

Overview

This document describes the structure of the DLMS solutions provided by Kalki with associated implementation notes. This document also serves as a step-by-step sequence of procedures to be followed by a Meter-Manufacturer, in effect a fully detailed roadmap for implementation of DLMS on their target meters and the subsequent Conformance testing for certification by the DLMS-UA.

About Us

Kalki Communication Technologies (www.kalkitech.com) is the leading provider of DLMS Source Code Libraries in Client and Server editions, enabling Meter Manufacturers to easily integrate a DLMS protocol communication solution to their meter models. The client editions of the DLMS SCL provide the foundation to create Meter-reading and Parameterization tools as well as scalable Data-collection gateways. Client and Server editions may even be combined to create Master-Slave hierarchies of Meters with a designated Master meter acting as a gateway to a collection of slave meters.

The DLMS server solution is delivered as Source Code which means that the Meter manufacturer is intended to integrate the stack into the source code of their meter and build the complete solution onto their target platforms. It may be noted that Kalki also provides the necessary integration services to optimize the time/cost for this activity.

SCL Structure

The SCL, as supplied, may be divided into two sections, the first being the basic DLMS protocol stack (consisting of a COSEM application layer on a HDLC/Mode-E-Bridge Datalink layer) and the second being a set of three interfaces which can be used to plug-in the SCL to a specific target system

Basic Stack
Editable Interfaces

Data Interface (data.c and data.h)	COSEM Appl Layer	Configuration Interface (OBIS.h and config.h)
	HDLC Datalink Layer (With Mode E Switch)	
	Physical Layer	
	Platform Interface (plaf.c and plaf.h)	

Users of the SCL will rarely have to modify the basic stack. The implementation procedure envisages the user editing the three interfaces to fully integrate the solution to the target. The three interfaces are

1) Configuration Interface

Configure the master object-list, logical devices and associations, association object-lists with access-rights, profile object capture-object lists etc. Also configure buffer-sizes, APDU size and Window-size depending upon available RAM

2) Platform Interface

Implement the designated functions to read/write through the communication port, handle modem init/close etc. Also implement Interval timer functions using target-platform timer modules.

3) Data Interface

Implement the data-interface functions to get/set meter data when requested by the DLMS stack. It is to be noted that a vast amount of data requested by a client can be responded by the SCL without calling the data interface functions. These data attributes are termed as 'static' attributes. For example a request to read the "Scaler & Unit" attribute of an Energy Register can be responded to by the SCL from the static configuration in the Configuration interface. However a request to read the Energy value will cause a Get Function in this interface to be called and the user must implement this function to return the requested value

The SCL is ANSI-C compliant and will compile on most hardware platforms. The Meter-Manufacturer must check on the availability of a C-Cross-Compiler and preferably a suitable IDE for building/debugging the application on their selected hardware

DLMS Implementation Roadmap

1 DLMS-UA Membership

It is necessary for the Meter-Manufacturer to take a DLMS-UA membership before any meter-models can be conformance tested and certified by the UA. This is mandated by the DLMS-UA as per current guidelines.

2 Register 3-Letter Manufacturer code with Flag Association

The Meter-Manufacturer can also register a unique 3-letter manufacturer code with the Flag Association. This code will be prefixed before the Logical Device Name of the meter logical device. The logical device name will be used as a unique name for the specific meter-model and can be read from the meter by a DLMS client

3 Meter Data Mapping to DLMS

Kalki will provide consultancy and documentation regarding the standard Interface Classes defined in IEC-62056-62. This document specifies the structure of several object classes that can be used to model Meter data-elements. While several of the object classes are straight-forward, some complex objects, especially related to Tariffication schedules may require a deeper understanding and mapping to the specific procedure followed in the OEM Meter.

4 Specify Meter's DLMS Data capabilities

At this stage, the Meter-manufacturer can obtain from Kalki a detailed meter capability checklist (MCC) which must be filled in to decide on the objects supported by the target meter. The completed checklist will be used to define the full OBIS code list of the meter. The OBIS code is a 6 character code that standardizes the identification of each and every Data element that may be served by an Energy Meter. The DLMS-UA maintains a comprehensive list of standard OBIS codes. The checklist will identify this full and exhaustive list of objects, however several objects may not be supported by the target meter and will need to be edited out. In addition, the meter manufacturer may choose to add Manufacturer-specific OBIS codes to provide DLMS support for special objects that are not available in the standard list maintained by the DLMS-UA. In this manner the Meter-Manufacturer can identify and map all the data in the target meter to standard DLMS OBIS codes.

The checklist will also define the number of different associations that are to be supported by the Meter as well as each Association's object list. The association object-list is an "Association-View" of the meter and defines which objects of the meter are to be exposed to a DLMS client for each particular association and with what access-privileges. Each association object-list is a subset of the Master object list. For example a meter-manufacturer may choose to have a "Meter-Reader" Association which will have an object list comprising of "Read-Only" privileges to a set of Billing Data objects. Another association named "Factory-Settings" may be created which will allow "Read-Write" access to the basic parameterization objects. Yet another Association named

“Utility-Settings” may be created with “Read-Write” access for all the configuration objects like Device Address, Activity Calendar settings etc.

The checklist will also define which attributes of each object will be static (can be configured in the SCL and automatically responded to the client without querying the meter) and which will be dynamic (needing to be read dynamically from the meter)

5 Decide on DLMS Protocol capabilities

Kalki will also provide a DLMS Protocol Capability checklist (PCC) which can be used to determine the protocol features that are intended to be supported by the target meter. Here the user can determine which referencing method to support (Logical name (LN) referencing OR Short name (SN) referencing or both), which bits of the DLMS conformance block are to be supported (subject to the available conformance support in the Kalki SCL), the maximum APDU size, window size and inter-protocol-layer data buffer sizes etc.

There are various other capabilities checklisted including which DLMS interface classes are required etc. All these information will be used to optimize the RAM/ROM size of the final build using conditional compilation.

6 Decide on target hardware for Meter/DLMS MCU

At this stage, Kalki will use the information from the MCC and the PCC to build a sample implementation on Kalki's test-bed MCU in order to provide a reasonable estimate of the RAM/ROM sizes required in the target hardware. This can be used to identify a broad range of hardware platforms for evaluation

This is a crucial stage in the implementation roadmap since at this stage the meter-manufacturer can elect to make trade-off decisions between required functionality support and available RAM/ROM resources. If the preferred target MCU cannot support the required level of resources, it may be necessary to either evaluate other hardware solutions or to go back to step 3 and revise the MCC and PCC. This may require a few iterations of steps 3,4 and 5 before finalization.

The Meter-Manufacturer must also ensure availability of build tools (C-compiler/debugger etc.) for the selected hardware platforms.

However the platform cannot be fully finalized before the next step, where the capabilities of the platform to support the DLMS implementation are described

7 Platform and Data Interface Design

At this stage, it is necessary to evaluate the facilities provided by the target platform for communication ports and hardware timers.

The DLMS stack can be used to operate through any communication channel that is available on the platform. The SCL interacts with the platform communication channel through two data buffers with configurable sizes. The “ReceiveBuffer” is implemented as

a simple and efficient circular buffer. The user of the SCL is required to provide code to fill this buffer from the communication channel selected. A simple implementation of this code would be to tie in the receive interrupt handler to copy the data from the port to the "Receive Buffer". The nature of the communication port is hidden from the protocol stack. It is the task of the user of the SCL to write code to handle RS232, PSTN/GSM Modem, Optical port etc and pipe the data to this buffer. Similarly for transmit, the SCL will call a transmit function and provide the data buffer pointer and length of data. The user of the SCL must write code in this transmit function to actually transmit the data through the port.

The SCL also calls functions to initialize, open and close the port as and when required. The user can implement these functions to perform the stated activity. For example, if the platform communication channel is a modem, the open port function can be implemented to perform modem initialization etc.

The SCL also requires a millisecond timer to be provided in the hardware platform. The SCL will call functions to read the timer value periodically where the user must return the value as well as a flag indicating rollover of the timer when it occurs. The SCL will use this function to time various intervals in the protocol operation.

As regards the Data interface, the user must implement three functions, namely

- **Get Function**
This function will be called by the SCL when it receives a request to read a data attribute that is dynamic (for example an Energy value). The arguments passed to the function will identify the OPBIS code, Interface Class and attribute index of the requested data.
The user must implement this function to return the requested value to the SCL, which will then encode it suitable in DLMS and return to the client.
- **Set Function**
This function will be called by the SCL when it receives a request to set the value of a data attribute in the meter. The arguments will be the same as above (OBIS code, IC number, Attribute index) with an additional data attribute which contains the data passed from the client
The user must implement this function to write the supplied data to the identified attribute
- **Execute Function**
This function will be called when the SCL receives a request to execute a method of any Interface class object. The arguments are the same as above except that instead of the attribute index, it will supply a method index. As before any argument data to be passed to the requested method will also be provided as an argument value
The user must implement this function to execute the specified method

At this stage, the user must evaluate the hardware platform and implementation topology to ensure that these actions can be implemented. In cases where the DLMS SCL is built as a task on the main Meter MCU, the implementation of the Data interface functions will

simply be a memory read/write. However in cases where the DLMS SCL is built on a separate Communications MCU, it will be necessary to write code in the above three functions to communicate to the main Meter MCU to read/write the data.

At the end of this stage, Kalki and the Meter Manufacturer will have a specific set of features (as stated in the finalized MCC and PCC) as well as a specific hardware solution

8 Build Kalki DLMS SCL on target platform

The first build of the SCL for the specific target will be done directly, before integrating (modifying) the three interfaces. This build is intended to set a reference point in the implementation process and ensure that errors are not caused by the basic stack. It is suggested to enable all warnings in the compiler flags to ensure a fully satisfactory build of the SCL.

The user can now proceed to start with the first of the three interfaces namely, perform the Configuration interface editing.

9 Configuration Interface

The configuration interface is simply a pair of C header files “obis.h and config.h”. These header files will contain a default and full configuration for a typical meter and needs to be edited by the user to specify the object and functionality supported by the target meter. This task will be a straight-forward mapping from the previously generated MCC and PCC checklists.

- The primary task for the user here is to edit the master OBIS list which lists all the DLMS objects supported by the meter.
- The next task is to configure the static information for all the objects. Static information is the information which does not change in the meter's operating lifetime (for example a register object's scaling factor and unit).
- Next the user must configure the Logical devices and associations to be supported by the meter. Each association represents a specific “view” of the meter. The user must also configure Association object-lists with the access-rights for each attribute and method of each object in the list. (The Association object-list is a subset of the master object-list, specifying which objects are to be exposed for that particular association)
- The capture-objects list for each configured Profile object should be filled in the relevant table.
- In config.h, the user can selectively turn on/off support for various conformance-block functions as well as set the previously decided APDU size, Info-field size, Window-size and other buffer sizes

After completing the configuration edit, the SCL will be built again to eliminate errors in this stage.

10 Platform Interface

The Platform interface spins out functions used by the protocol driver that will necessarily have to be platform-specific. For example this encompasses the lowest physical layer functions (serial ports, modem handling etc.) as well as hardware-timers. This interface is provided as empty functions with clear documentation regarding the function to be performed by each method, the structure and meaning of the arguments passed and the structure and meaning of the return variable. Each function in this interface is to be implemented to perform a specific task.

At the end of this stage, it will be possible to test the implementation so far (especially communication related functionality) with the simple DLMS client executable provided by Kalki. The client program can be used to communicate with the meter, establish associations and even read data (restricted to those data elements which are statically configured and do not require the Data interface – which will only be implemented in the next step). Various tests can be performed to test the HDLC Data link layer as well as the COSEM Application layer functions.

11 Data Interface

The Data-interface consists of a set of empty functions that are called by the SCL to request/pass data from/to the meter. Each DLMS Get/Read request received at the communications channel can result in this interface function being called to request the dynamic data from the meter.

It is to be noted that a vast amount of data requested by a DLMS client may be responded to automatically by the SCL based upon the static information stored in the Configuration Interface. Only requests that require dynamic data from the meter will result in a call to the functions in this interface.

Similarly each DLMS Set/Write request or Action/UnconfirmedWrite method execution will result in a call to a specific function in this interface.

These functions are left to the Meter-manufacturer to implement to actually get/set the data from/to the meter. In case the DLMS solution is being built as a task in the main Meter board, this implementation may be a simple memory read/write. In cases where the DLMS solution is built on a separate board (housed in the meter enclosure), this may require communication to the main meter board via whatever interfaces are available

At this stage, the DLMS implementation can be fully tested for data validity and performance stability etc.

12 Conformance Testing

Kalki will provide consultancy on setting the PICS/PIXIT files for configuring the Conformance Test Tool for the specific implementation.

The Meter-manufacturer can decide on one of the following two options to Conformance test and certify the meter.

- 1) The Meter manufacturer can purchase the Conformance Testing Tool license for testing and certifying their own meter models

- 2) The Meter-Manufacturer can enter a commercial agreement with an authorized DLMS Third-Party Conformance Testing Center to test their meter models

In either case, after performing the test, the Meter-Manufacturer (or the 3rd-Party testing center) will have to send the test report (generated by the tool) to the DLMS-UA to obtain the Conformance Certificate.