

International Electricity Metering Protocol

COSEM Server Stack

 <p>SANDS[®] Explore Excel Expand ISO 9001:2000 Certified Company</p>	<p>SANDS INSTRUMENTATION PVT. LTD. <i>ISO 9001:2000 Certified Company</i> H.O. : MF-7, CIPET Hostel Road, Thiru. Vi. Ka. Industrial Estate, Ekkaduthangal, Chennai - 600 097. Tel : (044) 2225 3832 / 2225 3833 Fax : (044) 2225 3831 E.mail : sands@sandsindia.com Website : www.sandsindia.com</p>	
--	--	---

Table of Contents

Table of Contents.....	2
Introduction.....	4
Operation.....	4
Implementation.....	5
Module I.....	5
UART_Init.....	5
UART_Tx.....	5
Interrupt Sub Routines.....	5
Module II.....	6
OBIS.h.....	6
IC AllObjects.....	6
IC SAPAssignment.....	6
IC LNAssociation.....	7
Struct xDLMSInfo.....	7
IC DataClass.....	7
IC RegisterClass.....	7
ICExtendedRegisterClass.....	8
ICDemandRegisterClass.....	8
ICProfileClass.....	8
IClock.....	8
SNRM and UA.....	9
AARQ and AARE	9
GET_Request and GET_Response.....	10
SET_Request and SET_Response.....	10

Action_Request and Action_Response..... 10

DISC..... 10

Module III..... 11

Introduction

The *DLMS Server Stack* is an implementation of the *International Standard IEC-62056* defined *COSEM Protocol* which makes the *Electricity Meters DLMS compliant*. The *three layer HDLC Protocol* is implemented in the *Direct HDLC Mode for Application Contexts Short Name and Logical Name Referencing*. The *Physical Interface* used is *RS-232*. The Stack is implemented in the *programming language C*.

DLMS is an *Application Layer Specification* and is independent of the lower layers and thus the mode of communication. The objective of DLMS is to provide an inter-operable environment for *Structured Modelling* and *Meter Data Exchange*. COSEM includes an evolution of DLMS providing a more metering specific view of the meter through the *COSEM Interface Objects* and *Interface Classes*. The current *DLMS Version* of *xDLMS* is 6.

DLMS/COSEM is an *Object Model* to view the functionality of the meter, as in it seen at its *communicating interface(s)*, a *Messaging Method* – based on *ASN.1 Encoding* – to communicate with the model and to turn the data to a series of bytes which a *Transportation Method* carries this information between the *Metering Equipment* and the *Data Collection System*.

Operation

The project can be segmented into the following major parts:

- 1) The first part involves the *initialisation* and *configuration* of the *UART*. The *UART* should be configured for the same *Baud Rate* as that of the Client (*Meter Reading Instrument* or *Data Collection System*).
- 2) The second part includes the *DLMS Application*, where the Meter is *Modelled* and all the *information held by the device is mapped to the model*. Both the model and the mapping is standardised and is completely independent of the communication method used.
- 3) The third section includes *Integration of the Stack with the Meter*. The data requested by a client can be either *static* or *dynamic* by nature. As the name suggests, the *static data* is the one which *does not change with time* while the *dynamic data* is a *time dependent variable*. All the static data associated with the meter is stored in the Stack.

When the client requests a static data, the Stack entertains the request. But if the information requested by the client is dynamic in nature, the Stack makes a call to the

meter to fetch the data. It is this data exchange the third section entails to.

The *Module I* and *III* entirely *meter specific* and the Stack will make calls to these function as need arises. The *Module II* involves the implementation of the *DLMS Model*. As a prerequisite, the following information is required from the Meter:

- 1) The available COSEM Object instances at the meter side. This information is supplemented by their *Logical Names*, *Short Names*, *Interface Classes*, and *Access Rights* etc.
- 2) The meter should also specify the set of COSEM Objects which can be accessed by the client, defined in terms of *Association Views*. An Association View defines the *perspective* of the *Server* (the Meter) available to the *Client* (the Data Collection System).
- 3) The *static information* associated with these objects if any.
- 4) The configuration details, such as the *maximum size of the data* that can be *received and transmitted* at once etc.

Implementation

The implementation of the above mentioned model is now described in detail.

Module I

The implementation of Module I – the Physical Layer – requires the following three functions:

UART_Init

This function *initialises* the *UART Module* of the Meter. The UART is configured for a baud rate known to the client prior and for *8 bits Word Length*, *No Parity* and *1 Stop Bit*. The meter should generate an *interrupt* on *reception* of a *character*, which is stored in a buffer to be analysed by the DLMS Model.

UART_Tx

This function takes in a character as input and *transmits* it via *RS-232* to the client.

Interrupt Sub Routines

In addition to the above two functions, the Meter should have necessary *Interrupt Sub Routines* to store the data received from the client through RS-232.

Module II

The DLMS Stack was developed from the following references:

- 1) IEC 62056-21 Direct Local Data Exchange
- 2) IEC 62056-46 Data Link Layer using HDLC Protocol
- 3) IEC 62056-53 COSEM Application Layer
- 4) IEC 62056-61 Object Identification System
- 5) IEC 62056-62 Interface Classes
- 6) IEC 13239:2000 Telecommunications and Information Exchange between Systems – HDLC Procedures

OBIS.h

The header file OBIS.h includes various interface classes (IC) – as defined in IEC 62056-62. These include the list of all meter data objects, the xDLMS services and the Association Objects.

IC AllObjects

All the available COSEM Objects at the meter are recorded here. The objects are defined by the following characteristics – *Logical Name* (the *OBIS Code*), *Short Name* (in case SN Referencing is used), *Interface Class (IC)* the object belongs to, the *Version of the IC*. These entries are as specified by the DLMS Model. Additionally, two other variables are associated, which are the number of attributes linked to the object and the index of the buffer which stores the static information of the object concerned.

IC SAPAssignment

Each *Physical Device* – the *Meter* – may contain one or more *Logical Devices* – a set of *COSEM Objects*. These *Logical Devices* are recognised by their *Service Access Point Addresses*. The *Interface Class SAP Assignment* contains the information on the *assignment* of the *logical devices* to their *SAPs*.

IC LNAssociation

COSEM Logical Devices able to establish application associations using *LN Referencing* (in case of *SN Referencing*, this will be subsequently modified); model the associations through instances of this class. A logical device has one instance of the class for each association the device is able to support.

This interface class *identifies* the *SAP* of the *Server* and *Client* in the association (also termed *Associated Partners*), *xDLMS Information* (as described below), *Access Rights* – which define whether or not the client is entitled to the requested information – and other relevant information such as *Authentication Value* (*Secret Password* to validate the client and/or server), *Application Context Name* (reference to the Application Context) etc. The *Status* of the *Association* can be obtained from this interface class.

Struct xDLMSInfo

The services provided by the *xDLMS Element* – eXtended Device Language Message Specification – are defined here. The *xDLMS Information* is characterised by the *Conformance Block* (as defined in IEC 62056-53) which defines which *data exchange services* (say *GET*, *SET* etc for *LN Referencing*) are supported by the server, the *maximum size of data* during *reception* and *transmission* and the *Quality of Service* etc.

During the *Association Establishment*, these *parameters* are *negotiated*. They are accordingly modified once the negotiation is concluded.

IC DataClass

A *Data Object* stores data related to *internal meter object(s)*; is used to *store* the *configuration data* and *parameters*. The data contained in the class is identified by the attribute *Logical Name* using the *OBIS Identification System* (as defined in IEC 62056-61). The data type of this value is instance specific and is stored in a variable *Data Type*, provided by the meter manufacturer. The data types are encoded as per the *ASN.1 Encoding* using *BER*, as defined in IEC 62056-53.

The *Static Data* (as explained above) associated with a Data Class is referenced using a variable *Data Index*. This variable defines the location where the static information of the object is stored.

IC RegisterClass

A *Register Object* stores a *Process Value* or a *Status Value* with its *Associated Unit*. The nature of the *Register Class* is same as that of the *Data Class* with the inclusion of the *Scaler Unit*. The *Scaler Unit* defines the *unit* and the *scaler* of the value. The *encoded value* of various *units* is as

specified in IEC 62056-62. As with the case of Data Class, all the static information is stored in a buffer, which is referenced by the variable *DataIndex*.

Using the above model, the server entertains the requests from the client. The following are the requests and the responses with which the server replies the client.

ICExtendedRegisterClass

Instances of *Extended Register Class* store a *Process Value* with its *Associated Status*, *Unit* and *Time Stamp*. The process value and unit are same as described in the *ICRegisterClass*. The *Status* provides a status information specific to the *Extended Register*. The *Capture Time* notifies when the value of the attribute value has been captured.

ICDemandRegisterClass

Instances of a *Demand Register* store a demand value with the addition of all the parameters in the *ICXRegisterClass*. The Demand Register delivers two types of demand, the *Current Average Value* and the *Last Average Value*. The former offers the average of the current value (running demand) of the energy accumulated over a time; the latter indicates the average value of the accumulated energy over the entire time period, excluding the current period.

The duration of a period and the number of periods are indicated by the attributes *Period* and *NumberOfPeriods* respectively.

ICProfileClass

The profile generic class is used to store dynamic process values of capture objects – register, clock or a profile. The capture objects can be collected either periodically or when a certain external event occurs. These values can be sorted by either their time stamp or magnitude.

ICClock

An instance of the Clock Interface Class provides information linked to date and time. This information includes the *Time Zone*, *Deviation from the GMT*, the *Status* of the *Clock* and its *Source*.

SNRM and UA

The SNRM Command – *Set Normal Response Mode* – is used to bring the server into the Normal Response Mode. The SNRM is used to negotiate parameters such as the maximum length of the data that can be received and transmitted by either party. On reception of an SNRM, the server examines the values of these parameter (if sent by the client) and chooses a value which both the client and server will be compliant to. This information is conveyed by the server to the client using a UA Frame. The following parameters are negotiated in SNRM:

- 1) Maximum Information Field Length, transmit
- 2) Maximum Information Field Length, receive
- 3) Window size, transmit and
- 4) Window size, receive.

AARQ and AARE

The AARQ – *Application Association Request* – is sent by the client to establish an association with the server. The AARE Frame – *Application Association Response* – is the response sent by the server to an AARQ. The AARQ PDU (Protocol Data Unit) communicates the Application Context used and the Authentication Value (Secret Password to authenticate Client and/or Server) if any to the Server.

The AARQ is used to negotiate the following parameters:

- 1) The Protocol Version,
- 2) Application Context Name,
- 3) Authentication Requirements (if authentication is used),
- 4) Association or xDLMS Information, which involves the following:
 - a. Dedicated Key,
 - b. Quality of Service,
 - c. Proposed DLMS Version,
 - d. Proposed DLMS Conformance Block and
 - e. Maximum Receive APDU Size.

When the parameters thus obtained comply with those at the server, the Server responds with a favourable AARE and it modifies the xDLMS Info in accordance to the negotiated parameters. In

case of any error, the AARE will carry the error message. The SNRM and the AARQ together *negotiate* the entire *xDLMS Information Block*.

GET_Request and GET_Response

A *GET request* is used by the *client to fetch data from the meter*. On reception of such a request, the server application layer first examines the *Access Rights* of the client with respect to the object and the attribute requested. If the client has the right to the information, the server application software builds the *Application Frame* with *requested data* and its *OBIS code, IC number, version* etc obtained from the file *OBIS.h* and sends it to the client.

In case of an error, a corresponding *error APDU* (Application Protocol Data Unit) is developed and sent to the client.

SET_Request and SET_Response

The *SET Request* is used by the client to *set a data at the meter*. The procedure as mentioned for the GET Request is followed in analysing the SET Request. Depending on the result of the exercise, a frame is generated by the Server Application and sent to the client.

Action_Request and Action_Response

The GET and SET requests are used by the client to fetch and modify the COSEM Object attribute respectively. The *Action Request* is used to execute the *Methods of the COSEM Object*. On reception of this request, the server analyses the access rights of the client. If the client is entitled to carry out the method, the server makes a call to the corresponding function.

In case the client is not permitted to execute the action, the server simply sends an APDU to the client, bearing the reason of the failure.

DISC

The *DISC – Disconnect* – command is used to disconnect the logical link layer of the client from the server. On reception of a DISC command, the server responds with a *UA*, similar to the one sent on response to the SNRM. The server is said to be in *NDM – Normal Disconnected Mode* – on reception of this command. *No data exchange* can be carried out in this state, except *Unnumbered Information* and *Mode Setting Command* such as the *SNRM*. The reception of

SNRM will bring the server out of the *NDM* and place it in NRM – *Normal Response Mode*.

Module III

The Module III involves *integration* of the *DLMS Stack* with the *Meter*. As described above, the data requested by the client can be either dynamic or static. The Module III caters to the needs of the client requesting dynamic data. Implementation of this module is purely manufacturer specific. It will take the ID of the object as its input and output the value of the object requested.