Attribute description	
logical_name	Identifies the “IEC HDLC setup” object instance. See 6.2.20.
comm_speed	The communication speed supported by the corresponding port. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud This communication speed can be overridden if the HDLC mode of a device is entered through a special mode of another protocol.
window_size_ transmit	The maximum number of frames that a device or system can transmit before it needs to receive an acknowledgement from a corresponding station. During logon, other values can be negotiated.
window_size_ receive	The maximum number of frames that a device or system can receive before it needs to transmit an acknowledgement to the corresponding station. During logon, other values can be negotiated.
max_info_length_ transmit	The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated.
max_info_length_ receive	The maximum information field length that a device can receive. During logon, a smaller value can be negotiated.
inter_octet_ time_out	Defines the time, expressed in milliseconds, over which, when no character is received from the primary station, the device will treat the already received data as a complete frame.
inactivity_ time_out	Defines the time, expressed in seconds over which, when no frame is received from the primary station, the device will process a disconnection. When this value is set to 0, this means that the inactivity_time_out is not operational.
device_address	Contains the physical device address of a device. In the case of one byte addressing: 0x00 NO_STATION Address, 0x01…0x0F Reserved for future use, 0x10...0x7D Usable address space, 0x7E ‘CALLING’ device address, 0x7F Broadcast address In the case of two byte addressing: 0x0000 NO_STATION address, 0x0001...0x000F Reserved for future use, 0x0010...0x3FFD Usable address space, 0x3FFE ‘CALLING’ physical device address, 0x3FFF Broadcast address

5.6.3 IEC twisted pair (1) setup (class_id = 24, version = 1)

5.6.3.1 Overview

The communication medium *twisted pair with carrier signalling* is widely used in metering. The main advantages of using this medium are the ease of installation and the reliability of communications due to carrier signalling. This medium can be used:

- between Local Network Access Points (LNAPs) and metering end devices (M interface);
- between Local Network Access Points (LNAPs) and Neighbourhood Network Access Points (NNAPs); and
- for direct connection between a HHU and the metering end device.

IEC 62056-3-1:2013 specifies three communication profiles using the medium twisted pair with carrier signalling:

- without DLMS;
- with DLMS; and
- with DLMS/COSEM.

IEC 62056-31:1999 supports only the first two profiles.

The new, DLMS/COSEM profile introduces a Support Manager Layer entity performing the initialisation of the bus, discovery management, alarm management and communication speed negotiation. It also allows higher baud rates up to 9 600 Bd. The Transport Layer supports segmentation and reassembly.

The IC “IEC Twisted pair (1) set up” (class_id = 24, version = 0) supports the first two communication profiles specified in IEC 62056-31:1999.

The new version 1 supports the DLMS/COSEM profile. With its introduction, the use of version 0 is deprecated.

The use of the communication profiles specified in IEC 62056-3-1:2013 requires using the registration services provided by the Euridis Association: www.euridis.org.

The following COSEM interface objects are necessary to set up data exchange over the medium *Twisted pair with carrier signalling:*

- “IEC Twisted pair (1) setup”: class_id = 24, version = 1;
- “MAC address”: class_id = 43, version = 0;
- “Data”: class_id = 1, version = 0.

For OBIS codes, see clause 6.2.21.

5.6.3.2 IEC twisted pair (1) setup (class_id = 24, version = 1)

Instances of this IC allow setting up data exchange over the medium *twisted pair with carrier signalling* as specified in IEC 62056-3-1:2013. Several communication channels can be configured.

IEC twisted pair (1) setup		0...n	class_id = 24, version = 1			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum	0		1	x + 0x08
3. comm_speed	(static)	enum	(2)	(7)	(2)	x + 0x10
4. primary_address_list	(static)	primary_address_list_type				x + 0x18
5. tabi_list	(static)	tabi_list_type				x + 0x20
Specific methods		***m/o***				

Attribute description	
logical_name	Identifies the “IEC twisted pair setup” object instance. See 6.2.21.
mode	This attribute specifies the working mode of this interface enum: (0) inactive. The interface ignores all frames received, (1) always active, (2) to (127) reserved, (128) to (250) manufacturer specific.
comm_speed	Holds the communication speed supported by the port. enum: (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud NOTE IEC 62056-3-1:2013 supports baud rates from 1 200 to 9 600.
primary_address_list	Holds the list of Primary Station Addresses (ADP) for which each logical device of the real equipment (the secondary station) has been programmed. See IEC 62056-3-1:2013, 5.2.4. primary_address_list_type::= array unsigned
tabi_list	Represents the list of the TAB(i) for which the real equipment (the secondary station) has been programmed in the case of forgotten station call (see IEC 62056-3-1:2013). When using IEC 62056-3-1 profile with DLMS/COSEM, the *tabi_list* attribute is made of only one element of value 0, the value used for the discovery process. tabi_list_type::= array tabi_element tabi_element: integer

5.6.3.3 MAC address

Secondary stations – typically metering end devices – hold a secondary station address (ADS). The length of the ADS shall be 6 octets and consists of the elements shown in Table 30. This address shall be worldwide unique and it shall assigned by the manufacturer to the secondary station. The ADS assigned is valid through the lifetime of the secondary station.

Table 30 – ADS address elements

Elements of ADS	Description	Length (byte)	Type	Range
manufacturer_id	Assigned upon request by the Euridis Association to the manufacturer. It shall be used in all devices using the *Twisted pair with carrier signalling* medium and made by this manufacturer.	1	BCD	0-99
year_of_manufacture	Holds the year of manufacturing the device (the lower two numbers only).	1		0-99
equipment_id	Related to the type of equipment concerned and provides information on the functionality available. Like the manufacturer_id, it is assigned by the Euridis Association.	1		0-99
device_serial_number	The manufacturing number of the device assigned by the manufacturer. It should have the same value as the value held by the object Device ID 1, see IEC 62056-6-1:2017, Table 8.	3		000001-999999 000000 is reserved
EXAMPLE 031267123456: 03: ITRON International; 12: Year 2012; 67: Smart meter LINKY Single phase 123456: Serial number beginning each year from 000001.				

5.6.3.4 Fatal error register

Each device implementing the DLMS/COSEM communication profile specified in IEC 62056-3-1:2013 shall provide an error register holding the result of the last communication with the primary station. The structure of the fatal error register shall be as specified in Table 31.

Table 31 – Fatal error register

Ref	Name	Description
Bit 0	EP-3F	Transmission error. The time out TOE is elapsed without the byte being sent, leading to a non-ability to send the remaining part of the frame.
Bit 1	EP-4F	Reception error. The number of bytes received is higher than the maximum expected.
Bit 2	EP-5F	Expiry of TARSO wake-up while receiving an RSO frame. Not relevant for secondary station (server); concerns the primary station (client) only.
Bit 3	EL-1F	Alarm indication received during an association. No relevant. Concern the primary station (client) only.
Bit 4	EL-2F	Incorrect response from the Secondary Station after MaxRetry repeated transmissions of a request.
Bit 5	EA-1F	Incorrect TAB from the server. Not relevant for secondary station (server); concerns the primary station (client) only.
Bit 6	EA-2F	Authentication error on the data received from the server. Not relevant for secondary station (server); concerns the primary station (client) only.
Bit 7	EA-3F	Authentication error detected by the secondary station.

5.6.4 Modem configuration (class_id = 27, version = 1)

This IC allows modelling the configuration and initialisation of modems used for data transfer from/to a device. Several modems can be configured.

Modem configuration		0...n	class_id = 27, version = 1			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	9	5	x + 0x08
3. initialization_string	(static)	array				x + 0x10
4. modem_profile	(static)	array				x + 0x18
Specific methods		***m/o***				

Attribute description	
logical_name	Identifies the “Modem configuration” object instance. See 6.2.6.
comm_speed	The communication speed between the device and the modem, not necessarily the communication speed on the WAN. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud

initialization_ string	Contains all the necessary initialization commands to be sent to the modem in order to configure it properly. This may include the configuration of special modem features.
	
	array initialization_string_element
	
	initialization_string_element::= structure
	{
	request: octet-string,
	response: octet-string,
	delay_after_response: long-unsigned
	}
	If the array contains more than one initialization_string_element, the requests are sent in a sequence. The next request is sent after the expected response matching the previous request and waiting a delay_after_response time [ms], to allow the modem to execute the request.
	NOTE It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization is needed, the initialization string is empty.

modem_profile	Defines the mapping from Hayes standard commands/responses to modem specific strings. array modem_profile_element modem_profile_element: octet-string The modem_profile array shall contain the corresponding strings for the modem used in following order: Element 0: OK, Element 1: CONNECT, Element 2: RING, Element 3: NO CARRIER, Element 4: ERROR, Element 5: CONNECT 1 200, Element 6 NO DIAL TONE, Element 7: BUSY, Element 8: NO ANSWER, Element 9: CONNECT 600, Element 10: CONNECT 2 400, Element 11: CONNECT 4 800, Element 12 CONNECT 9 600, Element 13: CONNECT 14 400, Element 14: CONNECT 28 800, Element 15: CONNECT 33 600, Element 16: CONNECT 56 000

5.6.5 Auto answer (class_id = 28, version = 2)

NOTE 1 Version 1 of the Auto answer class was an interim version.

Version 0 of the Auto answer class models how the device handles incoming calls to request the connection of the modem.

In version 2, new capabilities are added to manage wake-up requests that may be in the form of a wake-up call or a wake-up message e.g. an (empty) SMS message. After a successful wake-up request, the device connects to the network. See also Annex A.

For both functions, additional security is provided by adding the possibility of checking the calling number: calls or messages are accepted only from a pre-defined list of callers. This feature requires the presence of a calling line identification (CLI) service in the communication network used.

NOTE 2 The wake-up process is fully decoupled from AL services, i.e. a wake-up message cannot contain any xDLMS service requests. This is to avoid creating a backdoor. xDLMS messages may be exchanged in SMS messages once the wake-up process is completed.

Auto answer		0...n	class_id = 28, version = 2			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. listening_window	(static)	array				x + 0x10
4. status	(dyn.)	enum				x + 0x18
5. number_of_calls	(static)	unsigned				x + 0x20
6. number_of_rings	(static)	nr_rings_type				x + 0x28
7. list_of_allowed_callers	(static)	array				x + 0x30
Specific methods		***m/o***				

Attribute description	
logical_name	Identifies the “Auto answer” object instance. See 6.2.6.
mode	Defines the working mode of the line when the device is auto answering. enum: (0) line dedicated to the device, (1) shared line management with a limited number of calls allowed. Once the number of calls is reached, the window status becomes inactive until the next start date, whatever the result of the call, (2) shared line management with a limited number of successful calls allowed. Once the number of successful communications is reached, the window status becomes inactive until the next start date, (3) currently no modem connected, (200...255) manufacturer specific modes

listening_ window	Defines the time points when the communication window(s) become active (start_time) and inactive (end_time). The start_time implicitly defines the period.
	EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily listening window management. Daily, monthly …window management can be defined.
	array window_element
	
	window_element::= structure
	{
	start_time: octet-string,
	end_time: octet-string
	}
	start_time and end_time are formatted as specified in 4.6.1 for *date-time.*

status	Here the status of the window is defined. enum: (0) Inactive: the device will manage no new incoming call. This status is automatically reset to Active when the next listening window starts, (1) Active: the device can answer to the next incoming call, (2) Locked: This value can be set automatically by the device or by a specific client when this client has completed its reading session and wants to give the line back to the customer before the end of the window duration. This status is automatically reset to Active when the next listening window starts.
number_of_ calls	This number is the reference used in modes 1 and 2. When set to 0, this means there is no limit.

number_of_ rings	Defines the number of rings before the meter connects the modem. Two cases are distinguished: The number of rings within the window defined by the attribute *listening_window* and the number of rings outside the *listening_window*.
	nr_rings_type::= structure
	{
	nr_rings_in_window: unsigned (0 = no connection in the window),
	nr_rings_out_of_window: unsigned (0 = no connection outside the window)
	}
	If the number of rings inside and outside the window is the same, the modem always connects regardless of the settings of the *listening_window*.

list_of_ allowed_ callers	Contains an – optional – list of calling numbers which further limits the connectivity of the modem based on the calling number. It also controls the acceptance of wake-up calls or wake-up messages (e.g. SMS) from a calling number.
	This requires the presence of a calling line identification (CLI) service in the communication network used.
	list_of_allowed_callers::= array list_of_allowed_callers_element
	list_of_allowed_callers_element::= structure
	{
	caller_id: octet-string,
	call_type: enum
	}
	– the caller_id element holds a calling number from which calls or messages (e.g. SMS) are accepted. The wild-card characters '?' and '*' are supported. With '?' any single character matches, with '*' any character string matches. '*' can only be used at the beginning or at the end of a number, but neither in between nor alone.
	Example 1: "+994193500" = only calls from "+994193500" are accepted.
	Example 2: "+9941935????" = calls from all numbers in the range of "+99419350000" to "+99419359999" are accepted.
	Example 3: "7777*" = calls from all numbers starting with "7777" are accepted.
	Example 4: "*9000" = calls from all numbers ending with "9000" are accepted.

list_of_ allowed_ callers (continued)	– the call_type element defines the purpose of the call, i.e. if it's a standard CSD call or a wake-up call / wake-up message. enum: (0) = normal CSD call; the modem only connects if the calling number matches an entry in the list. This is tested in addition to all other attributes, e.g. *number_of_rings*, *listening_windows*, etc. (1) = wake-up request; calls or messages from this calling number are handled as wake-up requests. The wake-up request is processed immediately regardless of all other attributes like *number_of_rings* and *listening_window* (except if the calling number is also present in the list of normal CSD calls, see below). The received message shall be completely empty; otherwise it is not treated as a wake-up message. If the message contains a valid xDLMS APDU from a client in a pre-established AA, the corresponding xDLMS service is executed instead of processing the wake-up request. If the message is not empty but does not contain any valid xDLMS APDU from a client in a pre-established AA then the server does not react. For a call of type (1) the modem *does not* connect – independently of the outcome of the wake-up process. For a call of both type (1) and type (0): see below. If the same calling number is defined as initiator of a normal CSD call (type (0)) and as initiator of a wake-up request (type (1)) the following rule applies for the incoming calls: the wake-up request is only processed if the caller disconnects the line before the *number_of_rings* criterion (depending on *listening _window*) is reached. If the *number_of_rings criterion* is met then a modem connection is established. The *number_of_rings* parameter shall be set large enough to allow the initiator of the call to control the behaviour of the receiver of the call without any knowledge of the time instances of the rings at the receiver's side. NOTE 3 If the *list_of_allowed_callers* is empty (= array [0]) the auto answer function operates in type (0) = normal CSD call and the modem connects independently of the calling number.

5.6.6 Auto connect (class_id = 29, version = 2)

Version 1 of the "Auto connect" class models how the device performs auto dialling or sends messages using various services.

In version 2 new capabilities are added to model the connection of the device to a communication network. Network connection may be permanent, within a time window or on invocation of the connect method.

Auto connect		0...n	class_id = 29, version = 2			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. repetitions	(static)	unsigned				x + 0x10
4. repetition_delay	(static)	long-unsigned				x + 0x18
5. calling_window	(static)	array				x + 0x20
6. destination_list	(static)	array				x + 0x28
Specific methods		***m/o***				
1. connect (data)		o				x + 0x30

Attribute description	
logical_name	Identifies the “Auto connect” object instance. See 6.2.6.
mode	Controls the auto connect functionality in terms of the timing, the message type and the infrastructure to be used. Modes (1) to (3) are dedicated to CSD services. Modes (4) to (6) are dedicated to sending specific messages using a specific infrastructure. Modes (101) to (104) apply to packet switched network connections only (e.g. GPRS). enum: (0) no auto connect; the device never connects, (1) auto dialling allowed anytime, the values defined in the *calling_window* are ignored, (2) auto dialling allowed within the validity time of the *calling_window*, (3) “regular” auto dialling allowed within the validity time of the *calling_window*; “alarm” initiated auto dialling allowed anytime, (4) SMS sending via Public Land Mobile Network (PLMN), (5) SMS sending via PSTN, (6) email sending, (7...99) reserved, (101) the device is permanently connected to the communication network, (102) the device is permanently connected to the communication network within the validity time of the calling window. The device is disconnected outside the calling window. No connection possible outside the calling window, (103) the device is permanently connected to the communication network within the validity time of the calling window. The device is disconnected outside the calling window but it connects to the communication network as soon as the connect method is invoked,
mode (continued)	(104) the device is usually disconnected. It connects to the communication network as soon as the connect method is invoked, (105…199) reserved, (200...255) manufacturer specific modes.