

Companion Specification  
for Energy Metering

# **COSEM Extensions for Value Added Services**

DLMS User Association

version 1.2

date: 8 August 2000

---

# Table of Contents

<b>1.</b>	<b>Introduction.....</b>	<b>3</b>
1.1	Referenced Documents .....	4
<b>2.</b>	<b>Scope.....</b>	<b>5</b>
<b>3.</b>	<b>Basic Principles .....</b>	<b>6</b>
3.1	Object Modelling .....	6
3.2	Object Description Notation .....	7
<b>4.</b>	<b>The COSEM server model .....</b>	<b>9</b>
<b>5.</b>	<b>Value Added Services extensions .....</b>	<b>10</b>
<b>6.</b>	<b>The Interface Classes extensions for VAS.....</b>	<b>13</b>
6.1	EVENT (class_id=100) .....	14
6.2	REMOTE DIGITAL CONTROL (class_id=101) .....	15
6.3	REMOTE ANALOGUE CONTROL (class_id=102) .....	16
6.4	TUNNEL (class_id=103) .....	16
<b>Appendix A: Relation to OBIS.....</b>		<b>18</b>

---

# 1. Introduction

Driven by the need of the utilities to optimise their business processes, the meter becomes more and more part of an integrated metering and billing system. Whereas in the past the commercial value of a meter was mainly generated by its data acquisition and processing capabilities, nowadays the critical issues are system integration and interoperability.

The Companion Specification for Electricity (Energy) Metering (COSEM) considers these challenges by looking at the meter as an integrated part of a commercial process which starts with the measurement of the delivered product (energy) and ends with the revenue collection.

The meter is specified by its "behaviour" as seen from the utility's business processes. The formal specification of the behaviour is based on object modelling techniques (interface classes and objects). The specification of these business objects forms a major part of COSEM.

The COSEM server model represents only the externally visible elements of the meter. The client applications that support the business processes of the utilities, of the customers and of the meter manufacturers make use of this server model. The meter offers means to retrieve its structural model, the attributes of its elements and the measured data. The information can be obtained through read-services to the elements of the model. In addition, specific configuration services are offered to configure the meter, to fine-tune its model and to specify the values of the attributes of the elements.

The set of different objects forms a standardised library from which the manufacturer can assemble (model) its individual products. The elements are designed such that with them the entire range of applications (from residential to commercial and industrial metering) can be covered. The choice of the subset of objects used to build a meter, their instantiation and their implementation are part of the product design and therefore left to the manufacturer. The concept of the standardised metering object library provides the different users and manufacturers with a maximum of *diversity* without having to sacrifice *interoperability*.

The management committee of the DLMS UA gave in June 1998 an approval for a working group Value Added Services (VAS). The result of this VAS working group (5 meetings during a year) is this report.

The following title was chosen for the working group:  
***COSEM objects for Value Added Services.***

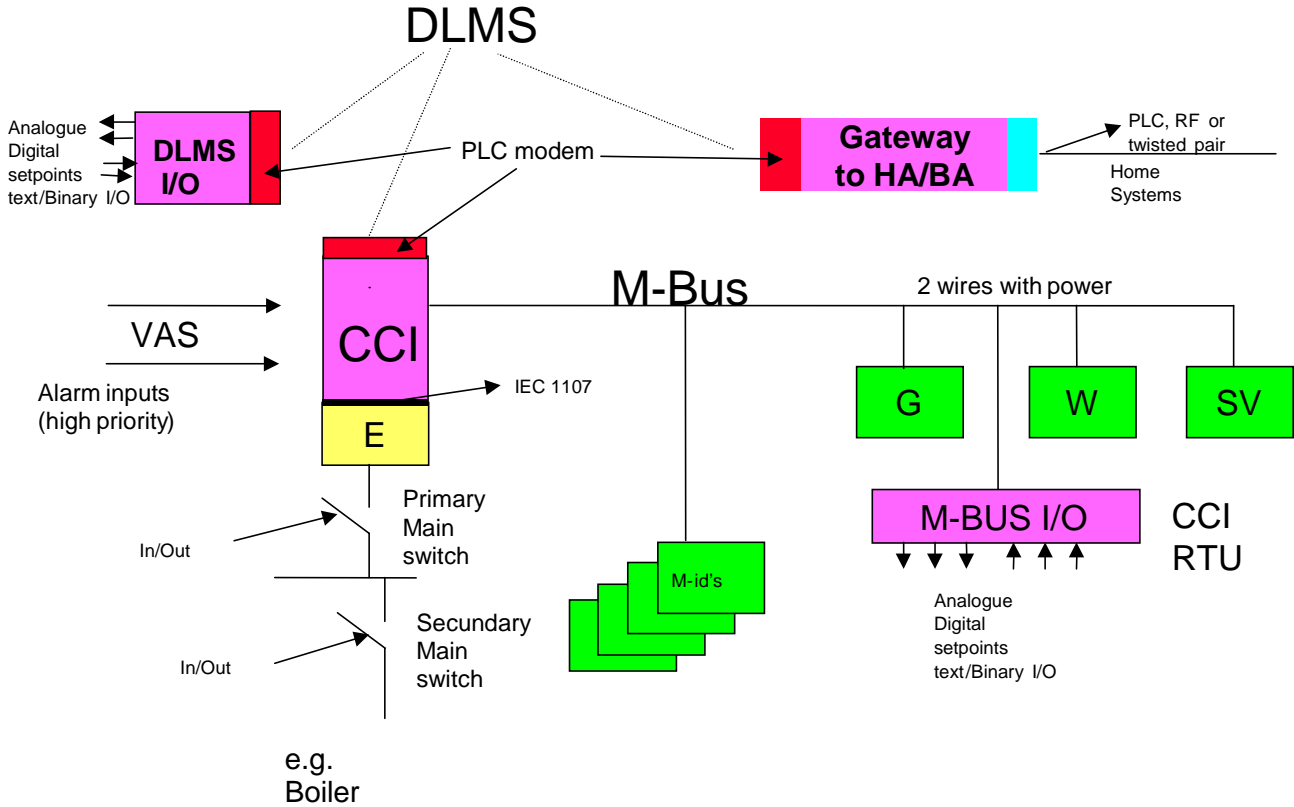
The scope of the working group is:

- Enhance Customer Services by selecting relevant applications
- Defining the objects to provide the selected Value Added Services
- Focus is on remote control applications and event reporting

The following applications we had in mind to cover in this working group:

- Load control
- Remote switching
- Customer (dis)connect
- Remote control
- Contract management
- Container services like mailbox
- Event handling → in relation to alarms (an alarm is an event with high priority)

The following picture introduces some of that new ideas.



[Action point for Tibor Somogyi to select/find a better picture for covering the VAS work]

During the second VAS meeting the following VAS applications for modelling were selected:

- Alarms for different purposes
- Remote controls
- Gateway to Home/Building Automation

## 1.1 Referenced Documents

Ref.	Title
[1]	Distributed Automation Using Distribution Line Carrier Systems Part 4: Data Communication Protocols; Section 4: Application Protocol; Clause 1: Distribution Line Message Specification (DLMS), IEC 61334-4-41, 1996-03
[2]	A-XDR Encoding Rule, IEC 61334-6, CDV, 31 July 1998
[3]	Object Identification System (OBIS), IEC 62056-61
[4]	Technical report: COSEM Interface objects (Blue book); DLMS UA 1000-1: 2000, Third Edition
[5]	Technical Report: COSEM Communication Profile (Green book); DLMS UA 1000-2: 1998, Release 1

## 2. Scope

This document specifies the functionality of Value Added Services (VAS) which is available at its interface (internal issues concerning the implementation are not covered by the specification) and how the functions and the data can be accessed from the outside. The functionality of these VAS functions is divided into generic building blocks. The COSEM specifications follow a three step approach as illustrated in Figure 1:

- step1: The VAS model [this document → Possibke new Yellow book]
- step2: The mapping of the model into protocol data units (pdu)
- step3: The transportation of the bits and bytes through the communication channel

The contents of this document concentrate on step 1 for Value Added Services (VAS) extensions to the basic applications. Step 2 is covered in the appendix of DLMS UA 1000-1[4]. Step 3 is treated in DLMS UA 1000-2 [6].

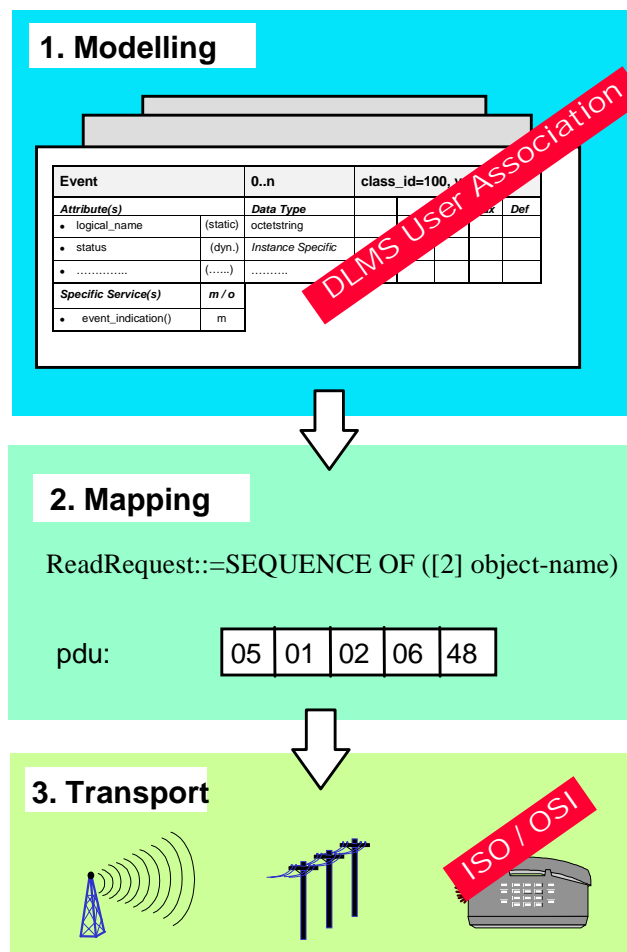


Figure 1: The three steps approach of COSEM: Modelling of the functionality - Mapping to bits and bytes - Transporting via a specific channel using the appropriate protocol.

# 3. Basic Principles

This section describes the basic principles on which the COSEM interface classes are built. It also gives a short overview on how interface objects (instantiations of the interface classes) are used for communication purposes. Meters, VAS devices, support tools and other system components that follow these specifications can communicate with each other in an interoperable way.

## 3.1 Object Modelling

Object modelling: For specification purposes this document uses the technique of object modelling. An object is a collection of attributes and methods. The information of an object is organised in attributes. They represent the characteristics of an object by means of attribute values. The value of an attribute may affect the behaviour of an object. The first attribute in any object is the "logical\_name". It is one part of the identification of the object.

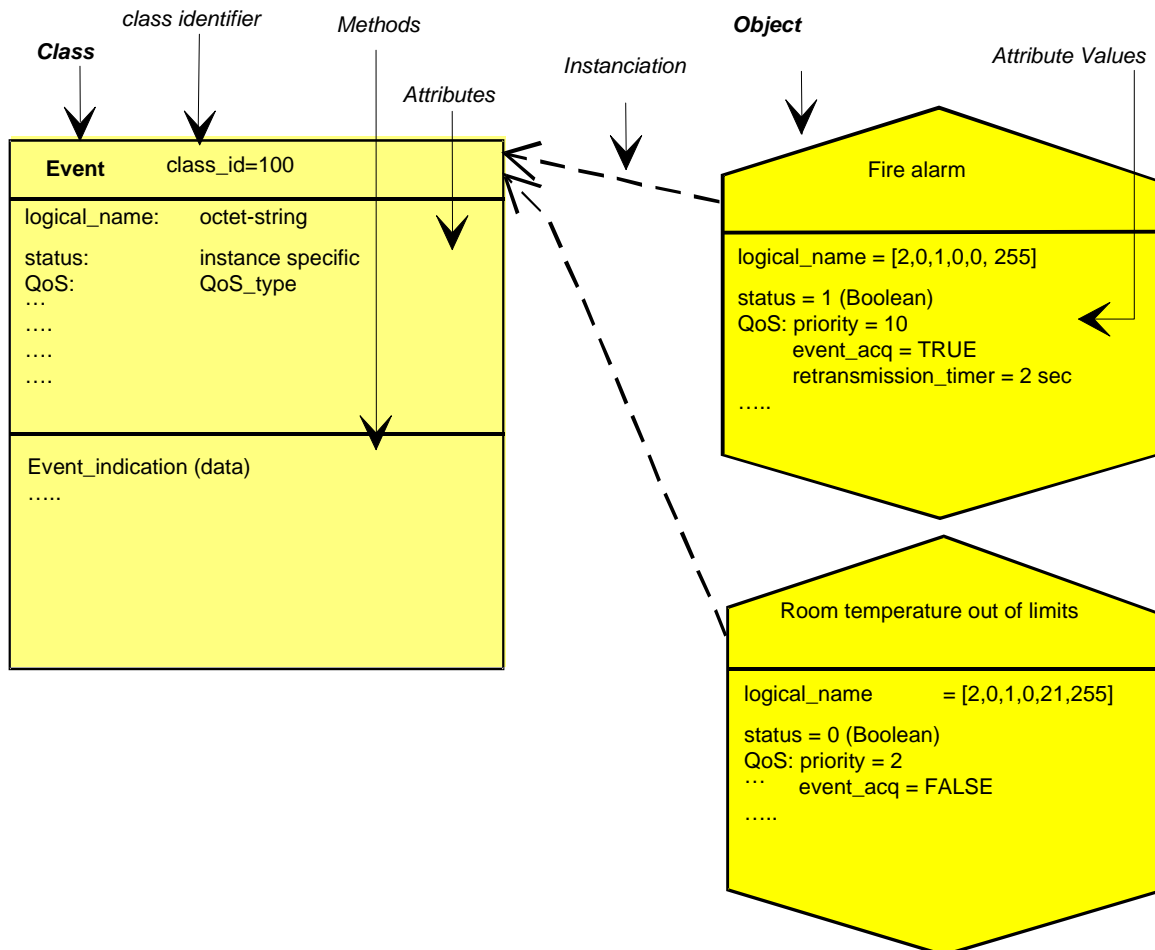


Figure 2: A class and the instantiation of objects

An object offers a number of methods to either examine or modify the values of the attributes. Objects that share common characteristics are generalized as a class with a class\_id. Within a specific class the common characteristics (attributes and methods) are described once for all objects. Instantiations of a class are called objects. Figure 2 illustrates these terms by means of an example. The class "Event" is formed by combining the features necessary to model the behavior of a

generic event (containing event information or “static” information like ‘priority’) as seen from the client (central unit, hand held terminal). The contents of the event are identified by the attribute “logical\_name”. The logical\_name contains an OBIS identifier (comp [3]). The actual (dynamic) content of the event is carried by its “status” attribute.

Defining a specific VAS device means defining several specific events. In the example of Figure 2 the meter contains 2 events; i.e. two specific objects of the class “event” are instantiated. This means that specific values are assigned to the different attributes. Through the instantiation one event becomes a “fire alarm” whereas the other becomes a “temperature out of limits event”.

### 3.2 Object Description Notation

This section describes the notation used to define the COSEM objects. A short text describes the functionality and application of the class.

A table gives an overview of the class including the class name, the attributes and the methods:

Class name	Cardinality	class_id, version		
<b>Attribute(s)</b>	<b>Data Type</b>	<b>Min</b>	<b>Max</b>	<b>Def</b>
1. logical_name (static)	octetstring		...	...
2. . (..)	..		..	..
3.				
<b>Specific Method(s) (if required)</b>	<b>m/o</b>			
1.	..			
2. ..				

Table 1: Object description template

Each attribute and method must be described in detail.

<b>Class name</b>	Describes the class (e.g. Event, Register, Clock, Profile, ...)
<b>Cardinality</b>	Specifies the number of instances of the class within a logical device. <i>val</i> The class shall be exactly “val” times instantiated <i>min..max</i> The class shall be instantiated at least “min” times and at most “max” times. If min is zero (0) then the class is optional, otherwise (min>0) “min” instantiations of the class are mandatory
<b>Class_id</b>	Identification code of the class (range 0 to 65535). The class_id is provided by methods of an “Association” object. The class-id's from 0 to 8191 are reserved to be specified by the DLMS User association. Class-id's from 8192 to 32767 are reserved for manufacturer specific interface classes. Class-id's from 32767 to 65535 are reserved for user group specific interface classes. DLMS UA reserved the right to assign ranges to individual manufacturers or user groups.
<b>Version</b>	Identification code of the version of the class. The version is provided by methods of an “Association” object. <b>Within one logical device all instances of a certain class must conform to the same version</b>
<b>Attribute(s)</b>	Specifies the attribute(s) that belong to the class <i>(dyn.)</i> Classifies an attribute that carries a process value which is updated by the meter/device itself. <i>(static)</i> Classifies an attribute which is not updated by the meter/device itself (e.g., configuration data)

<b>Logical_name</b>	Octet-string	The logical name is always the first attribute of a class. It identifies the instantiation (object) of this class. The value of the logical_name conforms to OBIS [3].
<b>Data Type</b>	Defines the data type of an attribute	
<b>Min</b>	Specifies if the attribute has a minimum value	
	x	The attribute has a minimum value
	<empty>	The attribute has no minimal value
<b>Max</b>	Defines if the attribute has a maximum value	
	x	The attribute has a maximum value
	<empty>	The attribute has no maximum value
<b>Def</b>	Specifies if the attribute has a default value. This is the value of the attribute after reset	
	x	The attribute has a default value
	<empty>	The default value is not defined by the class definition
<b>Specific Methods</b>	Provides a list of the specific methods (beside those declared in the conformance block) that belong to the object	
	<i>Method Name ()</i>	The method has to be described in the subsection "Method Description"
<b>m/o</b>	Defines if the method is mandatory or optional	
	<i>m (mandatory)</i>	The method is mandatory
	<i>o (optional)</i>	The method is optional

## Attribute Description

Describes each attribute with its data type (if the data type is not simple), its data formats and its properties (Minimum value, Maximum value and Default value)

## Method Description

Describes each method and the invoked behaviour of the instantiated object(s).

Note: Services for accessing attributes or methods (READ/WRITE or GET.SET, ACTION, as defined in the conformance block) are considered to be pre-defines. Therefore they are not included here

# 4. The COSEM server model

This chapter gives an overview of the basic concepts used to model a COSEM server. A more detailed description can be found in the document DLMS UA 1000-2.

The COSEM server is structured into 3 hierarchical levels as shown in Figure 3:

- level 1: the physical device
- level 2: the logical device
- level 3: the objects

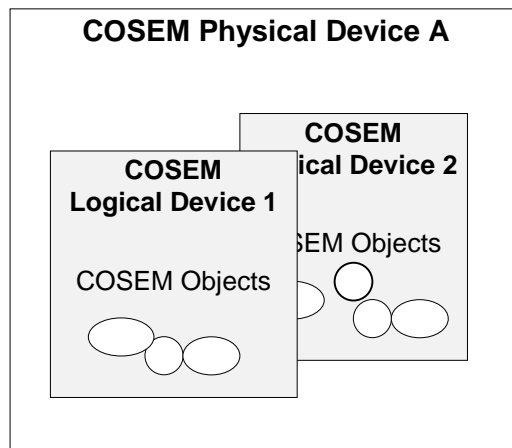
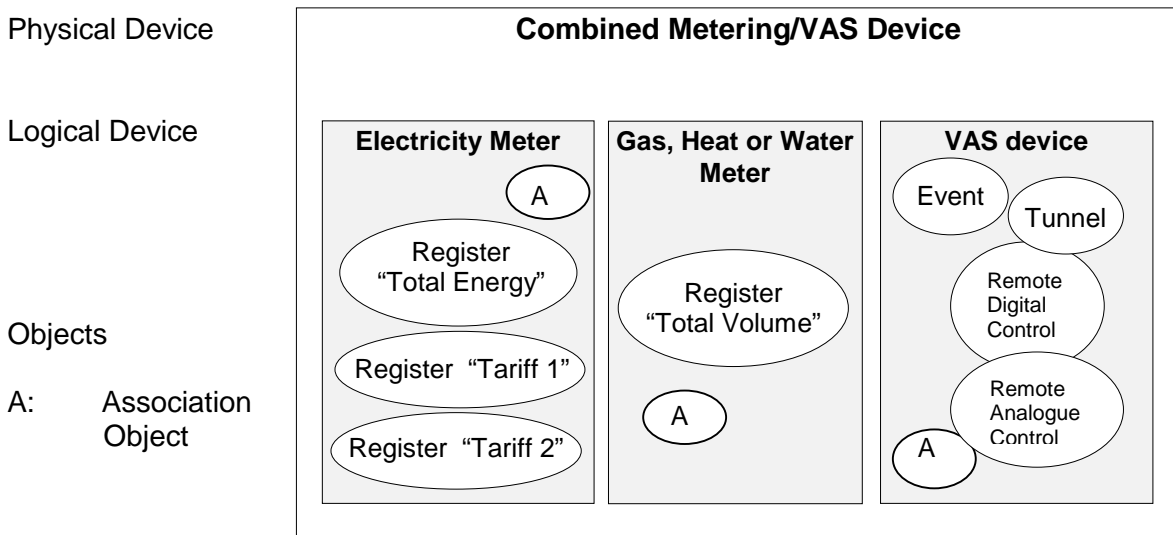


Figure 3: The COSEM server model

The following example shows how a combined metering or VAS device can be structured using the COSEM server model.



---

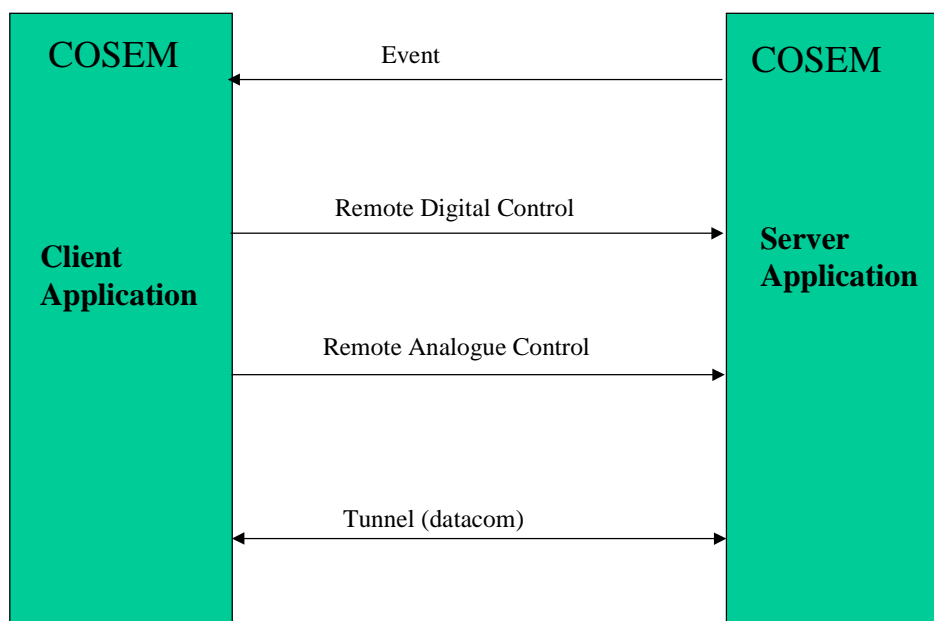
## 5. Value Added Services extensions

This document describes four extensions to the existing COSEM model. These objects were developed because they were necessary to implement possible Value Added Services (VAS).

The new defined objects/ Interface Classes (IC) are:

- Event
- Remote Digital Control
- Remote Analogue Control
- Tunnel

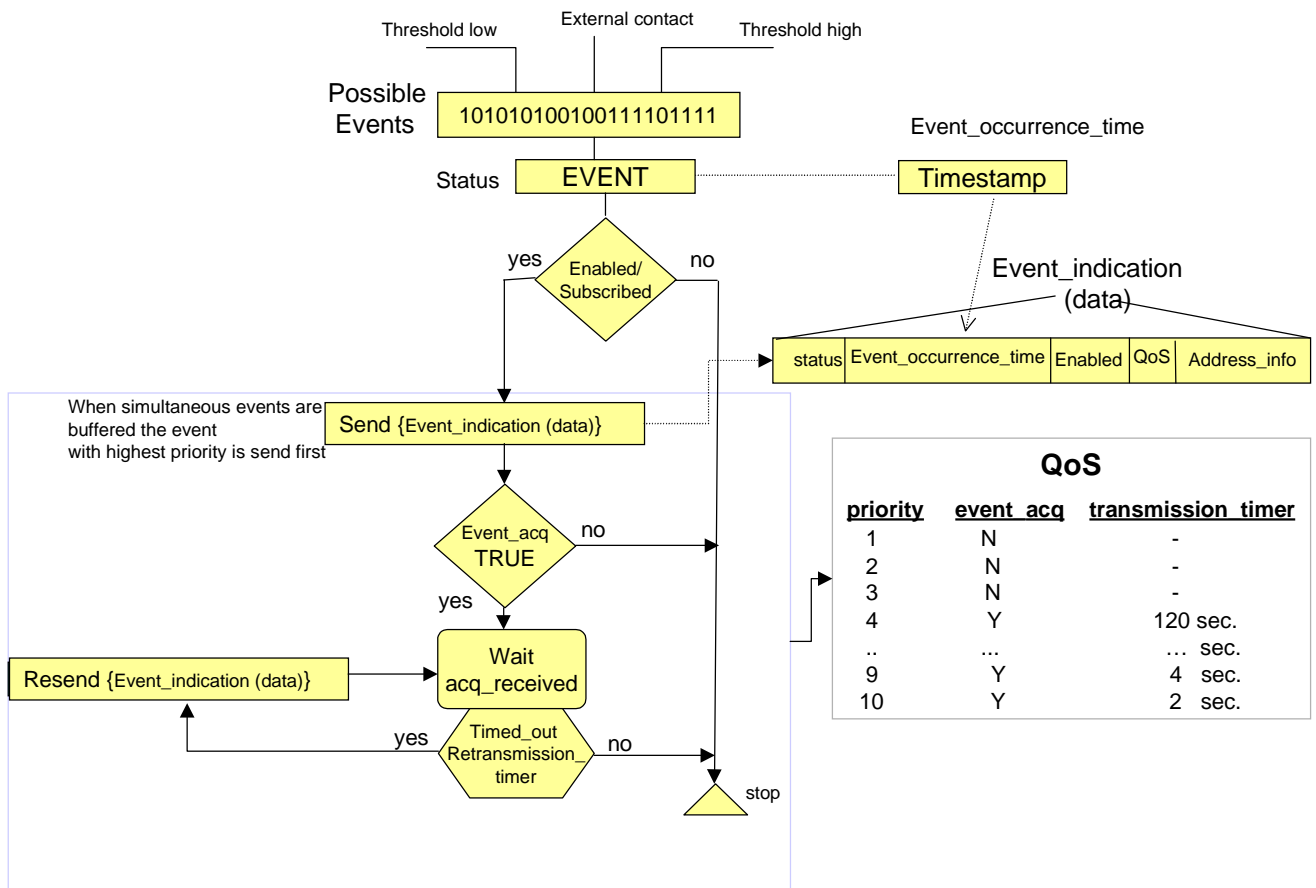
In this chapter for each new interface class (see picture below) an example is given for better understanding the application area of the object/interface class.



Examples :

### Event: (see chapter 6.1)

When a specified threshold is reached (eg. guarded by a monitor register object) or an external contact is activated this can be reported to a client system by using an Event as object. This Event can be enabled (subscribed yes/no) by the client system. When subscribed the event will be send with the time of occurrence ('event\_occurrence\_time') as soon as possible. Depending of the attribute 'event\_ack' the Client system needs to acknowledge the receiving of the event using the attribute 'ack\_received'. A retry mechanism is foreseen depending of the priority with 'retransmission\_timer' to check the response time of the acknowledgement by the Client system. After the 'retransmission\_timer' has reached its value and the acknowledgement is not received in the server system/device a retry is executed by the server. Inside the server device this is called the Quality of Service mechanism. The flow of the Event object is visualised in the following picture.



**Remote Digital Control (see chapter 6.2):**

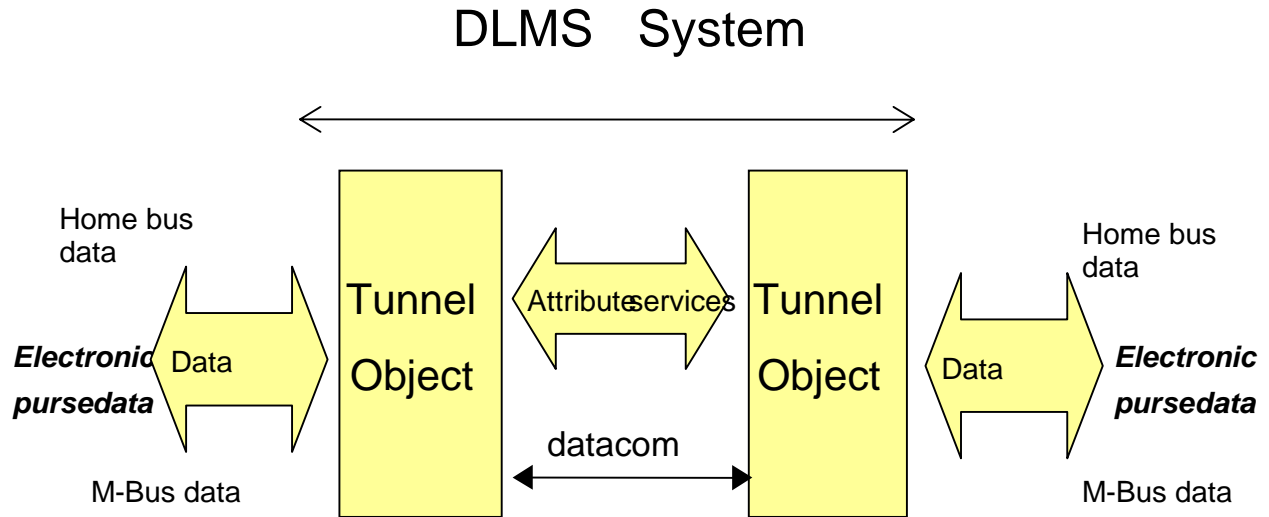
Within the class *Remote Digital Control* it is possible to define a switch array inside the logical device. Each entry can be connected to a digital relay for switching on/off equipment what could be connected to the physical device (which is modelled by the logical device). Each entry has also a 'group\_id' coupled for broadcasting purposes (load shedding/ripple control).

**Remote Analogue Control (see chapter 6.3):**

With *Remote Analogue Control* it is possible to define a setpoint table inside the logical device. Each entry can be connected to a analogue control point of the device (modelled by the logical device). Each entry has also a 'group\_id' coupled for broadcasting purposes.

**Tunnel Object (see chapter 6.4):**

In a vending machine based on electronic purse the creditcard transactions (each about 128 bytes of data) are buffered inside the machine. When for example during a day about 400 transactions are done, the data must be transferred to the bank (interpay) including security aspects. This transfer can be seen as a sort of filetransfer. The amount of data in each transmission is about 50k and the necessary datatransfer could be handled by the new Tunnel-object what is implemented/instantiated in the related vending machine/device/server and in the dataconcentrator/client without knowledge of the datastructure itself. The purpose of the DLMS tunnel object is to transfer 'data' with a defined QoS from place 1 to place 2. It could be a home bus, M-bus or other protocol data. You can call the Tunnel object as a sort of container service. See picture below:



---

## 6. The Interface Classes extensions for VAS

The set of existing Interface Classes (IC) (see DLMS UA 1000-1: 2000 Third Edition) which are already defined are:

- Data (Class\_id:1)
- Register (Class\_id:3)
- Extended Register (Class\_id: 4)
- Demand Register (Class\_id: 5)
- Register Activation (Class\_id: 6)
- Profile Generic (Class\_id:7)
- Clock (Class\_id:8 )
- Script Table (Class\_id: 9)
- Schedule (Class\_id:10)
- Special Days Table (Class\_id: 11)
- Activity Calender (Class\_id:20)
- Association LN (Class\_id: 15)
- Association SN (Class\_id: 12)
- SAP Assignment (Class\_id: 17)
- Register Monitor (Class\_id: 21)

Protocol related Interface classes are:

- IEC Optical Port Setup (Class\_id: 19)
- PSTN Modem Configuration (Class\_id: 27)
- PSTN Auto Answer (Class\_id: 28)
- PSTN Auto Dial (Class\_id: 29)
- IEC HDLC Setup (Class\_id: 23)
- IEC Twisted Pair (1) Setup (Class\_id: 24)

Four new IC's are defined related to VAS services:

- Event (Class\_id: 100)
- Remote Digital Control (Class\_id: 101)
- Remote Analogue Control (Class\_id: 102)
- Tunnel (Class\_id: 103)

These are described in the following paragraphs in more detail.

## 6.1 EVENT (class\_id=100)

An example of the Event object is described in chapter 5.

Aspects related to the definition of the event class are:

- Every change in bits is an event
- Event occurrence time stamp is necessary
- Some DA alarms do not need acknowledgements
- Alarm must be send out until it is acknowledged (check with the use of a retransmission\_timer)
- Alarms must be repeated (when not received within retransmission\_timer)
- Priority is important (using priority range with retransmission\_timer check)
- Optional event acknowledge with response time parameter
  - yes: keep on sending
  - no: send once or twice or more (leave it open)
- number/time of repetition depends on used media
- DLMS address source/destination comes from the lower layers but also the source (which source e.g. IPV6 address) and destination (operator, maintenance department, security department eg. based on IPV6) should be available **inside** the message.

Event		0..n	class_id=100, version=0				
Attribute(s)		Data Type			Min	Max	Def
1. logical_name	(static)	octetstring					
2. status	(dyn.)	<i>Instance Specific</i>					
3. event_occurrence_time	(dyn.)	UTC					
4. enabled	(dyn.)	Boolean					
5. QoS	(static)	QoS_type					
6. address_info	(static)	<i>Instance Specific</i>					
7. ack_received	(dyn.)	Boolean					
Specific Method(s)		M / o					
1. event_indication (data)		m					
2. event_acknowledge ( )		o					

### Attribute Description

<b>status</b>	Contains the current value of the event. <i>Instance Specific</i>	The data type of status depends on the characteristics of the instantiation defined by "logical_name".
<b>event_occurrence_time</b>	UTC	provides an Event specific date and time information showing when the value of the attribute "status" has been occurred.
<b>enabled</b>	Boolean	for maintenance purposes and subscription purposes

<b>QoS</b>	QoS_type ::= structure { prio: Enum event_ack: Boolean retransmission_timer: Time }	To define the wanted Quality of Service of the event; Priority could be in the range of 1 to 128; When event_ack is TRUE the client needs to acknowledge the receiving of the Event. Resending by the server depends of the parameter 'retransmission_timer' this timer could have a value between 1 sec and 24 hours. High priorities normally give short response time values
<b>address_info</b>	<i>Instance Specific</i>	This structure consists of the source (device info, e.g. IPV6) and destination (the department/company to which the event should be send; e.g. IPV6)
<b>ack_received</b>	Boolean	This parameter needs to be set by the Client system when event_ack=TRUE

### Service and Behaviour Description

<b>event_indication(data)</b>	When event occurs an event_indication is send to the Client system with data (status, event_occurrence_time, QoS, address_info)
<b>event_acknowledge ( )</b>	The client system uses this method in case an event was received with event_ack = TRUE

## 6.2 REMOTE DIGITAL CONTROL (class\_id=101)

A *Remote Digital Control* object is already described in chapter 5. The use of group\_id for broadcast purposes is necessary.

Remote Digital Control		0..n	class_id=101, version=0				
Attribute(s)		Data Type			Min	Max	Def
1. logical_name	(static)	octetstring					
2. switch_table	(static)	array of switch_type					
Specific Method(s)	m / o						
1. set_switch (index)	m						

### Attribute Description

<b>array</b>	switch_type ::= structure { switch: Boolean group_id: Unsigned Long }	this is an array of switches; for each entry when value is 1 (ON) then the switch/command/action is activated and when value is 0 (OFF) switch/command/action is deactivated, also for each switch a group_id is defined (eg. load shedding purposes/ ripple control broadcasting)
--------------	--	--

### Service and Behaviour Description

<b>set_switch (index)</b>	index is pointer in array of structure switch_type
---------------------------	--

## 6.3 REMOTE ANALOGUE CONTROL (class\_id=102)

A *Remote Control* object is already described in chapter 5.

Hysteresis is handled locally

- send Hi/Lo setpoint (pair of analogue values)
- when taking into the object it can be used or not (Hi=Lo => no hysteresis)

Remote Analogue Control		0..n	class_id=101, version=0				
Attribute(s)		Data Type			Min	Max	Def
1. logical_name	(static)	octetstring					
2. setpoint_table	(static)	Array of setpoint_type					
Specific Method(s)		m / o					
1. set_setpointpair (index)		m					

### Attribute Description

<b>setpoint_table</b>	setpoint_type ::= structure { setpoint: Analogue value group_id: Unsigned Long }	this is an array of analogue setpoints; for each analogue value a group_id is defined
-----------------------	---	---

### Service and Behaviour Description

<b>set_setpoint (index)</b>	index is pointer in array of analogue setpoints
-----------------------------	---

## 6.4 TUNNEL (class\_id=103)

The Tunnel object is related to the already existing DATA object but has been extended with specific attributes and a method. A Tunnel object is an object that can transfer data of some datasource by DLMS. You can also call it a container/mail service. It is difficult to know which attributes are needed for such an object. In Chapter 5 the idea by an example of the TUNNEL object is given. It is expected that all different sources (M-bus, EHS, LON, eg. electronic payment data) need to implement a corresponding TUNNEL object otherwise mismatches of attributes could occur. Important attributes are source and destination, Quality of Service (QoS), length of data message and End of Message (EoM) indicator. DLMS address source/destination comes from the lower layers but also the source (which source e.g. IPV6 address) and destination (operator, maintenance department, security department eg. based on IPV6) should be **inside** the message.

Tunnel		0..n	class_id=104, version=0				
Attribute(s)		Data Type			Min	Max	Def
1. logical_name	(static)	Octetstring					
2. message	(dyn)	Octetstring					
3. address_info	(static)	Instance Specific					
4. length	(dyn)	Long-unsigned					
5. QoS	(dyn)	Instance Specific					
6. EoM	(dyn)	Boolean					
Specific Method(s)		m / o					
1. event_message (data)		m					

---

## Attribute Description

<b>message</b>		this is the message in datapackets which must be transferred from source to destination (see address_info)
<b>address_info</b>	<i>Instance specific</i>	This structure consists of the source (source device info, eg. IPV6 addressing format) and destination (the department/ company to which the data should be send; eg IPV6 dest. address)
<b>length</b>		the length in bytes of the total message
<b>QoS</b>		Quality of Service (TP0-4) needed to transfer the message from source address to the destination address
<b>EoM</b>		End of Message: This attribute is FALSE until the last part of the source message is received in the buffer of the logical device containing the Tunnel object

## Service and Behaviour Description

<b>event_message (data)</b>	The datamessage including address_info, length, QoS and EoM
-----------------------------	---

# Appendix A: Relation to OBIS

The OBIS identification system (see DLMS UA 1000-3 ed. 1 and [3]) defines the necessary identification codes (ID-codes) for commonly used data items inside metering devices. The ID-codes are based on the current EDIS document. They shall be used as logical names in DLMS-COSEM for the specified purpose exclusively.

OBIS codes are a combination of 6 value groups, which describe (in a hierarchical way) the exact meaning of each data item.

OBIS ID code	A	B	C	D	E	F
Logical_name	Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6

Table: Mapping of OBIS Codes to Logical\_Names

Extensions to the existing OBIS [3] are suggested as follows:

Value group A	
0	Abstract objects
1	Electric related object
<b>2</b>	<b>Value Added Services object</b>
6	Heat related object
7	Gas related object
8	Water related object

For value group B no changes are necessary to support the new four Interface Classes.  
**B=0** is filled in for Value Added Services

Value group C	
Value Added Services (val. A = 2)	
<b>1</b>	<b>Objects of class Event</b>
<b>2</b>	<b>Objects of class Remote Digital Control</b>
<b>3</b>	<b>Objects of class Remote Analogue setpoint</b>
<b>4</b>	<b>Objects of class Tunnel</b>
.....	
....	

Value group D	
Value Added Services (val. A = 2)	
<b>0</b>	<b>("standard" instances )</b>
<b>1</b>	<b>(free space – "special" instances )</b>
<b>2</b>	<b>....</b>
<b>3</b>	<b>....</b>
<b>4</b>	<b>.....</b>
<b>5</b>	<b>....</b>

For value group **E** (for each instantiation within a device a new number, starting with 0, is given) and for **F** (will not be used= 0xFF) no changes are foreseen to support the VAS Interface classes.