

DLMS Overview

DLMS Metering Communication Protocol, standardized under the IEC-62056 series, comprises mainly of the following standard specifications

- IEC62056-53: COSEM Application Layer
COSEM (Companion Specification for Energy Metering) defines an Application Layer protocol that can handle Meter data and perform the basic functions of data set/get/action operations in the Meter. In addition to these, COSEM handles access rights, client-server connection handling, abstracting meter data from/to COSEM class instances, framing data into COSEM packets, high-level segmentation of data into blocks etc.
- IEC62056-46: HDLC Datalink Layer
HDLC defines a standard Datalink layer performing the functions of low-level addressing, data integrity checks, data sequencing, segmentation and assimilation, link-level handshaking, data-flow control etc. The Datalink layer's main function is to reliably transport COSEM data packets between the client and the server
- IEC62056-42: Physical Layer
This standard specifies the physical layer services for data communication
- IEC62056-21: Direct Local Data Exchange
Alternate ASCII-based protocol stack. A new mode (MODE E) in this specification allows the client to negotiate a switchover to COSEM/HDLC (62056-53&46) following which all communication will use the COSEM/HDLC protocol stack.
- IEC62056-61: OBIS
OBject Identification System defines a standard list of Meter data object identifiers in the form of a 6 character code for each object. This list is maintained by the DLMS-UA
- IEC62056-62: Interface Classes
This specification defines the standard Interface Classes which can be used to represent all possible kinds of Meter Data. Using this specification meter data is abstracted into high-level objects, which can then be operated upon by the protocol stack

DLMS Solutions from Kalki

- **Server Source Code Library (SCL)**

Kalki provides an easy upgrade path for Meter OEMs to implement DLMS protocol in their existing/new meters by providing a DLMS Server Source Code Library (ANSI C) containing the full protocol stack with three simple interfaces to hook-in the meter data, configuration and hardware-platform-specific features.

The delivery-model for this product is chosen to be Source Code to handle the variety of target hardware platforms and to provide maximum flexibility to adapt required features of the protocol, while eliminating un-necessary features. The stack has been written, keeping in mind the hardware-resource constraints typically found in metering hardware platforms and provides flexibility to trade-off between ROM and RAM storage for several large-footprint items (like configuration information, object OBIS lists etc.). In addition to this, to enhance the reliability of the stack, the entire stack runs on statically allocated memory and does not require any dynamic allocation. The static allocation of memory buffers used at various layers and levels can be tuned by editing a few macros to suit the target resource-availability.

The ROM size for a full stack, using all features is found to be about **16kB** (without meter configuration information) and runs on under **4kB** RAM.. Configuration info for a typical meter with about 150 objects, a few Associations and profile capture-object lists was seen to occupy an additional **10kB** (ROM or RAM)

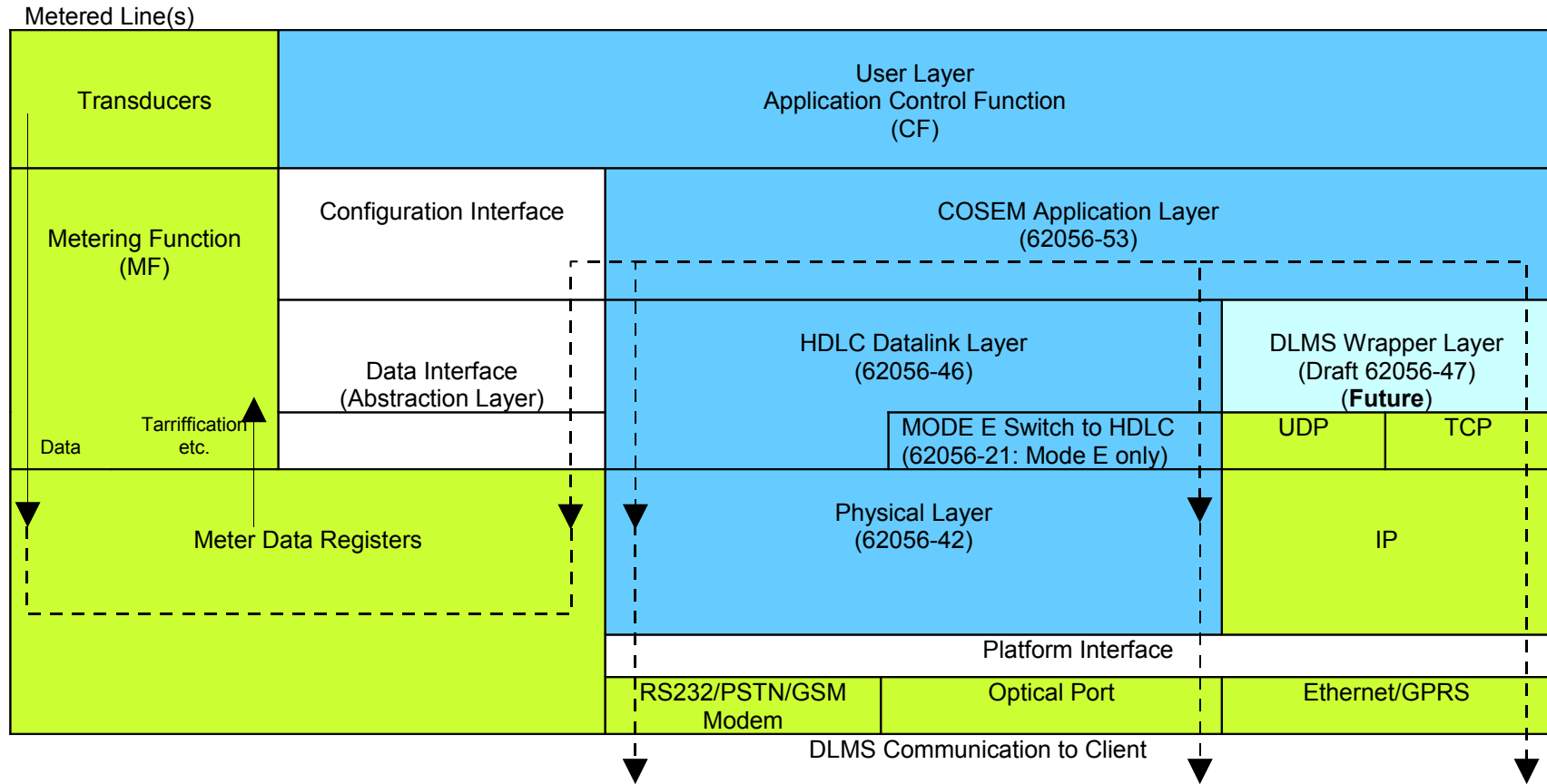
- **Client Source Code Library**

Kalki also provides a DLMS Client protocol stack. This product enables interested OEMs in creating Meter Data collection tools and Parameterization tools on proprietary platforms such as Hand-Held Units etc.

- **Client and Server Object Libraries for Windows/Linux**

Both Client and Server DLMS stacks are also available as binary libraries with a well-defined API for an OEM to implement DLMS Clients and Servers in Windows and Linux environments. This could be used to build Gateways/Data Concentrators/Store and Forward devices etc.

Functional Block Diagram of DLMS Source Code Library (Server edition for Meters)



Legend

Code	Description	Scope
	DLMS Protocol Stack:	Kalki
	Interfaces: Provided by Kalki, requires simple modification to integrate to OEM hardware	Kalki & OEM
	Future addition to DLMS stack communication options (Requires OEM TCP/IP support	Kalki - Future
	Basic Meter Function, Data and platform-specific communication options	Meter OEM

DLMS Compliance and Feature Support

COSEM Application Layer

Application Contexts: The protocol stack supports both Short-Name(SN) and Logical-Name(LN) referencing with no ciphering

Authentication Mechanisms: Supports NO_SECURITY and LOW_LEVEL_SECURITY mechanisms. HIGH_LEVEL_SECURITY mechanisms can be built-in on request.

Conformance block: The stack supports the following featured services in the DLMS Conformance-block

LN Services

- GET
- SET
- ACTION
- ATTRIBUTE 0 WITH GET
- BLOCK TRANSFER WITH GET
- BLOCK TRANSFER WITH SET
- BLOCK TRANSFER WITH ACTION

SN Services

- READ
- WRITE

Further advanced feature-bits in the conformance block in progress for an upcoming release

- ATTRIBUTE 0 WITH SET
- MULTIPLE REFERENCES
- SELECTIVE ACCESS
- PARAMETRIZED ACCESS

Interface Classes: The stack supports the following Interface Classes and their instantiation

- Data (IC: 1)
- Register (IC: 3)
- Extended Register (IC: 4)
- Demand Register (IC: 5)
- Profile Generic (IC: 7)
- Clock (IC: 8)
- Script Table (IC: 9)
- Special Days Table (IC: 11)
- Association SN (IC: 12)
- Association LN (IC: 15)

- SAP Assignment (IC: 17)
- IEC Local Port Setup (IC: 19)
- Activity Calendar (IC: 20)
- Single Action Schedule (IC: 22)
- IEC HDLC Setup (IC: 23)
- PSTN Modem Configuration (IC: 27)
- PSTN Auto Answer (IC: 28)

OBIS Codes: The complete range of standard OBIS Codes related to the above-mentioned Interface Classes can be configured into the system.

HDLC Datalink Layer

Addressing: Supports 1-byte, 2-byte and 4-byte addressing

Timeouts: Supports Inactivity, Inter-frame and Response timeouts

Configuration Interface

The DLMS SCL provides a rich configuration interface with most configuration-lists being statically-initialized arrays that can be located to ROM/RAM depending upon the resource constraints.

Implementers of the SCL can configure

- Logical Devices
- Associations
 - Multiple associations may be configured with different Application Contexts, Authentication Mechanisms and Conformance block services
- Complete OBIS and/or SN list of supported objects
- Object Lists for Associations
 - Object lists that can be accessed under each Association along with the access-privileges for each attribute and method of the listed objects
- Static Information for each object
 - The SCL divides the data of each object into a dynamic and a static part. Static information can easily be configured at build-time
- Capture Object lists for Profile Generic objects

The capture-objects list for each Profile Generic object can be statically created

- **Buffer Sizes**
Configure the sizes of different data structures by modifying the maximum APDU size, maximum Info-field length and Window-size parameters to suit the available resources
- **LN/SN support**
Implementer can choose to support LN or SN or both. Specifying this can help optimize the ROM size of the executable through conditional compilation.
- **Interface Classes support**
Implementer can selectively turn on/off support for each IC. This will help reduce ROM size by conditionally compiling in support only for objects that are needed

Data Interface

The Data interface to the Meter consists of a set of functions that are called by the SCL on receipt of DLMS get/set or action requests from a client (or their SN equivalents).

In DLMS, meter data is requested through Read/Get services by specifying an attribute of an Object. In SN referencing, each attribute of each object has an individual address (called the SN or the base name) whereas in LN referencing each object has an individual address (called LN for Logical/Long Name which is nothing but the OBIS code) and attributes within that object are addressed by an attribute index.

In case of Write/Set operations, the set operation similarly addresses an attribute of an object (directly through the SN or through the LN/Attribute-index pair) and also supplies data to be set.

In all such cases, the SCL will process the request, take care of access-privilege checking and call appropriate methods in the data interface.

Integration to Meter Data

The methods of the data interface are designed to automatically fill in and return the static information of requested objects and the

implementer only needs to add code (in the clearly-marked sections) to fill in the dynamic values from the meter registers.

The default SCL, as it ships, assumes a typical subset of each object to be static and sub-divides the object accordingly. Implementers can modify this sub-division and choose which elements of an object need to be dynamic (where values change dynamically and need to be updated from meter) by simply moving the definition of that element from the static part to the dynamic part of the object.

NOTE: The values for the static elements for all objects are filled in the Configuration interface. If the implementer chooses to modify the static/dynamic sub-division of objects, the implementer will necessarily have to modify the static-initialization of the objects in the Configuration interface to suit the new arrangement

Typically several objects can be configured entirely static. For example the Activity Calendar instance that defines the timing for ratification changes based upon a season/week/day profile can be filled in entirely as a static entity. The default SCL will have several examples filled in to clarify the structure and semantics of each object and the implementer's role may be limited to just editing a small part of the interface

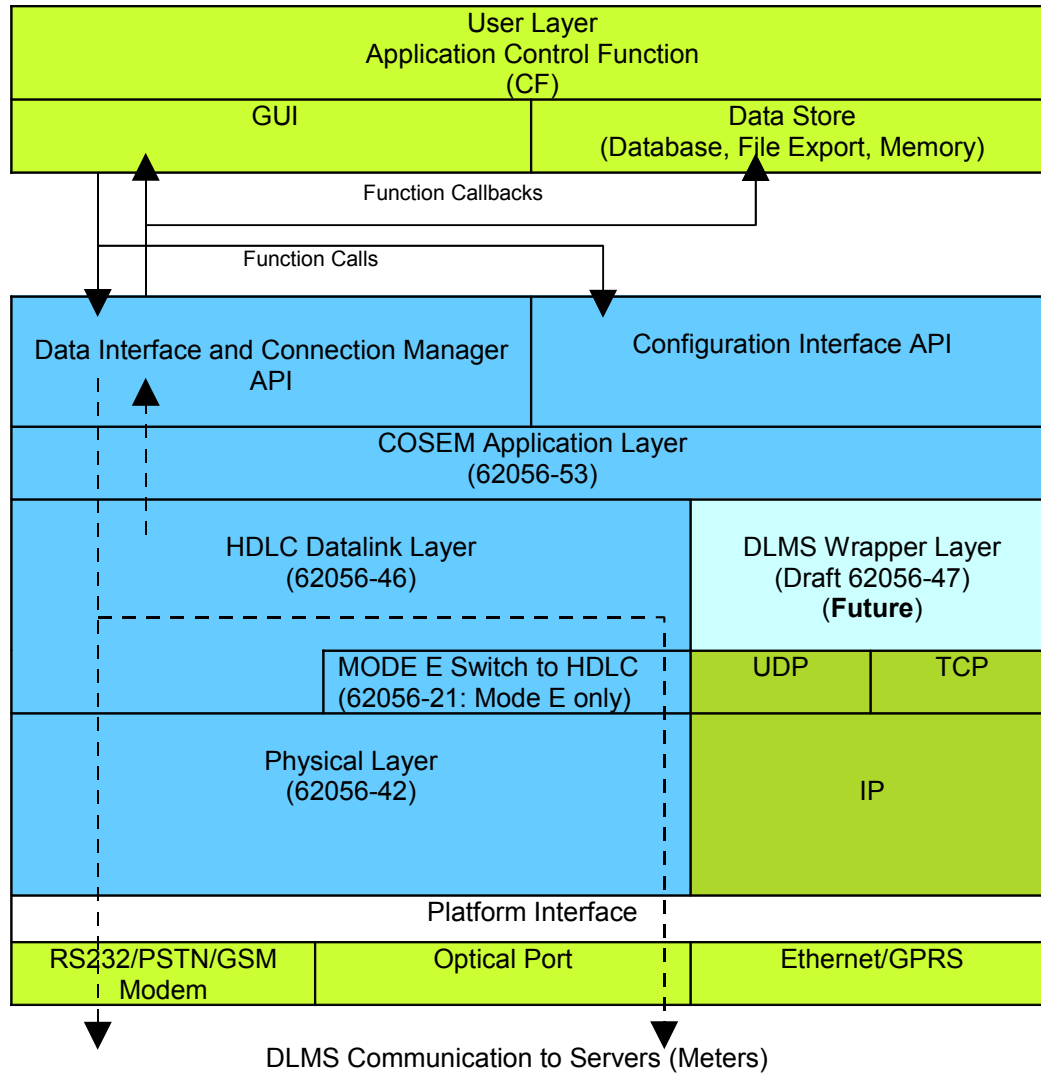
A special design feature allows the implementer to assign a user-handle for each object at edit-time. The type of this handle can be decided by the implementer (as long as it is a statically-initializable type) and can be set to match an Internal ID of each object (if such exists). The implementer can easily retrieve this handle when processing requests and can use it for quicker data retrieval.

Platform Interface

All functionality of the SCL that has to rely upon a platform-specific feature (such as serial port open/close/transmit/receive) are spun out into this interface.

The implementer must add code to perform the function described for each method in this interface and adhere to the structure definitions for the arguments and return values as defined here.

Functional Block Diagram of DLMS SCL & API (Client edition for Meter readers and Parameterization Tools)



Legend

Code	Scope
	Kalki
	Kalki and OEM. For Windows platforms Kalki scope
	Future additions by Kalki
	For Windows, Kalki Scope. For non-Windows platforms OEM Scope
	Meter OEM

The DLMS client solutions are available as both SCL as well as binary Windows/Linux object library form.

In either form, it is to be noted that the SCL/API does not have a data store (except for configuration information that is stored in memory) and does not retain data received from a Meter between transactions. Also the solution does not have a main method and the implementer is required to create an application launch point (with GUI and data-store, if necessary) that will call the library functions

The interface between a client application and the DLMS client solution is a well-defined single API that incorporates Connection handling, Data requests and Configuration updates.

Making calls to the API can trigger DLMS operations and results as well as data are sent back to the User layer via callback functions. In the SCL form, clients are free to modify the code that invokes the callback functions directly

Ordering Part Numbers:

DLMS Server SCL		KLKSCLDLMS201
DLMS Client SCL		KLKSCLDLMS301
DLMS Server Object Library	for Windows	KLKOLDLMS202
DLMS Client Object Library	for Windows	KLKOLDLMS302
DLMS Server Object Library	for Linux	KLKOLDLMS203
DLMS Client Object Library	for Linux	KLKOLDLMS303

Contact Information

Kalki Communication Technologies Limited
#147, 2nd Floor, 5th Main Road, Sector 7
HSR Layout, Bangalore 560034
INDIA
Phone: +91-80-4052-7900
Fax: +91-80-2572 5473
Web: <http://www.kalkitech.com/>

Sales

E-mail: sales@kalkitech.com

Support and Pre-Sales

Technical Sales Support for DLMS Products
E-mail: supportcps@kalkitech.com