned32 u32Counter;

// Load Data
Value.ucType = IEC61850_DATATYPE_INT32;
Value.uiBitLength = sizeof(u32Counter)*8;
Value.pvData = &u32Counter;

// Load ID
Object.uiNumber = 43;

error = IEC61850_Read(myServer,&Object,&Value);
// Read object with uiNumber = 43
if(error != IEC61850_ERROR_NONE)
{
      printf("error has occured: %i",error);
}
else
{
      printf("Count = %u",u32Counter);
}
```

5.2.1.2 *int _ IEC61850_Update (IEC61850 server IEC61850_ObjectID * ptObjectID, const IEC61850_ObjectData * ptNewValue)*

Update a given Object of tObjectID with value of ptNewValue.

Parameters:

← ***server*** Server/Client object to update
← ***ptObjectID*** Pointer to Object ID that has been updated
← ***ptNewValue*** Pointer to Object Data structure that hold the new value of the object

Returns:

IEC61850_ERROR_NONE on success otherwise error code

Example Usage:

```
IEC61850_ObjectData Value;
IEC61850_ObjectID Object;
Unsigned32 u32Counter;
u32Counter = 34;

// Load Data
Value.ucType = IEC61850_DATATYPE_INT32;
Value.uiBitLength = sizeof(u32Counter)*8;
Value.pvData = &u32Counter;

// Load ID
Object.uiNumber = 596;
error = IEC61850_Update(myServer, &Object, &Value);
// Update object with uiNumber = 596 with value of 34

if(error != IEC61850_ERROR_NONE)
{
      printf("Update failed: %i",error);
}
```

5.2.1.3 *API int _ IEC61850_Write (IEC61850 client,IEC61850_ObjectID * ptObjectID, const IEC61850_ObjectData * ptNewValue)*

Write a value to a given Object ID via the client.

Parameters:

← ***client*** Client object to write to
← ***ptObjectID*** Pointer to Object ID structure that is to be written
← ***ptNewValue*** Pointer to Object Data structure that hold the new value of the tObjectID

Returns:

IEC61850_ERROR_NONE on success otherwise error code

Example Usage:

```
IEC61850_ObjectData Value;
IEC61850_ObjectID Object;
char bFlag;
bFlag = 1; // Set flag equal to True

// Load Data
Value.ucType = IEC61850_DATATYPE_BOOLEAN;
Value.uiBitLength = sizeof(bFlag)*8;
Value.pvData = &bFlag;

// Load ID
Object.uiNumber = 36;
error = IEC61850_Write(myServer, &Object, &Value);
// Update object with uiNumber = 36 with value of TRUE

if(error != IEC61850_ERROR_NONE)
{
      printf("Write has failed: %i",error);
}
```

6 API Data Structure

6.1 IEC61850_ObjectData Struct Reference

A Data object structure. Used to exchange data objects between IEC61850 object and application.

```
#include <IEC61850API.h>
```

Data Fields

Void* pvData	Pointer to data of length equal to uiBitLength bits
unsigned char ucType	Type of data. Values can from IEC61850_DataTypes
unsigned int uiBitLength	Bit Length of data at pvData (NOTE: This is in Bits! So one octel equal to 8)

A Data object structure. Used to exchange data objects between IEC61850 object and application.

6.2 IEC61850_ObjectID Struct Reference

This structure hold the identification of a IEC61850 data object. This is an example structure that matches the PIS10 schema. This can be customised to suit your requirements.

```
#include <IEC61850API.h>
```

Data Fields

unsigned int uiCFENumber	The ID Number of the CFE that this data belongs. (Equal to CFENo attribute in the PIS10 schema)
unsigned int uiClass	Data Class of this data instance (Equal to Class attribute in the PIS10 schema)
unsigned int uiNumber	The Object number of this data instance (Equal to Number attribute in the PIS10 schema)
unsigned int uiField	The data field this data belongs. (Data, Quality or Timestamp) (Equal to Field attribute in the PIS10 schema)
unsigned int uiCommandPort	The Command Port of this object

This structure hold the identification of a IEC61850 data object. This is an example structure that matches the PIS10 schema. This can be customised to suit your requirements.

6.3 IEC61850_Parameters Struct Reference

Create Server/client parameters structure.

```
#include <IEC61850API.h>
```

Data Fields

enum IEC61850_ClientServerFlag ClientServerFlag	Flag set to indicate if this is to be a server (ClientServerFlag = 0) or a client (ClientServerFlag = 1)
IEC61850_ReadCallback ptReadCallback	Read callback function. If equal to NULL then callback is not used.
IEC61850_WriteCallback ptWriteCallback	Write callback function. If equal to NULL then callback is not used.
IEC61850_UpdateCallback ptUpdateCallback	Update callback function. If equal to NULL then callback is not used.

Create Server/client parameters structure.

6.4 IEC61850TimeStamp Struct Reference

IEC61850 NTP Time Stamp Structure (as specifed in IEC61850_DATATYPE_-TIMESTAMP).

```
#include <IEC61850API.h>
```

Data Fields

unsigned long int u32Seconds	Number of Seconds.
unsigned long int u32FractionsOfSecond	Fraction of a second as a binary fraction.

IEC61850 NTP Time Stamp Structure (as specifed in IEC61850_DATATYPE_-TIMESTAMP).

7 Appendix

7.1 IEC 61850 Error Codes

enum IEC61850_ErrorCodes

Enumerator:

IEC61850_ERROR_NONE	**0**	Everything was ok
IEC61850_ERROR_INVALID_PARAMETERS	**-1**	Supplied parameters are invalided
IEC61850_ERROR_NO_MEMORY	**-2**	Allocation of memory has failed
IEC61850_ERROR_SCL_FILE_OPEN_FAILED	**-3**	Provided SCL file failed to load
IEC61850_ERROR_SERVICE_FAILED	**-4**	Service failed to start
IEC61850_ERROR_SCL_SYNTAX_ERROR SCL	**-5**	File Failed to parse due to syntax error
IEC61850_ERROR_SCL_IO_ERROR SCL	**-6**	File Failed to parse due IO error
IEC61850_ERROR_SCL_NO_IED_CONNECTEDAP	**-7**	Can't find a matching Connected AP for this IED
IEC61850_ERROR_TYPE_MISMATCH	**-8**	The Type you are trying to write/update does not match the type in the server/client
IEC61850_ERROR_LICENCE_INVALID	**-9**	The licence file is not valid or present
IEC61850_ERROR_OBJECTID_NOT_FOUND	**-10**	The given Object ID was not found in the loaded SCL file
IEC61850_ERROR_OBJECTID_NO_CONNECTION	**-11**	There is no ready connection to given Object ID
IEC61850_ERROR_SCL_LN_TYPE_NOT_FOUND The	**-12**	LN Type specified in the LN element was not found in the data template
IEC61850_ERROR_SCL_DATASET_NOT_FOUND	**-13**	The Dataset specified in a the control block was not found
IEC61850_ERROR_SCL_GSE_COMM_NOT_FOUND	**-14**	The GSE Communication access point specified in a the control block was not found
IEC61850_ERROR_SCL_SV_COMM_NOT_FOUND	**-15**	The Sampled Value Communication access point specified in a the control block was not found
IEC61850_ERROR_SCL_INVALID_MAC_ADDRESS	**-16**	The SCL contains a Invalid MAC address
IEC61850_ERROR_GOOSE_INIT_FAILED GOOSE	**-17**	Service failed to initialise
IEC61850_ERROR_SV_INIT_FAILED	**-18**	Sampled Value Service failed to initialise
IEC61850_ERROR_MMS_SAP_FAILED	**-19**	Unable to create a service access point for the MMS server
IEC61850_ERROR_SCL_DATA_TYPE_TEMPLATE_NAME	**-20**	Missing or invalid Data type name in Data Type Template
IEC61850_ERROR_SCL_DATA_TYPE_TEMPLATE_ID	**-21**	Missing or invalid Data type ID in Data Type Template
IEC61850_ERROR_SCL_DATASET_NAME	**-22**	Missing or invalid Data set Name
IEC61850_ERROR_SCL_IED_ELEMENT_NAME	**-23**	Missing or Invalid IED element name (e.g. name attribute for IED, LD, LN, DOI, SDI or DAI)

IEC61850_ERROR_SCL_COMM_NAME	**-24**	Missing or Invalid Communication element name
IEC61850_ERROR_INVALID_STATE	**-25**	The function cannot be performed while the client/server object is in the current state

enum IEC61850_CallbackReturnServiceErrorCodes

Enumerator:

IEC61850_CB_ERROR_NONE	**0**	Everything when OK
IEC61850_CB_ERROR_INSTANCE_NOT_AVAILABLE	**-1**	Action failed due to instance being not available
IEC61850_CB_ERROR_ACCESS_VIOLATION	**-2**	Action failed due to access violation
IEC61850_CB_ERROR_PARAMETER_VALUE_INCONSISTENT	**-3**	Action failed due to inconsistent parameter value
IEC61850_CB_ERROR_INSTANCE_LOCKED_BY_OTHER_CLIENT	**-4**	Action failed due to data locked by other client
IEC61850_CB_ERROR_TYPE_CONFLICT	**-5**	Action failed data type conflict
IEC61850_CB_ERROR_FAILED_DUE_TO_SERVER_CONSTRAINT	**-6**	Action failed due to server constraint

7.2 IEC 61850 Data Types

enum IEC61850_DataTypes

IEC61850 Data Types (see IEC 61850-7-2:2003 section 5.5.2).

IEC61850_DATATYPE_BOOLEAN	**1**	Data is of type Boolean. If value is equal to 0 then false, otherwise it true
IEC61850_DATATYPE_INT8	**2**	An integer of 8 bits.
IEC61850_DATATYPE_INT16	**3**	An integer of 18 bits.
IEC61850_DATATYPE_INT32	**4**	An integer of 32 bits.
IEC61850_DATATYPE_INT8U	**5**	An unsigned integer of 8 bits.
IEC61850_DATATYPE_INT16U	**6**	An unsigned integer of 16 bits.
IEC61850_DATATYPE_INT32U	**7**	An unsigned integer of 32 bits.
IEC61850_DATATYPE_FLOAT32	**8**	A IEEE 754 single precision floating point
IEC61850_DATATYPE_FLOAT64	**9**	A IEEE 754 double precision floating point
IEC61850_DATATYPE_ENUMERATED	**10**	Ordered set of values. extended allowed
IEC61850_DATATYPE_CODED_ENUM	**11**	Ordered set of values. Not allowed to be extended
IEC61850_DATATYPE_OCTEL_STRING	**12**	A String of Octels characters
IEC61850_DATATYPE_VISIBLE_STRING	**13**	A String of Visible characters
IEC61850_DATATYPE_UNICODE_STRING	**14**	A String of Unicode characters
IEC61850_DATATYPE_TIMESTAMP	**15**	TimeStamp type (5.5.3.7.1 of IEC 61850-7-2:2003)
IEC61850_DATATYPE_QUALITY	**16**	Quality Data type

enum IEC61850QualityFlags

IEC61850 Quality flags. (see IEC 61850-8-1:2004 section 8.2).

IEC61850_QUALITY_INVALID	**0x4000**	Invalid
IEC61850_QUALITY_RESERVED	**0x8000**	Reserved
IEC61850_QUALITY_QUESTIONABLE	**0xC000**	Questionable
IEC61850_QUALITY_OVERFLOW	**0x2000**	Overflow
IEC61850_QUALITY_OUTOFRANGE	**0x1000**	Out of Range
IEC61850_QUALITY_BADREFERENCE	**0x0800**	Bad Reference
IEC61850_QUALITY_OSCILLATORY	**0x0400**	Oscillatory
IEC61850_QUALITY_FAILURE	**0x0200**	Failure
IEC61850_QUALITY_OLDDATA	**0x0100**	Old Data
IEC61850_QUALITY_INCONSISTENT	**0x0080**	Inconsistent
IEC61850_QUALITY_INACCURATE	**0x0040**	Inaccurate
IEC61850_QUALITY_SUBSTITUED	**0x0020**	Substituted
IEC61850_QUALITY_TEST	**0x0010**	Test
IEC61850_QUALITY_OPERATOR_BLOCKED	**0x0008**	Blocked by Operator

enum IEC61850TimeQualityFlags

IEC61850_TIMEQUALITY_LEAP_SECOND_KNOWN	**0x0000000080**
IEC61850_TIMEQUALITY_CLOCK_FAILURE	**0x0000000040**
IEC61850_TIMEQUALITY_CLOCK_NOT_SYNCHRONIZED	**0x0000000020**
IEC61850_TIMEQUALITY_0_BIT_OF_ACCURACY	**0x0000000000**
IEC61850_TIMEQUALITY_1_BIT_OF_ACCURACY	**0x0000000001**
IEC61850_TIMEQUALITY_24_BIT_OF_ACCURACY	**0x0000000018**
IEC61850_TIMEQUALITY_ACCURACY_UNSPECIFIED	**0x000000001F**

7.3 Examples

7.3.1 Server Source

```
/*****************************************************
 This is 61850 Demo Souce Code
The DIP Switch and LED's are used as Inputs and Outputs
Please refer to power point presentation "61850 ICD Editor DK61.pps"

To run

SV61850.EXE <FILENAME.ICD>

e.g DK61.ICD
*****************************************************/

#include <clib.h>
#include <stdio.h>
#include <mem.h>

/* Only SystemCORP Generic is supported in the protcol */
#define SUPPORTED_PROTOCOL            IEC61850_SYSTEMCORP_GENERIC
/* Include IEC 61850 API */
#include "IEC61850API.h"

//#define DEBUG_SV61850 1
/* Maximum Object types */
#define OBJECT_TYPES          2

/* Total Objects in each object type */
#define OBJECTS               8

/* SC143 Input Output Address Location */
#define IO_ADDR               0xC00

/* Input Output Handler Task Priority */
#define IOHANDLER_PRIO        120

/* Input Output Handler Task Stack Size */
#define TASK_STACKSIZE        1024


/* Object Types */
enum
{
DIGITAL_INPUT          = 1,    // Digital Input       (DIP Switch )
   DIGITAL_OUTPUT      = 2,    // Digital Output (LED )
}eObjectTypes;

enum
{
DIGINPUT_INDEX         = 0,    // Digital Input
   DIGOUTPUT_INDEX     = 1,    // Digital Output
}eObjectIndex;

/* Object Information Index */
enum
{
VALUE_INDEX            = 1,    // Value Index
   QUALITY_INDEX       = 2,    // Quality Index
   TIME_STAMP_INDEX    = 3,    // Time Stamp Index
}eObjectInfoIndex;

/* NTP Time Stamp */
typedef  struct        tag_NTPTimeStamp
{
   unsigned long int          uliSeconds;            /* Number of Seconds */
   unsigned long int          uliFractionsOfSecond;  /* Fraction of a second as a binary
                                                                        fraction. */
}tNTPTimeStamp;
```

```
/* Objects */
typedef struct tag_DK61Object
{
unsigned char                       ucObjectNo;                 /* Object Number */
   unsigned char                    ucObjectType;               /* Object Type */
   unsigned char                    ucObjectValue;              /* Object Value */
   unsigned short int               usiObjectQuality;           /* Object Quality */
   tNTPTimeStamp                    tObjectTime;                /* Obect Time */
}tDK61Object;

/* Full Quality */
typedef struct tag_FullQuality
{
   unsigned char                    ucvalidity;                 /* validity  */
   unsigned char                    ucdetailQual;               /* detail quality */
   unsigned char                    ucsource;                   /* source */
   unsigned char                    uctest;                     /* testing ? */
   unsigned char                    ucoperatorBlocked;          /* operator blocked */
}tFullQuality;


/* Local Database */
tDK61Object atObj[OBJECT_TYPES][OBJECTS];


/* IO Handler Task Stack */
static unsigned int IOHandler_stack[TASK_STACKSIZE];

/* IO Handler Function */
void huge IOHandler(void);

/* Read Callback function */
int MyReadFunction(IEC61850_ObjectID * ptObjectID, IEC61850_ObjectData * ptReturnedValue);

/* Write Callback function */
int MyWriteFunction(IEC61850_ObjectID * ptObjectID, const IEC61850_ObjectData * ptNewValue);

/* Create Local Database */
void CreateLocalDatabase(void);

/* Function to convert time to 61850 Time */
void ConvertTo61850Time(tNTPTimeStamp *ptObjectTime);


/* IO Handler Task Definition Block */
static TaskDefBlock  IOHandlerTask =
{
  IOHandler,
  {'I','O','H','L'},                        // a name: 4 chars
  &IOHandler_stack[TASK_STACKSIZE],         // top of stack
  TASK_STACKSIZE*sizeof(int),               // size of stack
  0,                                        // attributes, not supported now
  IOHANDLER_PRIO,                           // priority 20(high) ... 127(low)
  0,                                        // time slice (if any), not supported now
  0,0,0,0                                   // mailbox depth, not supported now
};

/* IEC61850 Server */
IEC61850       myServer                                                 = 0;

/* Main Function */
int main(int argc, char *argv[])
{

IEC61850_Parameters   tServerParam   = {0};                         // Server Parameters
int                   error          = IEC61850_ERROR_NONE;         // Error
   int                taskID         = -1;                          // Task ID
   TimeDate_Structure tTimeDate      = {0};
```

```
   /* Enable Programmable Chip Select for SC143 */
pfe_enable_pcs( 6 );

   /* ICD file as parameter */
   if(argc < 2)
   {
        printf("\r\n Usage : SV61850 <FILENAME.ICD>");
      return 0;
   }

   tTimeDate.yr = 10;
   tTimeDate.mn = 01;
   tTimeDate.dy = 01;
   tTimeDate.hr = 00;
   tTimeDate.mn = 00;
   tTimeDate.sec = 00;

   RTX_Set_TimeDate(&tTimeDate);

   do
   {
      tServerParam.ClientServerFlag   = IEC61850_SERVER;     // This is a Server
      tServerParam.ptReadCallback     = MyReadFunction;      // Assign Read Callback function
      tServerParam.ptWriteCallback    = MyWriteFunction;     // Assign Write Callback function
      tServerParam.ptUpdateCallback   = NULL;                // No Update callback for Server

      printf("\r\n Server Create");

      myServer = IEC61850_Create(&tServerParam,&error);     //Create a server
      if(myServer == NULL)
      {
         printf(" Failed : %i", error);
         break;
      }

      printf("\r\n Server Load SCL");

      error = IEC61850_LoadSCLFile(myServer,argv[1]);       // Load in ICD file
      if(error != IEC61850_ERROR_NONE)
      {
         printf(" Failed : %i",error);
         break;
      }

      printf("\r\n Server Start");
      error = IEC61850_Start(myServer);                     // Starts myServer
      if(error != IEC61850_ERROR_NONE)
      {
         printf(" Failed : %i",error);
         break;
      }
   }while(0);                         /* Dummy while loop to avoid nested If's */

   /* Create Local Database */
   CreateLocalDatabase();

   /* Check for Any Error */
   if(error == IEC61850_ERROR_NONE)
   {
        /* Create and Start IO Handler Task */
     error = RTX_Create_Task(&taskID , &IOHandlerTask);
     if(error != 0)
     {
               printf("\r\n IO Handler task Create Failed : %i", error);
     }

     /* Do not Exit */
     while(1)
     {
               RTX_Sleep_Time(30000);
     }

}
   else
   {
        /* If any errors in Create and Starting the Server */
```

```
      /* Stop the Server */
      printf("\r\n  Server Stop");
      error = IEC61850_Stop(myServer);
      if(error != IEC61850_ERROR_NONE)
      {
         printf(" Failed : %i",error);
      }

      /* Free all Memory */
      printf("\r\n  Server Free");
      IEC61850_Free(myServer);
   }

   return 0;
}

/* Function to Create Local Database */
void CreateLocalDatabase(void)
{
unsigned char          ucObjTypeCnt;
unsigned char          ucObjects;
IEC61850_ObjectData    UpdateValue    = {0};         // Value to send on Change
IEC61850_ObjectID      Object         = {0};         // ID of the Object


   /* For all Object Types */
   for(ucObjTypeCnt    = 0; ucObjTypeCnt < OBJECT_TYPES; ucObjTypeCnt++)
   {
       Object.uiField2 = ucObjTypeCnt + 1;           // Object Type

        /* Each Object within Object Type */
        for(ucObjects  = 0; ucObjects < OBJECTS;  ucObjects++)
      {

        /* Assign Object Number */
        atObj[ucObjTypeCnt][ucObjects].ucObjectNo           = ucObjects + 1;

        /* Assign Object Type DIP Switch : 1 , LED : 2 */
               atObj[ucObjTypeCnt][ucObjects].ucObjectType  = ucObjTypeCnt + 1;
               Object.uiField1                              = ucObjects + 1; // Object Number

         /* Initialise Value to 0 */
               atObj[ucObjTypeCnt][ucObjects].ucObjectValue = 0;
               Object.uiField3              = VALUE_INDEX;
               UpdateValue.pvData           = &atObj[ucObjTypeCnt][ucObjects].ucObjectValue;
               UpdateValue.ucType           = IEC61850_DATATYPE_BOOLEAN;
               UpdateValue.uiBitLength      = 8;

        /* Send Update for Value */
        IEC61850_Update(myServer, &Object, &UpdateValue);

               if(Object.uiField2 == DIGITAL_INPUT)
         {
            /* Initialise Quality */
            atObj[ucObjTypeCnt][ucObjects].usiObjectQuality
                                  = (IEC61850_QUALITY_FAILURE | IEC61850_QUALITY_INVALID |

                               IEC61850_QUALITY_OLDDATA | IEC61850_QUALITY_QUESTIONABLE );

               Object.uiField3              = QUALITY_INDEX;
               UpdateValue.pvData           = &atObj[ucObjTypeCnt][ucObjects].usiObjectQuality;
               UpdateValue.ucType           = IEC61850_DATATYPE_QUALITY;
               UpdateValue.uiBitLength      = IEC61850_QUALITY_BITSIZE;

            /* Send Update for Quality */
                       IEC61850_Update(myServer, &Object, &UpdateValue);

            /* Initialise Time */
               ConvertTo61850Time(&atObj[ucObjTypeCnt][ucObjects].tObjectTime);
               Object.uiField3              = TIME_STAMP_INDEX;
               UpdateValue.pvData           = &atObj[ucObjTypeCnt][ucObjects].tObjectTime;
               UpdateValue.ucType           = IEC61850_DATATYPE_TIMESTAMP;
               UpdateValue.uiBitLength      = IEC61850_TIMESTAMP_BITSIZE;

            /* Send Update for Time Stamp */
                       IEC61850_Update(myServer, &Object, &UpdateValue);
```

```
         }
      }
   }

}

/* Read Callback Function */
int MyReadFunction(IEC61850_ObjectID * ptObjectID, IEC61850_ObjectData * ptReturnedValue)
{
   unsigned char ucObjects            = 0;
   unsigned char ucFound              = 0;

        /* Each Object within Object type */
   for(ucObjects = 0; ucObjects < OBJECTS;  ucObjects++)
   {
      /* Check if the Field matches */
      if((ptObjectID->uiField1        == atObj[DIGINPUT_INDEX][ucObjects].ucObjectNo) &&
         ((ptObjectID->uiField2       == atObj[DIGINPUT_INDEX][ucObjects].ucObjectType)))
      {

        if(ptObjectID->uiField3       == VALUE_INDEX)
         {
            /* Return Value */
            memcpy(ptReturnedValue->pvData,
               &atObj[DIGINPUT_INDEX][ucObjects].ucObjectValue,(ptReturnedValue->uiBitLength/8));
            ucFound = 1;
         }

         if(ucFound) break;


        if(ptObjectID->uiField3 == QUALITY_INDEX)
         {
            /* Return Value */
            memcpy(ptReturnedValue->pvData,
               &atObj[DIGINPUT_INDEX][ucObjects].usiObjectQuality,2);
            ucFound = 1;
         }

         if(ucFound) break;

               if(ptObjectID->uiField3 == TIME_STAMP_INDEX)
         {
            /* Return Value */
              memcpy(ptReturnedValue->pvData,
               &atObj[DIGINPUT_INDEX][ucObjects].tObjectTime,(IEC61850_TIMESTAMP_BITSIZE/8));
            ucFound = 1;
         }

               if(ucFound) break;


      }

        if(ucFound) break;

   }
return (0);
}



/* Write Function */
int MyWriteFunction(IEC61850_ObjectID * ptObjectID, const IEC61850_ObjectData * ptNewValue)
{
//      static unsigned char ucPrevLED = 0;
   unsigned char                      ucLED                = 0;
   unsigned char                      ucObjects            = 0;
   unsigned char                      blLEDChange    = 0;
   unsigned char                      ucFound        = 0;


  /* Each Object within Object type */
   for(ucObjects = 0; ucObjects < OBJECTS;  ucObjects++)
   {
```

```
      /* Check if the Field matches */
      if((ptObjectID->uiField1       == atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectNo) &&
         ((ptObjectID->uiField2      == atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectType)))
      {

        if(ptObjectID->uiField3       == VALUE_INDEX)
          {
            /* Get the Value of the */
            memcpy(&ucLED, ptNewValue->pvData, sizeof(unsigned char));

            /* Check if the LED has changed */
            if(atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue != ucLED)
            {
               /* Set the Value */
               atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue = ucLED;
               blLEDChange = 1;
               ucFound = 1;
            }
          }
      }

      if(ucFound) break;
   }

/* LED Changed */
if(blLEDChange)
   {
        ucLED          = 0;
      /* Get all the values of the LED */
      for(ucObjects    = 0; ucObjects < OBJECTS;  ucObjects++)
      {
        /* Form Byte to Output */
        ucLED   = (ucLED | (atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue << ucObjects));
      }
#ifdef DEBUG_SV61850
      printf("\r\n Write LED : %X ", ucLED);
#endif
      /* Output the LED */
      outportb(IO_ADDR,ucLED);
   }

return(0);
}

/* IO Handler Task */
void huge IOHandler(void)
{
  unsigned char               ucDIPValue            = 0;    // DIP Switch Value
  unsigned char               ucPrevDIPValue        = 0;    // Previous DIP Swithc Value
  unsigned char               nucObjects            = 0;    // Total Objects
        IEC61850_ObjectData   UpdateValue           = {0};  // Value to send on Change
IEC61850_ObjectID             Object                = {0};  // ID of the Object
   unsigned char                        ucObjectVal    = 0;    // Local Object Value
   unsigned short      int   usiQuality             = 0;    // Local Quality
   tNTPTimeStamp                        tNTPTime       = {0};  // Local Time Stamp


   while(1)    // Indefinite Loop
   {
              /* Read DIP Switch Value */
         ucDIPValue = inportb(IO_ADDR);
         /* If previous value does not match current value */
         if(ucPrevDIPValue != ucDIPValue)
         {
#ifdef DEBUG_SV61850
            printf("\r\n DIP Value : %X", ucDIPValue);
#endif

            /* Check which swithc has changed */
            for(nucObjects = 0; nucObjects < OBJECTS; nucObjects++)
            {
               /* Get the Values which have changed */
              ucObjectVal = ((ucDIPValue & (1 << nucObjects)) >> nucObjects);
              if(ucObjectVal != atObj[0][nucObjects].ucObjectValue)
              {
#ifdef DEBUG_SV61850
```

```
                printf("\r\n  %u : %X",  atObj[0][nucObjects].ucObjectNo, ucObjectVal);
#endif
                                /* Common Field to all Index */
                Object.uiField1             = nucObjects + 1;      // Object Number
                Object.uiField2             = DIGITAL_INPUT;       // Object Type

                /* Object Value */
                UpdateValue.pvData          = &ucObjectVal;
                UpdateValue.ucType          = IEC61850_DATATYPE_BOOLEAN;
                UpdateValue.uiBitLength     = 8;

                /* Object Value Index */
                Object.uiField3 = VALUE_INDEX;

                /* Update Value in the database */
                atObj[0][nucObjects].ucObjectValue          = ucObjectVal;

                /* Send Update for Value */
                IEC61850_Update(myServer, &Object, &UpdateValue);

                /* Object Quality  */

             /* No way to determine if DIP Switch failed so */
                usiQuality          = 0;
                UpdateValue.pvData   = &usiQuality;
                UpdateValue.ucType   = IEC61850_DATATYPE_QUALITY;
                                UpdateValue.uiBitLength = IEC61850_QUALITY_BITSIZE;
                Object.uiField3 = QUALITY_INDEX;

                /* Update Quality in the database */
                atObj[0][nucObjects].usiObjectQuality    = usiQuality;

                /* Send Update for Quality */
                IEC61850_Update(myServer, &Object, &UpdateValue);

                /* Send Time */
                /* Convert to 61850 Time */
                ConvertTo61850Time(&tNTPTime);
                UpdateValue.pvData   = &tNTPTime;
                UpdateValue.ucType   = IEC61850_DATATYPE_TIMESTAMP;
                                UpdateValue.uiBitLength = IEC61850_TIMESTAMP_BITSIZE;
                Object.uiField3      = TIME_STAMP_INDEX;

                /* Update Time in the database */
                memcpy(&atObj[0][nucObjects].tObjectTime, &tNTPTime, sizeof(tNTPTimeStamp));

                /* Send Update for Time Stamp */
                IEC61850_Update(myServer, &Object, &UpdateValue);
             }
          }
          ucPrevDIPValue =   ucDIPValue;
       }
       RTX_Sleep_Time(1000);
    }
}

/* Convert to 61850 Time */
void ConvertTo61850Time(tNTPTimeStamp *ptObjectTime)
{
TimeDate_Structure tTimeDate  = {0};
       struct time    tTime   = {0};
struct date    tDate          = {0};
   unsigned long int uliDigit = 0;
   unsigned long int uliMicrosec = 0;

RTX_Get_TimeDate (&tTimeDate);

   tTime.ti_hour      = tTimeDate.hr;
   tTime.ti_min       = tTimeDate.min;
   tTime.ti_sec       = tTimeDate.sec;

   tDate.da_day       = tTimeDate.dy;
   tDate.da_mon       = tTimeDate.mn;
   tDate.da_year      = tTimeDate.yr + 2000;
```

```
   //Load Seconds since 1 Jan 1900
ptObjectTime->uliSeconds = dostounix(&tDate, &tTime) - 18000; // Calculate Number of second

   ptObjectTime->uliFractionsOfSecond = 0;

for(uliDigit = 0x80000000L; uliDigit > 0; uliDigit = uliDigit /2)
                                                    // Loop down throught 2^-1 to 2^-16

   {
      uliMicrosec = uliMicrosec * 2;                // Check for a muilt of fraction
      if(uliMicrosec >= 1000000L)
      {
         ptObjectTime->uliFractionsOfSecond  = ptObjectTime->uliFractionsOfSecond | uliDigit;
                                                    // Set bit to 1
         uliMicrosec = uliMicrosec - 1000000L;      // Remove 1000000
      }
   }

   ptObjectTime->uliFractionsOfSecond = (ptObjectTime->uliFractionsOfSecond & 0xFFFFFF00L) |
IEC61850_TIMEQUALITY_ACCURACY_UNSPECIFIED | IEC61850_TIMEQUALITY_LEAP_SECOND_KNOWN; // Set Time
accurcy and Leap Second Known
}
```

7.3.2 CID File for Server

```
<?xml version="1.0" encoding="UTF-8"?>
<SCL xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.iec.ch/61850/2003/SCL">
   <Header id="" version="3"/>
   <Communication>
      <SubNetwork name="SubNetworkName">
         <ConnectedAP iedName="DK61" apName="SubstationRing1">
            <Address>
               <P type="OSI-AP-Title">1,1,9999,1</P>
               <P type="OSI-AE-Qualifier">12</P>
               <P type="OSI-PSEL">00000001</P>
               <P type="OSI-SSEL">0001</P>
               <P type="OSI-TSEL">0001</P>
               <P type="IP">192.168.1.124</P>
               <P type="IP-SUBNET">255.255.255.0</P>
               <P type="IP-GATEWAY">192.168.1.1</P>
            </Address>
            <GSE ldInst="LDevice1" cbName="GSE_CB_GOOSE">
               <Address>
                  <P type="MAC-Address">01-0C-CD-01-00-00</P>
                  <P type="VLAN-PRIORITY">4</P>
                  <P type="VLAN-ID">0</P>
                  <P type="APPID">0</P>
               </Address>
            </GSE>
         </ConnectedAP>
      </SubNetwork>
   </Communication>
   <IED type="RTUType" manufacturer="SystemCORP Pty Ltd" configVersion="1.0" name="DK61">
      <Services/>
      <AccessPoint name="SubstationRing1">
         <Server timeout="30">
            <Authentication/>
            <LDevice inst="LDevice1" desc="">
               <LN0 lnClass="LLN0" inst="" lnType="LLN0_0">
                  <DataSet name="Indicate_DataSet">
                     <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="1" doName="Ind1"
fc="ST"/>
                     <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="1" doName="Ind2"
fc="ST"/>
                     <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="1" doName="Ind3"
fc="ST"/>
                     <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="1" doName="Ind4"
fc="ST"/>
                  </DataSet>
                  <DataSet name="Goose_Alarm_DataSet">
                     <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm5"
daName="stVal" fc="ST"/>
                     <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm5"
daName="q" fc="ST"/>
```

```
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm5"
daName="t" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm6"
daName="stVal" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm6"
daName="q" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm6"
daName="t" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm7"
daName="stVal" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm7"
daName="q" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm7"
daName="t" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm8"
daName="stVal" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm8"
daName="q" fc="ST"/>
                    <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO" lnInst="2" doName="Alm8"
daName="t" fc="ST"/>
                </DataSet>
                <ReportControl rptID="MyRepURCB_ID" confRev="0" datSet="Indicate_DataSet"
name="UNBUFFERED_RCB" desc="Unbuf RCB">
                    <TrgOps dchg="true" qchg="true" dupd="true"/>
                    <OptFields seqNum="true" timeStamp="true" dataSet="true" reasonCode="true"
entryID="true" configRef="true"/>
                </ReportControl>
                <GSEControl type="GOOSE" appID="GSE_CB_ID" confRev="0" datSet="Goose_Alarm_DataSet"
name="GSE_CB_GOOSE" desc="For GOOSE"/>
            </LN0>
            <LN lnClass="GGIO" inst="1" prefix="DIPS_" lnType="GGIO_0">
                <DOI name="Ind1">
                    <DAI name="stVal">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="1" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="q">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="1" Field3="2" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="t">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="1" Field3="3" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                </DOI>
                <DOI name="Ind2">
                    <DAI name="stVal">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="2" Field2="1" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="q">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="2" Field2="1" Field3="2" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="t">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="2" Field2="1" Field3="3" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                </DOI>
                <DOI name="Ind3">
                    <DAI name="stVal">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="3" Field2="1" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="q">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="3" Field2="1" Field3="2" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
```

```
                    <DAI name="t">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="3" Field2="1" Field3="3" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                 </DOI>
                 <DOI name="Ind4">
                    <DAI name="stVal">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="4" Field2="1" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="q">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="4" Field2="1" Field3="2" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="t">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="4" Field2="1" Field3="3" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                 </DOI>
              </LN>
              <LN lnClass="GGIO" inst="2" prefix="DIPS_" lnType="GGIO_10">
                 <DOI name="Alm5">
                    <DAI name="stVal">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="5" Field2="1" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="q">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="5" Field2="1" Field3="2" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="t">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="5" Field2="1" Field3="3" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                 </DOI>
                 <DOI name="Alm6">
                    <DAI name="stVal">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="6" Field2="1" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="q">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="6" Field2="1" Field3="2" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="t">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="6" Field2="1" Field3="3" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                 </DOI>
                 <DOI name="Alm7">
                    <DAI name="stVal">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="7" Field2="1" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="q">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="7" Field2="1" Field3="2" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                    <DAI name="t">
                       <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="7" Field2="1" Field3="3" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                    </DAI>
                 </DOI>
```

```
                  <DOI name="Alm8">
                     <DAI name="stVal">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="8" Field2="1" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                     </DAI>
                     <DAI name="q">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="8" Field2="1" Field3="2" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                     </DAI>
                     <DAI name="t">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="8" Field2="1" Field3="3" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="GGIO" inst="3" prefix="LEDO_" lnType="GGIO_17">
                  <DOI name="SPCSO1">
                     <SDI name="Oper">
                        <DAI name="ctlVal">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="2" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                        </DAI>
                     </SDI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SPCSO2">
                     <SDI name="Oper">
                        <DAI name="ctlVal">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="2" Field2="2" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                        </DAI>
                     </SDI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SPCSO3">
                     <SDI name="Oper">
                        <DAI name="ctlVal">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="3" Field2="2" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                        </DAI>
                     </SDI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SPCSO4">
                     <SDI name="Oper">
                        <DAI name="ctlVal">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="4" Field2="2" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                        </DAI>
                     </SDI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SPCSO5">
                     <SDI name="Oper">
                        <DAI name="ctlVal">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="5" Field2="2" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                        </DAI>
                     </SDI>
                     <DAI name="ctlModel" valKind="Set">
```

```
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SPCSO6">
                     <SDI name="Oper">
                        <DAI name="ctlVal">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="6" Field2="2" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                        </DAI>
                     </SDI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SPCSO7">
                     <SDI name="Oper">
                        <DAI name="ctlVal">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="7" Field2="2" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                        </DAI>
                     </SDI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SPCSO8">
                     <SDI name="Oper">
                        <DAI name="ctlVal">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="8" Field2="2" Field3="1" Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/></Private>
                        </DAI>
                     </SDI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
               </LN>
            </LDevice>
         </Server>
      </AccessPoint>
   </IED>
   <DataTypeTemplates>
      <LNodeType lnClass="LLN0" id="LLN0_0">
         <DO name="Mod" type="INC_1"/>
         <DO name="Beh" type="INS_2"/>
         <DO name="Health" type="INS_3"/>
         <DO name="NamPlt" type="LPL_0"/>
      </LNodeType>
      <LNodeType lnClass="GGIO" id="GGIO_0">
         <DO name="Mod" type="INC_0"/>
         <DO name="Beh" type="INS_0"/>
         <DO name="Health" type="INS_1"/>
         <DO name="NamPlt" type="LPL_1"/>
         <DO name="Ind1" type="SPS_0"/>
         <DO name="Ind2" type="SPS_0"/>
         <DO name="Ind3" type="SPS_0"/>
         <DO name="Ind4" type="SPS_0"/>
      </LNodeType>
      <LNodeType lnClass="GGIO" id="GGIO_1">
         <DO name="Mod" type="INC_2"/>
         <DO name="Beh" type="INS_4"/>
         <DO name="Health" type="INS_5"/>
         <DO name="NamPlt" type="LPL_2"/>
         <DO name="Alm1" type="SPS_1"/>
         <DO name="Alm2" type="SPS_1"/>
         <DO name="Alm3" type="SPS_1"/>
         <DO name="Alm4" type="SPS_1"/>
      </LNodeType>
      <LNodeType lnClass="GGIO" id="GGIO_2">
         <DO name="Mod" type="INC_2"/>
         <DO name="Beh" type="INS_4"/>
         <DO name="Health" type="INS_5"/>
         <DO name="NamPlt" type="LPL_2"/>
```

```
      <DO name="Alm3" type="SPS_1"/>
      <DO name="Alm4" type="SPS_1"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_3">
      <DO name="Mod" type="INC_2"/>
      <DO name="Beh" type="INS_4"/>
      <DO name="Health" type="INS_5"/>
      <DO name="NamPlt" type="LPL_2"/>
      <DO name="Alm4" type="SPS_1"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_4">
      <DO name="Mod" type="INC_2"/>
      <DO name="Beh" type="INS_4"/>
      <DO name="Health" type="INS_5"/>
      <DO name="NamPlt" type="LPL_2"/>
      <DO name="Alm1" type="SPS_2"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_5">
      <DO name="Mod" type="INC_2"/>
      <DO name="Beh" type="INS_4"/>
      <DO name="Health" type="INS_5"/>
      <DO name="NamPlt" type="LPL_2"/>
      <DO name="Alm1" type="SPS_3"/>
      <DO name="Alm2" type="SPS_3"/>
      <DO name="Alm3" type="SPS_3"/>
      <DO name="Alm4" type="SPS_3"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_6">
      <DO name="Mod" type="INC_2"/>
      <DO name="Beh" type="INS_4"/>
      <DO name="Health" type="INS_5"/>
      <DO name="NamPlt" type="LPL_2"/>
      <DO name="Alm2" type="SPS_3"/>
      <DO name="Alm3" type="SPS_3"/>
      <DO name="Alm4" type="SPS_3"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_7">
      <DO name="Mod" type="INC_2"/>
      <DO name="Beh" type="INS_4"/>
      <DO name="Health" type="INS_5"/>
      <DO name="NamPlt" type="LPL_2"/>
      <DO name="Alm4" type="SPS_3"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_8">
      <DO name="Mod" type="INC_2"/>
      <DO name="Beh" type="INS_4"/>
      <DO name="Health" type="INS_5"/>
      <DO name="NamPlt" type="LPL_2"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_9">
      <DO name="Mod" type="INC_2"/>
      <DO name="Beh" type="INS_4"/>
      <DO name="Health" type="INS_5"/>
      <DO name="NamPlt" type="LPL_2"/>
      <DO name="Alm1" type="SPS_4"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_10">
      <DO name="Mod" type="INC_2"/>
      <DO name="Beh" type="INS_4"/>
      <DO name="Health" type="INS_5"/>
      <DO name="NamPlt" type="LPL_2"/>
      <DO name="Alm5" type="SPS_5"/>
      <DO name="Alm6" type="SPS_5"/>
      <DO name="Alm7" type="SPS_5"/>
      <DO name="Alm8" type="SPS_5"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_11">
      <DO name="Mod" type="INC_3"/>
      <DO name="Beh" type="INS_6"/>
      <DO name="Health" type="INS_7"/>
      <DO name="NamPlt" type="LPL_3"/>
      <DO name="SPCSO2" type="SPC_0"/>
      <DO name="SPCSO3" type="SPC_0"/>
      <DO name="SPCSO4" type="SPC_0"/>
      <DO name="SPCSO5" type="SPC_0"/>
      <DO name="SPCSO6" type="SPC_0"/>
```

```
      <DO name="SPCSO7" type="SPC_0"/>
      <DO name="SPCSO8" type="SPC_0"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_12">
      <DO name="Mod" type="INC_3"/>
      <DO name="Beh" type="INS_6"/>
      <DO name="Health" type="INS_7"/>
      <DO name="NamPlt" type="LPL_3"/>
      <DO name="SPCSO3" type="SPC_0"/>
      <DO name="SPCSO4" type="SPC_0"/>
      <DO name="SPCSO5" type="SPC_0"/>
      <DO name="SPCSO6" type="SPC_0"/>
      <DO name="SPCSO7" type="SPC_0"/>
      <DO name="SPCSO8" type="SPC_0"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_13">
      <DO name="Mod" type="INC_3"/>
      <DO name="Beh" type="INS_6"/>
      <DO name="Health" type="INS_7"/>
      <DO name="NamPlt" type="LPL_3"/>
      <DO name="SPCSO5" type="SPC_0"/>
      <DO name="SPCSO6" type="SPC_0"/>
      <DO name="SPCSO7" type="SPC_0"/>
      <DO name="SPCSO8" type="SPC_0"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_14">
      <DO name="Mod" type="INC_3"/>
      <DO name="Beh" type="INS_6"/>
      <DO name="Health" type="INS_7"/>
      <DO name="NamPlt" type="LPL_3"/>
      <DO name="SPCSO6" type="SPC_0"/>
      <DO name="SPCSO7" type="SPC_0"/>
      <DO name="SPCSO8" type="SPC_0"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_15">
      <DO name="Mod" type="INC_3"/>
      <DO name="Beh" type="INS_6"/>
      <DO name="Health" type="INS_7"/>
      <DO name="NamPlt" type="LPL_3"/>
      <DO name="SPCSO8" type="SPC_0"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_16">
      <DO name="Mod" type="INC_3"/>
      <DO name="Beh" type="INS_6"/>
      <DO name="Health" type="INS_7"/>
      <DO name="NamPlt" type="LPL_3"/>
   </LNodeType>
   <LNodeType lnClass="GGIO" id="GGIO_17">
      <DO name="Mod" type="INC_3"/>
      <DO name="Beh" type="INS_6"/>
      <DO name="Health" type="INS_7"/>
      <DO name="NamPlt" type="LPL_3"/>
      <DO name="SPCSO1" type="SPC_1"/>
      <DO name="SPCSO2" type="SPC_1"/>
      <DO name="SPCSO3" type="SPC_1"/>
      <DO name="SPCSO4" type="SPC_1"/>
      <DO name="SPCSO5" type="SPC_1"/>
      <DO name="SPCSO6" type="SPC_1"/>
      <DO name="SPCSO7" type="SPC_1"/>
      <DO name="SPCSO8" type="SPC_1"/>
   </LNodeType>
   <DOType cdc="INC" id="INC_1">
      <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Mod"/>
      <DA qchg="true" fc="ST" name="q" bType="Quality"/>
      <DA fc="ST" name="t" bType="Timestamp"/>
      <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
   </DOType>
   <DOType cdc="INS" id="INS_2">
      <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Mod"/>
      <DA qchg="true" fc="ST" name="q" bType="Quality"/>
      <DA fc="ST" name="t" bType="Timestamp"/>
   </DOType>
   <DOType cdc="INS" id="INS_3">
      <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Health"/>
      <DA qchg="true" fc="ST" name="q" bType="Quality"/>
      <DA fc="ST" name="t" bType="Timestamp"/>
```

```
</DOType>
<DOType cdc="LPL" id="LPL_0">
   <DA fc="DC" name="vendor" bType="VisString255"/>
   <DA fc="DC" name="swRev" bType="VisString255"/>
   <DA fc="DC" name="d" bType="VisString255"/>
</DOType>
<DOType cdc="INC" id="INC_0" desc="Controllable integer status">
   <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
   <DA fc="CF" name="ctlModel" bType="Enum" type="CtlModels"/>
</DOType>
<DOType cdc="INS" id="INS_0" desc="Integer status">
   <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="INS" id="INS_1" desc="Integer status">
   <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="LPL" id="LPL_1" desc="Logical Node name plate">
   <DA fc="DC" name="vendor" bType="VisString255"/>
   <DA fc="DC" name="swRev" bType="VisString255"/>
   <DA fc="DC" name="d" bType="VisString255"/>
</DOType>
<DOType cdc="SPS" id="SPS_0" desc="Single point status">
   <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="INC" id="INC_2" desc="Controllable integer status">
   <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
   <DA fc="CF" name="ctlModel" bType="Enum" type="CtlModels"/>
</DOType>
<DOType cdc="INS" id="INS_4" desc="Integer status">
   <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="INS" id="INS_5" desc="Integer status">
   <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="LPL" id="LPL_2" desc="Logical Node name plate">
   <DA fc="DC" name="vendor" bType="VisString255"/>
   <DA fc="DC" name="swRev" bType="VisString255"/>
   <DA fc="DC" name="d" bType="VisString255"/>
</DOType>
<DOType cdc="SPS" id="SPS_1" desc="Single point status">
   <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="SPS" id="SPS_2" desc="Single point status">
   <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="SPS" id="SPS_3" desc="Single point status">
   <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="SPS" id="SPS_4" desc="Single point status">
   <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="SPS" id="SPS_5" desc="Single point status">
   <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
```

```
      <DA fc="ST" name="t" bType="Timestamp"/>
    </DOType>
    <DOType cdc="INC" id="INC_3" desc="Controllable integer status">
      <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
      <DA qchg="true" fc="ST" name="q" bType="Quality"/>
      <DA fc="ST" name="t" bType="Timestamp"/>
      <DA fc="CF" name="ctlModel" bType="Enum" type="CtlModels"/>
    </DOType>
    <DOType cdc="INS" id="INS_6" desc="Integer status">
      <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
      <DA qchg="true" fc="ST" name="q" bType="Quality"/>
      <DA fc="ST" name="t" bType="Timestamp"/>
    </DOType>
    <DOType cdc="INS" id="INS_7" desc="Integer status">
      <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
      <DA qchg="true" fc="ST" name="q" bType="Quality"/>
      <DA fc="ST" name="t" bType="Timestamp"/>
    </DOType>
    <DOType cdc="LPL" id="LPL_3" desc="Logical Node name plate">
      <DA fc="DC" name="vendor" bType="VisString255"/>
      <DA fc="DC" name="swRev" bType="VisString255"/>
      <DA fc="DC" name="d" bType="VisString255"/>
    </DOType>
    <DOType cdc="SPC" id="SPC_0" desc="Controllable single point">
      <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
      <DA fc="CF" name="ctlModel" bType="Enum" type="CtlModels"/>
    </DOType>
    <DOType cdc="SPC" id="SPC_1" desc="Controllable single point">
      <DA fc="CO" name="Oper" bType="Struct" type="SPCOperate_0"/>
      <DA fc="CF" name="ctlModel" bType="Enum" type="CtlModels"/>
    </DOType>
    <DAType id="Originator_0">
      <BDA name="orCat" bType="Enum" type="OrCat"/>
      <BDA name="orIdent" bType="Octet64"/>
    </DAType>
    <DAType id="SPCOperate_0">
      <BDA name="ctlVal" bType="BOOLEAN"/>
      <BDA name="origin" bType="Struct" type="Originator_0"/>
      <BDA name="ctlNum" bType="INT8U"/>
      <BDA name="T" bType="Timestamp"/>
      <BDA name="Test" bType="BOOLEAN"/>
      <BDA name="Check" bType="Check"/>
    </DAType>
    <EnumType id="ctlModel">
      <EnumVal ord="0">status-only</EnumVal>
      <EnumVal ord="1">direct-with-normal-security</EnumVal>
      <EnumVal ord="2">sbo-with-normal-security</EnumVal>
      <EnumVal ord="3">direct-with-enhanced-security</EnumVal>
      <EnumVal ord="4">sbo-with-enhanced-security</EnumVal>
    </EnumType>
    <EnumType id="Mod">
      <EnumVal ord="1">on</EnumVal>
      <EnumVal ord="2">blocked</EnumVal>
      <EnumVal ord="3">test</EnumVal>
      <EnumVal ord="4">test/blocked</EnumVal>
      <EnumVal ord="5">off</EnumVal>
    </EnumType>
    <EnumType id="Health">
      <EnumVal ord="1">Ok</EnumVal>
      <EnumVal ord="2">Warning</EnumVal>
      <EnumVal ord="3">Alarm</EnumVal>
    </EnumType>
    <EnumType id="CtlModels">
      <EnumVal ord="0">status-only</EnumVal>
      <EnumVal ord="1">direct-with-normal-security</EnumVal>
      <EnumVal ord="2">sbo-with-normal-security</EnumVal>
      <EnumVal ord="3">direct-with-enhanced-security</EnumVal>
      <EnumVal ord="4">sbo-with-enhanced-security</EnumVal>
    </EnumType>
  </DataTypeTemplates>
    </SCL>
```

7.3.3 Client Source

```
#include "IEC61850API.h"

void main()
{
	// Create Client
	IEC61850 myClient;
	IEC61850_Parameters tClientParam;
	int error;

	tClientParam.ClientServerFlag = IEC61850_CLIENT;
	tClientParam.ptReadCallback = NULL;
	tClientParam.ptWriteCallback = NULL;
	tClientParam.ptUpdateCallback = UpdateFunction;

	myClient = IEC61850_Create(&tClientParam,&error); //Create a client
	if(myClient == NULL)
		{
			printf("Client Failed to create:%i",error);
		}
	error = IEC61850_LoadSCLFile(myClient,"myIDE.scd");
	if(error != IEC61850_ERROR_NONE)
		{
			printf("Loading error has occured: %i",error);
		}
	error = IEC61850_Start(myClient);
	if(error != IEC61850_ERROR_NONE)
		{
			printf("Failed to start client: %i",error);
		}

	// Do something else
	// ...

	// Read in a value
	IEC61850_ObjectData Value;
	IEC61850_ObjectID Object;
	Unsigned32 u32Counter;

	// Load Data
	Value.ucType = IEC61850_DATATYPE_INT32;
	Value.uiBitLength = sizeof(u32Counter)*8;
	Value.pvData = &u32Counter;

	// Load Object ID
	Object.uiNumber = 43;
	error = IEC61850_Read(myClient, &Object, &Value);
	if(error != IEC61850_ERROR_NONE)
		{
			printf("error has occured: %i",error);
		}
	else
		{
			printf("Count = %u",u32Counter);
		}

	// Do something else
	// ...

	u32Counter = u32Counter/2; // Half the value given and write it back
	error = IEC61850_Write(myServer, &Object, &Value);
	if(error != IEC61850_ERROR_NONE)
		{
			printf("Write has failed: %i",error);
		}
	// Do something else
	// ...

	// Shutting down client
	error = IEC61850_Stop(myClient);
	if(error != IEC61850_ERROR_NONE)

		{
```

```
                        printf("Failed to stop client: %i",error);
                }

                // End of program
        IEC61850_Free(myClient);
}


// Update Function

void UpdateFunction(IEC61850_ObjectID * ptObjectID,const IEC61850_ObjectData * ptNewValue)
{
        if(ptNewValue->ucType = IEC61850_DATATYPE_INT32) // I am only interested in 32 bit integers
                {
                        printf("Update on Object %i with value of %i", ptObjectID->
                        uiNumber, *(Unsigned32 *)(ptNewValue->pvData));
                }
```

7.4 Schema for the Private Elements

```
        <?xml version="1.0" encoding="utf-8" ?>
        <xs:schema xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"
        attributeFormDefault="unqualified" finalDefault="extension" elementFormDefault="qualified"
        targetNamespace="http://www.systemcorp.com.au/61850/SCL/Generic" version="1.0"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
                <xs:annotation>
                        <xs:documentation xml:lang="en">
                                COPYRIGHT SystemCORP Pty Ltd, 2009. Version 1.0.Release
                        </xs:documentation>
                </xs:annotation>
                <xs:complexType name="tGenericPrivateObject">
                        <xs:attribute name="Field1" type="xs:unsignedInt" use="required" />
                        <xs:attribute name="Field2" type="xs:unsignedInt" use="required" />
                        <xs:attribute name="Field3" type="xs:unsignedInt" use="required" />
                        <xs:attribute name="Field4" type="xs:unsignedInt" use="required" />
                        <xs:attribute name="Field5" type="xs:unsignedInt" use="required" />
                </xs:complexType>
                <xs:element name="GenericPrivateObject" type="tGenericPrivateObject" />
        </xs:schema>
```