ICS 17.220.20
N 22

中华人民共和国国家标准

GB/T 19882.32—2007/IEC 62056-62:2002

自动抄表系统
第3-2部分:应用层数据交换协议　接口类

Automatic meter reading system—
Part 3-2: Application layer data exchange protocols—Interface clasess

(IEC 62056-62: 2002, Electricity metering—
Data exchange for meter reading, tariff and laod control—
Part 62: Interface clasess, IDT)

2007-10-11 发布　　2007-12-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　次

前　言

《自动抄表系统　应用层数据交换协议》分为三个部分：

——第 3-1 部分：对象标识系统(IEC 62056-61:2002,IDT)；

——第 3-2 部分：接口类(IEC 62056-62:2002,IDT)；

——第 3-3 部分：COSEM 应用层(IEC 62056-53:2002,IDT)。

本部分为《自动抄表系统　应用层数据交换协议》的第 3-2 部分。

本部分为等同采用 IEC 62056-62:2002。

《自动抄表系统　应用层数据交换协议》是《自动抄表系统》国家标准体系的一个重要组成部分。下面列出《自动抄表系统》国家标准的颁布和预计结构及对应的国际标准。

a)　GB/T 19882.1—2005《自动抄表系统　总则》

b)　《自动抄表系统　抄表系统》

——第 2-1 部分：低压电力线载波抄表系统；

——第 2-2 部分：无线通信抄表系统；

——第 2-3 部分：基于 IP 网络的抄表系统。

c)　《自动抄表系统　应用层数据交换协议》

——第 3-1 部分：对象标识系统(IEC 62056-61:2002,IDT)；

——第 3-2 部分：接口类(IEC 62056-62:2002,IDT)；

——第 3-3 部分：COSEM 应用层(IEC 62056-53:2002,IDT)。

d)　GB/T 19897—2005《自动抄表系统　低层通信协议》

——第 1 部分：直接本地数据交换 (IEC 62056-21:2002,IDT)；

——第 2 部分：基于双绞线载波信号的局域网使用 (IEC 62056-31:1999,IDT)；

——第 3 部分：面向连接的异步数据交换的物理层服务进程 (IEC 62056-42:2002,IDT)；

——第 4 部分：基于 HDLC 协议的链路层(IEC 62056-46:2002,IDT)。

本部分的附录 A、附录 B、附录 C、附录 D 为规范性附录，附录 E 为资料性附录。

本部分由中国机械工业联合会提出。

本部分由全国电工仪器仪表标准化技术委员会归口。

本部分起草单位：哈尔滨电工仪表研究所、西北电力试验研究院、西安汇通测控技术有限公司、广州科立通用电气公司、北京握奇智能科技有限公司、杭州智源电子有限公司、北京供电局、华立集团公司、浙江正泰仪器仪表有限公司、长沙威胜仪表有限公司、深圳开发科技有限公司、南京电力自动化研究院。

本部分主要起草人：杨晓西、黄国兵、区建斌、陈红军、张小平、范国平、郑小平、郭华喜、冯玉贵、关文举。

引　言

公共事业部门为获取最佳商业利益，在计量计费系统中越来越多地使用仪表设备。在过去，仪表的主要商业价值来源于其数据采集与处理能力，如今主要来源于系统集成与互操作性。

能量计量配套技术规范(COSEM)致力于解决把仪表看成是一个商业过程的组成部分后带来的诸多挑战，这个商业过程从对交付产品(能量)的测量开始，到费用征收为止。

根据公共事业部门的商业过程的特点，仪表的规范由其"行为特征"来确定，正式技术规范建立在对象建模技术(接口类与对象)的基础上，这些对象的技术规范构成了 COSEM 的主要部分。

COSEM 服务器模型(见 4.5)仅代表仪表的外部可见部分。支持公共事业部门、用户和仪表制造商的客户机应用都采用了这种服务器模型。仪表为追索其结构模型(在表盘上可看到的对象列表)提供了检索手段，并为这些对象的属性和特殊处理方法提供了访问措施。

各种不同接口类的集合构成一个标准库，制造商可选择这个标准库的一个子集来生产(建模)自己的产品。元件的设计要求能适用于所有产品(从居民到商业和工业应用)。为制造一种仪表选择接口类子集，以及其安装和实现方法都是产品设计的组成部分，这些工作留给制造商自行处理。标准化仪表接口类库的概念为不同的用户和制造商提供多种多样的选择而且又不失互操作性。

自动抄表系统
第 3-2 部分:应用层数据交换协议　接口类

1　范围

本部分规定了从通信接口视角观看的仪表模型。本部分采用面向对象的方法定义了多种通用组合块,以接口类的形式构造了简单功能到非常复杂功能的仪表模型。

2　规范性引用文件

下列文件中的条款通过 GB/T 19882 的本部分的引用而成为本部分的条款。凡是注日期的引用文件,其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本部分,然而,鼓励根据本部分达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件,其最新版本适用于本部分。

GB/T 19897.1—2005　自动抄表系统低层通信协议　第 1 部分:直接本地数据交换 (IEC 62056-21:2002,IDT)

GB/T 19897.2—2005　自动抄表系统低层通信协议　第 2 部分:基于双绞线载波信号的局域网使用 (IEC 62056-31:1999,IDT)

GB/T 19897.4—2005　自动抄表系统低层通信协议　第 4 部分:基于 HDLC 协议的链路层 (IEC 62056-46:2002,IDT)

GB/T 19882.31—2007　自动抄表系统　第 3-1 部分:应用层数据交换协议　对象标识系统 (IEC 62056-61:2002,IDT)

GB/T 19882.33—2007　自动抄表系统　第 3-3 部分:应用层数据交换协议　COSEM 应用层 (IEC 62056-53:2002,IDT)

DL/T 790.41—2002　采用配电线载波的配电自动化　第 4 部分:数据通信协议第 1 篇:通信系统参考模型 (IEC 61334-4-41:1996,IDT)

IEC 60050.300:2001　国际电工词汇　电气和电子测量方法与测量仪器
第 311 部分:有关测量的基本术语
第 312 部分:有关电气测量的基本术语
第 313 部分:电气测量仪器的类型
第 314 部分:按仪器分类的专用术语

IEC 62051:1999　电气测量　术语表

ANSI C12.19:1997/IEEE 1377:1997　公共事业部门终端设备数据表

3　术语、定义和缩略语

3.1　术语和定义

IEC 60050-300 及 IEC 62051 确立的以及下列术语和定义适用于本部分。

3.1.1

基址名　base_name

与 COSEM 接口对象的第一个属性“Logical_name”(逻辑名)相对应的 short_name(短名)。

3.1.2

类标识码　class_id

接口类的标识码。

3.1.3

COSEM 对象　COSEM object

COSEM 接口类的一个实例。

3.2　缩略语

AARE	Application Association Response	应用连接响应
AARQ	Application Association ReQuest	应用连接请求
ACSE	Application Control Service Element	应用控制服务元件
APDU	Application Protocol Data Unit	应用协议数据单元
ASE	Application Service Element	应用服务元件
A-XDR	Adapted Extended Data Representation	适应扩展数据的表示
COSEM	Companion Specification for Energy Metering	能量计量配套规范
DLMS	Distribution Line Message Specification	配电线报文规范
GMT	Greenwich Mean Time	格林威治标准时间
HLS	High level Security	高级安全
IC	Interface Class	接口类
LLS	Low Level Security	低级安全
LN	Logical Name	逻辑名
LSB	Least Significant Bit	最低有效位
M	Mandatory	强制的，必选的
MSB	Most Significant Bit	最高有效位
O	Optional	可选的
OBIS	OBject Identification System	对象标识系统
PDU	Protocol Data Unit	协议数据单元
SAP	Service Access Point	服务访问点
SN	Short Name	短名

4　基本原理

4.1　概述

本章详细描述构造 COSEM 接口类的基本原理，也简要说明了接口对象（接口类的范例）如何用于通信。遵循这些规范的仪表、支撑工具和其他系统元件能够采用互操作的方式进行相互通信。

对象建模：为制定技术规范，本部分采用对象建模技术，对象是属性和算法的集合。

对象的信息包含在其属性中。属性的值表示对象的特征，它能影响对象的行为特征。所有对象的第一个属性都是“逻辑名（logical_name）”，逻辑名是对象标识的一部分。

每个对象都提供了一些检查或修改属性值的方法。共享公共特征的对象被归纳为接口类，由类标识码（class_id）标识。对一个特定类，公共特征（属性和方法）为所有对象描述。接口类的范例称为 COSEM 对象。

制造商可以用负的数码为任何对象添加私有的方法和属性。

"寄存器"接口类从客户机(中央单元、手持终端)来看,是通过组合通用寄存器(包含测量信息或静态信息)必要的特征模型而形成的。寄存器的内容由属性"逻辑名(logical_name)"标识。逻辑名(logical_name)包含有 OBIS 标识符(见 GB/T 19882.31),寄存器实际内容(动态的)在"数值"属性中。

定义一种特定的仪表实际上意味着定义几种特定的寄存器。在图 1 的范例中,该仪表包含两个寄存器,即"寄存器"类范例化的两个特定 COSEM 对象,这表明赋给不同的属性以特定的数值,尽管"寄存器"类一个范例化的 COSEM 对象为"总的、正向的、有功电量寄存器",而另一个为"总的、正向的、无功电量寄存器"。

图 1 举例说明这些术语:

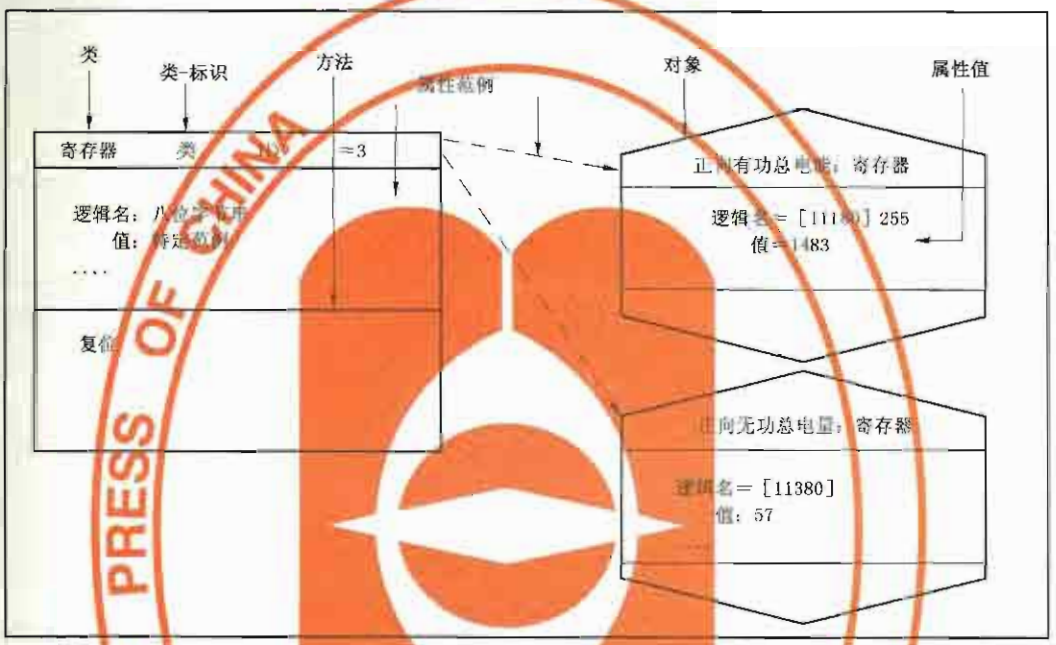

图 1 接口类及其范例

注:从"外部"看来,COSEM 对象(接口类的范例)代表仪表的行为特征。因此,修改属性的值需由外部开始(如复位寄存器的值)。由仪表修改属性的值不在本模型中描述(例如刷新寄存器的值)。

4.2 类的表述方法

本条介绍用于定义接口类的表述方法。

首先,简要介绍类的功能和应用如下。以表格的形式给出类的概貌,表格中包括类名、属性和方法(类说明模板)。

类 名	基本情况	类标识码,版本		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. .....(..) 3. .....(..)	octet-string			
特定方法(如需要)	基本/选用			
1. ..... 2.	....			

每个属性和方法都应详细说明。

类　名	类的说明(如:寄存器,时钟,集...)
基本情况	规定逻辑设备内类的范例号(参见 4.5) value　类范例化的准确次数("value")。 min...max　类范例化的最小次数("min.")和最大次数("max.")。如果最小次数为 0,则类是可选的。反之,如 min.>0,类至少需范例化"min."次。
类标识码	类的标识码(范围:0~65535)。类标识码(class_id)可以从"连接"对象中获得。从 0~8191 的类标识码保留由 DLMS UA 定义;从 8192~32767 的类标识码保留由制造商接口类定义;从 32768~65535 的类标识码保留由用户组的接口类定义。DLMS UA 保留给各制造商或用户组分配范围的权利。
版本	类版本的标识码,此版本可以从"连接"对象中获得。 在一个逻辑设备之内,类的所有范例都应为同一版本。
属性	规定类的属性。 (dyn.)　载有过程值的属性,此属性由仪表自己刷新。 (static)　仪表自身不能更改的属性(例如:配置数据)。
逻辑名	octet-string　逻辑名是类的第一属性,它用来标识类的范例(COSEM 对象)。逻辑名的值与 OBIS 一致(见 GB/T 19882.31)。
数据类型	定义属性的数据类型(参见 4.3)。
最小值	指明属性是否有最小值。 ×　属性有最小值。 < empty >　属性无最小值。
最大值	指明属性是否有最大值。 ×　属性有最大值。 < empty >　属性无最大值。
缺省值	指明属性是否有缺省值。这是属性复位后的值。 ×　属性有缺省值。 < empty >　类没有定义缺省值。
方法	提供属于对象的方法的表。 Method Name ()　此方法应在"方法说明"中说明。
必选/可选	定义方法是必选的还是可选的。 m(mandatory)　方法是必选的。 o(optional)　方法是可选的。

属性说明(Attribute description)

说明属性的数据类型(如果是复杂数据类型)、数据格式和性质(最小值、最大值和缺省值)。

方法说明(Method description)

说明 COSEM 对象的每个方法和执行的行为特征。

注:相关的访问属性或方法的服务在 GB/T 19882.33 中描述。

选择性访问(Selective access)

常用方法 READ/WRITE 和 GET/SET 以整体访问属性为参考典型，也可以提供对属性一部分的选择性访问。属性选择访问部分由特定的选择性访问参数标识，这些选择性访问参数在属性规范中定义。

4.3 公用数据类型

下表包括所有接口类使用的公用数据类型。

简单数据类型	仅载有一个数据项		
integer，long，double-long，unsigned，long-unsigned，double-longunsigned，boolean	简单数据类型在 DL/T 790.41—2002 的 A.12 中定义，数据举例：		
	integer	Integer8	1 字节
	long	Integer16	2 字节
	double-long	Integer32	4 字节
enum(枚举)	需在“属性说明”中定义的枚举类型的元素。 没有值列表的枚举被缺省保留。		
real32，real64	按 DL/T 790.41—2002 的“REAL”规范定义的实数类型。		
visible-string，octet-string	有序的 ASCII 字符由八位字节表示(octet：8 位字节)。		
bit-string(位串)	有序的布尔值序列。		
复杂数据类型	包括多个数据项，或数据项本身不是简单数据类型		
array	数组元素需在“属性说明”部分定义。		
compact array	数组元素需在“属性说明”部分定义。		
structure	结构类型需在“属性说明”部分定义。		
instance specific	属性的数据类型需在特殊仪表(范例模型)对象的范例中规定。		

4.4 表示日期和时间的数据格式

日期和时间常采用八位字节串(octet-string)作为数据类型来表示，但要明确定义其数据格式。

date(日期)	八位字节串 1年 高字节，年 低字节，月，日，星期： 年：数据类型为 unsigned16，范围 0..最大 0xFFFF = 非定义 年的高字节和低字节采用 unsigned 16 的 2 个字节。 月：数据类型为 unsigned8， 范围 1..12，0xFD，0xFE，255 1 为 1 月 0xFD = 夏令时结束 0xFE = 夏令时开始 0xFF = 未规定 日：数据类型为 unsigned8 范围：1..31，0xFD，0xFE，255 0xFD = 当月的倒数第二天 0xFE = 当月的倒数第一天 0xE0 ~ 0xFC = 保留 0xFF = 未规定 星期：数据类型为 unsigned8， 范围：1..7，255 1 为星期一 255 = 未规定 对重复的日期，没有使用的部分应置为“未规定”。

time(时间)	八位字节串（小时，分，秒，1/100 秒） 小时：数据类型为 unsigned8， 范围：0..23，255 255 = 未规定 分：数据类型为 unsigned8 范围：0..59，255 255 = 未规定 秒：数据类型为 unsigned8 范围：0..59，255 255 = 未规定 1/100 秒：数据类型为 unsigned8 范围：0..99，255 255 = 未规定 对复置的时间，没有使用的部分应置为"未规定"。
deviation(偏差)	数据类型为 Integer16， 范围：-720..720， 本地时间与 GMT(格林威治时间)之间的分钟差值。 0x8000 = 未规定
clock_status(时钟状态)	数据类型为 Unsigned8，为 8 位位串， 状态位定义如下， 0(LSB)：数值无效[a] 位 1：数值不确定[b] 位 2：时钟基准不同[c] 位 3：时钟状态无效[d] 位 4：保留 位 5：保留 位 6：保留 位 7：(MSB)夏令时[e]
date_time(日期_时间)	八位字节串 (年高字节 年低字节 月 日 小时 分 秒 1/100 秒 偏差高字节 偏差低字节 时钟状态)
日期_时间的各字段按上述结构编码，按照上述日期_时间的描述，有些可能会被置为"未规定"。	
a 发生事件后时间不能恢复，详情由制造商规定(例如：在时钟的电源被中断后)。 b 发生事件后时间能恢复，但是数值不能被保证，详情由制造商具体规定。 c 如果实际的时钟基准与规定源的时钟基准不同，该比特将会被置位。 d 此位表示至少有一个时钟状态位是无效的，某些位可能是正确的，准确的含义由制造商的文件解释。 e 标志位置为真：发出时间包含夏令时偏差(夏季)；标志位置为假：发出时间不包含夏令时偏差(正常时间)。	

4.5 COSEM 服务器模型

COSEM 服务器为 3 层体系结构，如图 2 所示。

第 1 层：物理设备(Physical device)

第 2 层：逻辑设备(Logical device)

第 3 层：可访问的 COSEM 对象(Accessible COSEM objects)

图 3 举例说明了如何用 COSEM 服务器模型构建一台组合式计量设备。

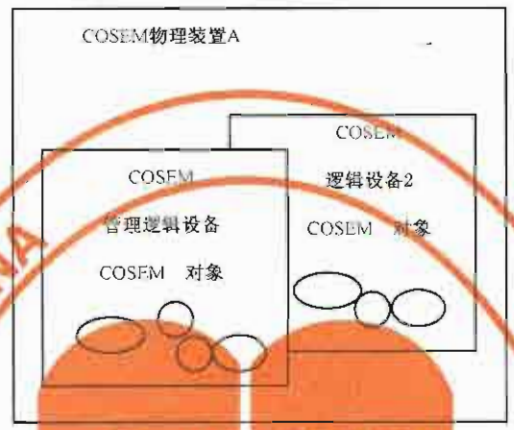

图 2 COSEM 服务器模型

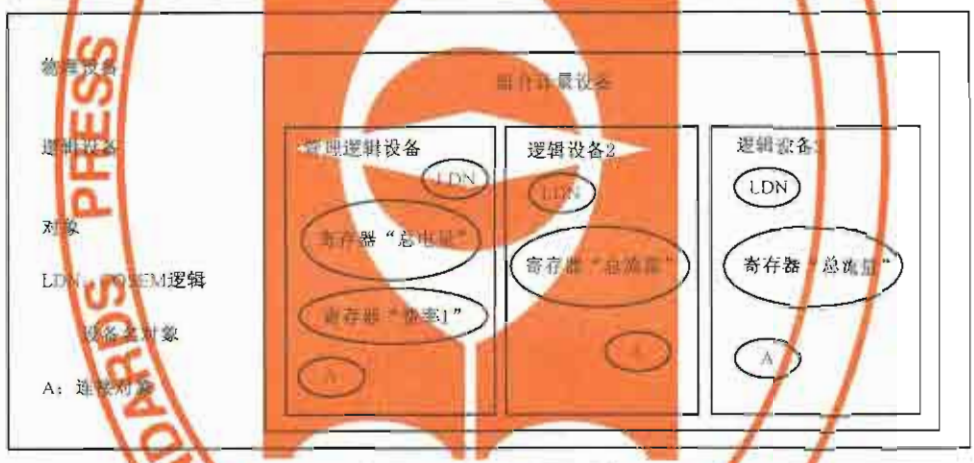

图 3 组合式计量设备

4.6 COSEM 逻辑设备

4.6.1 概述

COSEM 逻辑设备是一组 COSEM 对象。每个物理设备将包含一个"管理逻辑设备"。对 COSEM 逻辑设备的访问由所使用的协议低层的访问表提供。

4.6.2 COSEM 逻辑设备名

COSEM 逻辑设备可由其唯一的 COSEM 逻辑设备名标识。设备名可从"SAP 分配表"接口类的范例(见 5.14)或者 COSEM 对象"COSEM 逻辑设备名"(见 D.1.1.17)中检索。

该设备名由最多 16 个八位字节串定义，前三个八位字节唯一标识设备制造商[1]。制造商负责保证其余八位字节串(最多 13 个八位字节)的唯一性。

4.6.3 逻辑设备的"连接窗口"

为访问服务器端的 COSEM 对象，首先要建立一个应用连接，创建一个已连接的应用可以相互通信的语境。这个语境主要包含：

1) 由 DLMS 用户协会管理。

- 应用语境的信息；
- COSEM 语境的信息；
- 使用身份验证机制的信息；
- 其他。

这些信息包含在一个称之为"连接"对象的特定 COSEM 对象中。本部分定义了两种连接对象，一种是连接使用短名引用的连接对象(association SN)，另一种是使用逻辑名引用的连接对象(association LN)。

服务器可对在客户机与服务器之间建立的应用连接授予不同的访问权限。访问权限涉及一组 COSEM 对象——可视对象，这组对象可以在给定的连接内进行访问——"可看见"。此外，对这组 COSEM 对象的属性与方法的访问也受该连接的限制(例如，某些客户机只能读一个 COSEM 对象的某特定属性)。

客户机通过读取相应的连接对象的"object_list"属性可以获得可见的 COSEM 对象列表("连接窗口")。

利用连接对象所提供的特定方法，在已建立的应用连接内，可以获得对于属性的访问权(只读、只写，读和写)和方法(在已建立的应用连接内)的可用性方面更多的信息(参见 5.12，5.13)。

4.6.4 COSEM 逻辑设备的必选内容

下列对象是每个 COSEM 逻辑设备的必需部分，使用该逻辑设备应用连接中的 GET/READ 方法可以访问这些对象。

- COSEM 逻辑设备名对象；
- 当前连接(LN 或 SN)对象。

4.7 身份验证过程

4.7.1 低级安全(LLS)身份验证

在 GB/T 19882.33 中已说明，ACSE 提供低级安全(LLS)身份验证服务。低级安全身份验证适用于为通信通道提供足够的安全性以避免偷听和信息(密码)重现的情况。

对于低级安全，所有身份验证服务都由 ACSE 提供，连接对象只提供改变"秘密"(如口令)的方法和属性(见 5.12，5.13)。

对于低级安全身份验证，客户机使用其应用层的服务原语 COSEM-OPEN. request 中的"Calling_Authentication_Value"参数，向服务器传送一个"秘密"(如口令)，服务器检查接收到的"秘密"是否与客户机标识相符合，如符合，则客户机身份验证通过，在客户机和服务器之间建立起连接。

4.7.2 高级安全(HLS)身份验证

在 GB/T 19882.33 中已说明，ACSE 提供部分高级身份安全(HLS)验证服务。高级身份安全验证适用于通信通道不能提供内部安全，应采取防范措施以防止偷听和信息(密码)重现的情况。这时，采用 4 步身份验证协议，客户机和服务器采用下列方式进行 4 步身份验证：

第 1 步：客户机向服务器传送"密码"CtoS(例如：随机数)。

第 2 步：服务器向客户机传送"密码"StoC(例如：随机数)。

第 3 步：客户机对 StoC 进行加密处理(例如：用密钥加密)，处理结果 f(StoC)返送给服务器。服务器检查 f(StoC)是否正确，如果正确，服务器认为客户机的身份验证通过。

第 4 步：服务器对客户机的身份验证通过后，服务器对 StoC 进行加密处理(例如：用密钥加密)，处理结果 f(CtoS)返送给客户机。客户机检查 f(CtoS)是否正确，如果正确，客户机认为服务器的身份验证通过。

高级安全(HLS)验证服务的第 1 步由客户机应用层的服务原语 COSEM-OPEN. request 实现。参数"Security_Mechanism_Name"包含 HLS 机制的标识，参数"Calling_ Authentication_Value"包含密码 CtoS。

高级安全(HLS)验证服务的第 2 步由服务器应用层的服务原语 COSEM-OPEN. response 实现。参数"Security_Mechanism_Name"包含 HLS 机制的标识，参数"Responding_Authentication_Value"包含密码 StoC。

第 2 步完成后，连接已正式建立，但是客户机的访问受到当前"连接"对象的方法"reply_to_HLS_authentication"的限制。

第3步和第4步由连接对象的方法 reply_to HLS_authentication 支持(参见5.12、5.13)。如果这两步都成功地完成,则准予当前连接包含所有访问权限,否则,客户机或服务器中断本次连接。

另外,连接对象提供改变 HLS"秘密"的方法(例如:密钥),即 change_HLS_secret。

注:在客户机已经执行 change_HLS_secret()(或 change_LLS_secret())方法后,它等待服务器响应确认密码已经修改,有可能服务器已传送了确认,但是由于通信问题,确认信息并未传达到客户机端。因此,客户机不知道密码是否已经更改。为简单起见,在此种情况下,服务器不提供任何特别支持,也就是说它留给客户机处理这种情况。

5 接口类

目前为仪表所定义的接口类及其相互关系如图4所示。

注1:接口类"base"本身没有明确规定,它只包含了一个属性"逻辑名"。

注2:在"需量寄存器"、"时钟"、"通用表"的说明中,接口类的第二属性与"数据"接口类的第二属性标志不同,也就是说"current_average_value"、"time"和"buffer"与"value"对应。"数值",这是强调"value"的自然特性。

图4 接口类概况

5.1 数据(Data)(class_id: 1)

数据对象存储与内部仪表对象相关的数据，其含义由逻辑名来标识，其数据类型是特定的范例。数据的典型应用是存储组态数据和参数。

数据(Data)	0..n	class_id = 1, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. value	octet-string instance specific			
方法	m/o			

属性说明

Logical_name(逻辑名)	标识包含在数值中的数据，标识符在D.1中规定。
value(数值)	包含数据。 instance specific 数据类型取决于由数值的"逻辑名"定义的范例。

5.2 寄存器(Register)(class_id: 3)

寄存器对象保存过程值或与过程值单元相关的状态值，寄存器对象认知的过程值的自然特性或状态值，数值的特性由"逻辑名"属性用OBIS标识系统(见D.1及GB/T 19882.31)说明。

寄存器(Register)	0..n	class_id = 3, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. value 3. scaler_unit	octet-string instance specific scal_unit_type			
方法	m/o			
1. reset	o			

属性描述

value(数值)	包含当前的过程值或状态值。 特定范例 其值的数据类型取决于由"逻辑名"定义的范例，也可以由制造商规定。因此，当客户机访问此属性时，它应提供属性的值和数据类型。数据类型应是给定的，只有这样，才能和逻辑名一起对数值进行明确的解释。
scaler_unit (换算单位)	包含数值的单位(unit)和换算(scaler)的信息。 如果数值使用复杂数据类型，则换算和单位适用于所有元素。 数据类型为 scal_unit_type，其定义为： scal_unit_type: structure { scaler, unit } scaler: integer 倍数因子的指数(基数为10) 注：如果数值不是数字的，则换算应被置0。 unit: enum 枚举类型定义物理单位：详见下面。

方法说明

reset (data) (复位)	此方法强迫对象复位，通过调用此方法将数值置为缺省值，缺省值是特定范例(instance specific)的参数。 data ::= integer(0)

unit ::= enum

代码	//单位符号		量	单位名称	SI 定义
(1)	a	//	时间	年	
(2)	mo	//	时间	月	
(3)	wk	//	时间	周	7×24×60×60 s
(4)	d	//	时间	日	24×60×60 s
(5)	h	//	时间	小时	60 ×60 s
(6)	min.	//	时间	分	60 s
(7)	s	//	时间(t)	秒	s
(8)	°	//	(相)角	度	rad×180/π
(9)	℃	//	温度(Θ)	摄氏度	K-273.15
(10)	货币		(当地)货币		
(11)	m	//	长度(l)	米	m
(12)	m/s	//	速度(v)		m/s
(13)	m^3	//	体积(V)		m^3
(14)	m^3	//	修正的体积		m^3
(15)	m^3/h	//	流量		$m^3/(60\times60$ s)
(16)	m^3/h	//	修正的流量		$m^3/(60\times60$ s)
(17)	m^3/d	//	流量		$m^3/(24\times60\times60$ s)
(18)	m^3/d	//	修正的流量		$m^3/(24\times60\times60$ s)
(19)	l	//	容积		10^{-3} m^3
(20)	kg	//	质量(m)	千克	kg
(21)	N	//	力(F)	牛顿	N
(22)	Nm	//	能量	牛顿米	J = Nm = Ws
(23)	Pa	//	压力(p)	帕斯卡	N/m^2
(24)	bar	//	压力	巴	10^5 N/m^2
(25)	J	//	能量	焦耳	J = Nm = Ws
(26)	J/h	//	热功		J /(60×60 s)
(27)	W	//	有功功率 (P)	瓦特	W = J/s
(28)	VA	//	视在功率(S)		
(29)	var	//	无功功率(Q)		
(30)	Wh	//	有功能量		W×(60×60 s)
(31)	VAh	//	视在能量		VA×(60×60 s)
(32)	varh	//	无功能量		var×(60×60 s)
(33)	A	//	电流(I)	安培	A
(34)	C	//	电量(Q)	库仑	C = As
(35)	V	//	电压(U)	伏特	V
(36)	V/m	//	电场强度(E)		V/m
(37)	F	//	电容(C)	法拉	C/V = As/V
(38)	Ω	//	电阻(R)	欧姆	Ω = V/A

(39)	$\Omega m^2/m$	//	电阻系数(ρ)		Ωm
(40)	Wb	//	磁通量(Φ)	韦伯	Wb = Vs
(41)	T	//	磁通密度(T)	泰斯拉	Wb/m^2
(42)	A/m	//	磁场强度(H)		A/m
(43)	H	//	电感(L)	亨利	H = Wb/A
(44)	Hz	//	频率	赫兹	1/s
(45)	Rac	//	有功能量表常数		1/(Wh)
(46)	Rre	//	无功能量表常数		
(47)	Rap	//	视在能量表常数		
(48)	V^2/h	//	伏特平方每小时		$V^2(60\times60\ s)$
(49)	A^2/h	//	安培平方每小时		$A^2(60\times60\ s)$
(50)	kg/s	//	质量流量		kg/s
(51)	S mho	//	电导率	西门子	$1/\Omega$
(52)		//	保留		
……			……		
(253)		//	保留		
(254)	其他	//	其他单位		
(255)	计数	//	无单位,缺单位,计数		

数值范例:

数值	换算	单位	数据
263788	−3	m^3	263.788 m^3
593	3	Wh	593 kWh
3 467	0	V	3 467 V

5.3 扩展寄存器(Extended register)(class_id:4)

扩展寄存器类的范例保存过程值及其相关的状态、单位和时间信息。扩展寄存器对象认知此过程值的特性,其特性由属性"逻辑名"用OBIS标识系统来说明。

扩展寄存器(Extended register)		0..n	class_id = 4, version = 0		
属性		数据类型	Min.	Max.	Def
1. logical_name	(static)	octet-string			
2. value	(dyn.)	instance specific			
3. scaler_unit	(static)	scal_unit_type			
4. status	(dyn.)	instance specific			
5. capture_time	(dyn.)	octet-string			
方法		m/o			
1. reset		o			

属性说明

Logic_name...scaler_unit(换算单位)等属性的定义,见"寄存器"类的说明。

Status(状态)	包含扩展寄存器特定的状态信息，数据类型和状态编码由扩展寄存器的范例提供。 Instance specific 缺省值 缺省值取决于状态类型的定义。
capture_time (捕获时间)	包含扩展寄存器特定的日期和时间信息，表明属性"数值"的值是何时被捕获的。 octet-string，格式为 4.4 中的 date_time。

方法说明

reset (data)(复位)	此方法强迫对象复位。通过调用此方法将数值设为缺省值，缺省值是特定范例的常数。 量属性的状态表明此时已调用一个复位方法。 属性捕获时间置为复位执行的时间。 data ::= integer(0)

5.4 需量寄存器(Demand register)(class_id：5)

需量寄存器类的范例保存需量值以及相关的状态、单位和时间信息。需量寄存器周期测量并计算current_average_value(当前平均值)。时间间隔 T 通过测量和计算需量的"number_of_periods(周期数)"和"period(周期)"定义。

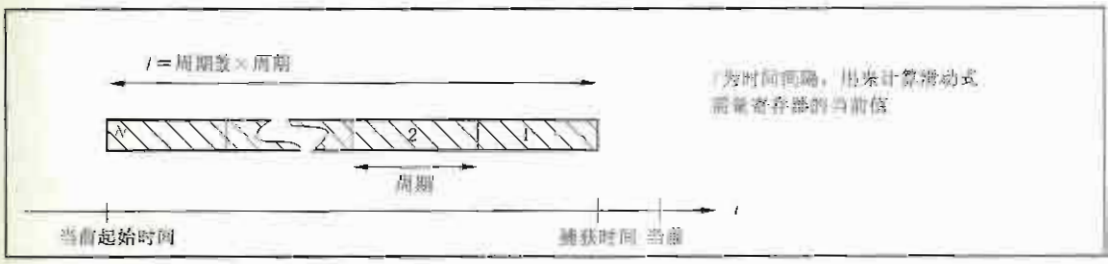

图 5 测量滑动式需量寄存器时的属性

需量寄存器有两种类型：current_average_value(当前平均值)和 last_average_value(最终平均值)(见图 6 和图 7)。

需量寄存器知道它的过程数值的类型，此类型由"逻辑名"用 OBIS 标识系统说明。

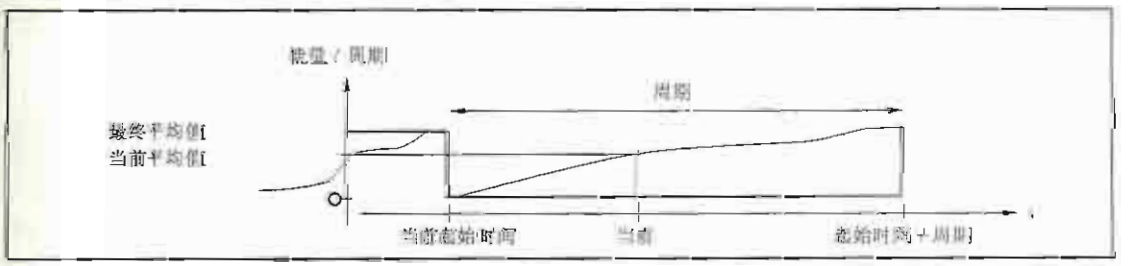

图 6 周期数＝1 时测量当前平均值的属性

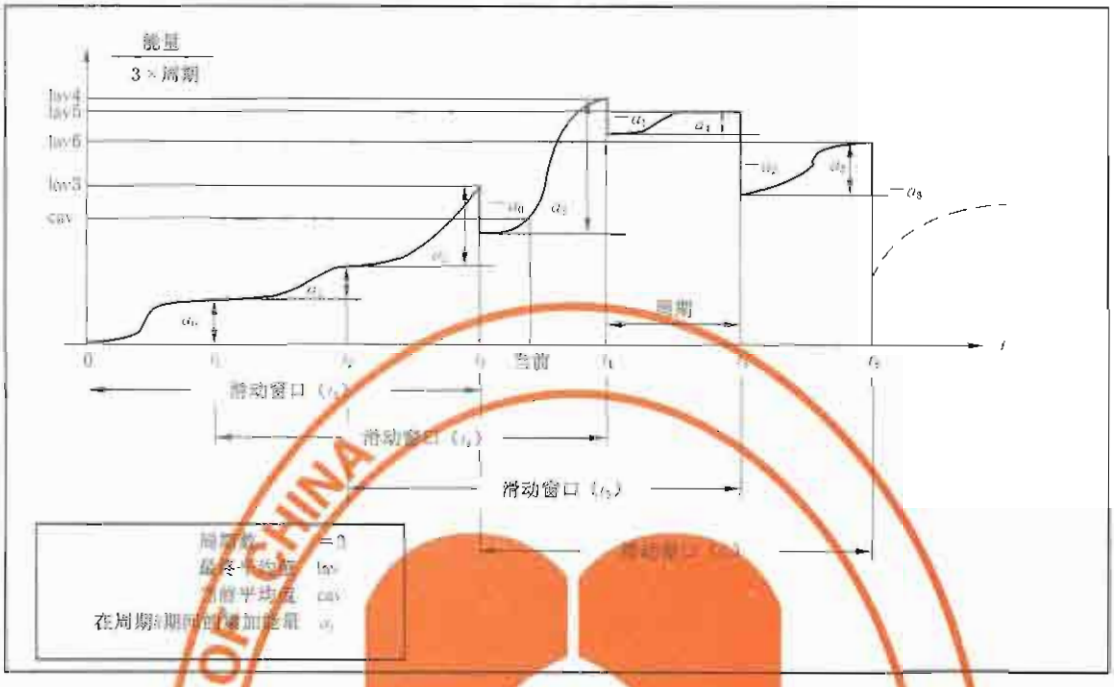

图 7 周期数＝3 时的属性

需量寄存器(Demand register)		class_id = 5, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static)	octet-string			
2. current_average_value (dyn.)	instance specific			0
3. last_average_value (dyn.)	instance specific			0
4. scaler_unit (static)	scal_unit_type			
5. status (dyn.)	instance specific			
6. capture_time (dyn.)	octet-string			
7. start_time_current (dyn.)	octet-string			
8. period (static)	double-long-unsigned	1		
9. number_of_periods (static)	long-unsigned	1		1
方法	m/o			
1. reset				
2. next_period o	o			

属性说明

logical_name，scaler_unit 等属性的定义见"寄存器"类说明。

status(状态)	包含需量寄存器相关的状态信息。应为每个需量寄存器的范例提供状态的类型和编码。 范例定义 缺省值　　缺省值取决于状态类型的定义。
current_average_value（当前平均值）	当前值(运行需量)是从 start_time 时间开始的累计值除以 number_of_periods * period(周期数 * 周期)的电能量。 该值仅用于简单数据类型。

last_average_value (最终平均值)	上一需量周期累计值除以 number_of_periods×period(周期数×周期)的电能值。 当前周期的能量(没有结束)在计算中不考虑。 该值仅用于简单数据类型。
capture_time (捕获时间)	包含最终平均值计算的日期和时间。 8 位字节串,格式见 4.4 中的 date_time 组。
start_time_current (当前开始时间)	包含当前平均值开始测量时的日期和时间。 octet-string,格式见 4.4 中的 date_time 组。
period(周期)	指两个连续的最终平均值更新的时间间隔(number_of_ periods×period 是需量计算公式中的分母)。 数据类型为 double-long-unsigned,测量周期以秒为单位。 该属性重新写入新数值后的仪表行为特征应由制造商规定。
number_of_periods (周期数)	周期数用来计算最终平均值,number_of_periods >= 1。数据类型为 long-unsigned。周期数>1,表明最终平均值为"滑动式需量"。周期数=1,表明最终平均值代表"块式需量"。 该属性重新写入新数值后的仪表行为特征应由制造商规定。

方法说明

reset (data)(复位)	为使测量对象复位,调用该方法启动下列动作: 终止当前周期; current_average_value 和 last_average_value 置为缺省值; capture_time 和 start_time_current 置为 reset(data)执行时间; data 不返回(= integer(0))
next_period (data) (下一个周期)	该方法用触发器定期终止前一个周期(重新启动),关闭(终止)当前的测量周期; 使用 capture_time 和 start_time_current_average_value 的复制给 last_average_value,将当前平均值设为缺省值,启动下一个测量周期。 注:旧的 last_average_value 和 capture_time 可以在时间"周期"内读出。但是由于 current_average_value 及其前后 last_average_value 时不再可用。 data [1] = integer(0)

5.5 寄存器激活(Register activation)(class_id: 6)

寄存器激活类的范例展示处理不同的费率结构。如果规定的激活掩码是激活的(active_mask),则对应的寄存器、扩展寄存器和需量寄存器对象是使能的。在寄存器分配表中没有 active_mask 的所有其他寄存器对象是非使能的。在任何寄存器分配表中,所有寄存器对象缺省的无定义是使能的。

寄存器激活(Register activation)		0..n	class_id = 6, version = 0		
属性		数据类型	最小值	最大值	缺省值
1. logical_name	(static)	octet-string			
2. register_assignment	(static)	array			
3. mask_list (掩码列表)	(static)	array			
4. active_mask (激活掩码)	(dyn.)	octet-string			
方法		m/o			
1. add_register		o			
2. add_mask		o			
3. delete_mask		o			

属性说明

logical_name (逻辑名)	包含在 COSEM 对象中定义的掩码组(代表费率)的标识符,正常情况下,此标识符与定义在寄存器分配中的寄存器特性有联系(如:能量寄存器,需量寄存器……)
register_assignment (寄存器分配表)	规定分配给本寄存器激活对象的 COSEM 对象的有序列表,此列表可同时包含各种不同的 COSEM 对象(如:寄存器、扩展寄存器和需量寄存器等)。 array object_definition object_definition ::= structure (class_id: long-unsigned; logical_name: octet-string (SIZE(6)))
mask_list (掩码列表)	规定一个寄存器激活掩码的列表,每个入口(掩码)都由其 mask_name(掩码名)标识,并包含一个指向掩码分配寄存器的索引数组(在寄存器分配表中的第一个对象索引号为 1,第二个对象的索引号为 2,...)。 Array register_act_mask register_act_mask ::= structure (mask_name: octet-string; index_list: index_array) index_array ::= array unsigned 定义在对象内的 mask_name(掩码名)应是唯一的。
active_mask (激活掩码)	定义当前的激活掩码,此掩码由其 mask_name(掩码名)(参见 mask_list)定义,数据类型为 octet-string,它是 mask_list 中的一个 mask_name。 active_mask 定义当前寄存器的使能,而在寄存器分配表中的所有其他的寄存器是非使能的。

方法说明

add_register(data) (附加寄存器)	本方法用于再增加一个寄存器给寄存器分配表属性。新添加的寄存器放在数组末尾,也就是说,新添加寄存器的索引号最大,原有寄存器的索引号不变。 data ::= structure (class_id: long-unsigned logical_name: octet-string)
add_mask (data) (附加掩码)	本方法用于再添加一个掩码给属性掩码列表。如果掩码列表中已有一个掩码与之同名,原有的掩码被新的掩码取代。 data ::= register_act_mask(同上)
delete_mask(data) (删除掩码)	本方法用于从掩码列表属性中删除一个掩码,此掩码由其掩码名定义。 data ::= octet-string (mask_name)

5.6 通用集(Profile generic)(class_id: 7)

通用集类定义了捕获对象存储动态过程值的广义概念。捕获对象可以是寄存器,也可以是时钟或集。捕获对象的过程值可以定期采集,也可以随机采集。集有一个缓冲区用于存储捕获到的数据。为了检索该缓冲区的部分数据,可以规定一个数值范围或一个记录范围,要求检索的所有记录数值或记录号落在给定范围内。

通过把相应的对象分配给集，可以规定捕获对象应保留的数值（用捕获方法）。集的缓冲区入口的结构是统一的（所有的入口的大小和结构相同），分配表的定义是静态的。更改捕获对象分配表将导致集的缓冲区被完全清除，而且所有把该集作为捕获对象的集也将被清除，以保证其记录的统一。

缓冲区中的记录可以根据其中某一个寄存器或时钟来排序，也可以采用"后进先出"的堆栈顺序存放。例如，采用记录深度为一的一个有序捕获集，按需量寄存器排序就可以很容易地构造一个"最大量需量寄存器"。定义一个保留某一周期内三个最大值的集也同样简单。

集的数据大小由下列三个参数来决定：

a) 填写记录号。清除集以后该参数归零。

b) 需保留的最大入口数。如果缓冲区的所有入口已写满又发生一次 capture() 请求，则最不重要的入口（依据所请求的排序方法）将被丢失。最大的入口数可以被规定；一旦改变本参数，缓冲区将进行调整。

c) 缓冲区的物理限制。这种限制主要取决于要捕获的对象。该对象将拒绝企图超出物理限制的最大入口数的设置。

通用集（Profile generic）		0. . n	class_id = 7，version = 1		
属性		数据类型	最小值	最大值	缺省值
1. logical_name	（static）	octet-string			
2. buffer	（dyn.）	compact array or array			×
3. capture_objects	（static）	array			
4. capture_period	（static）	double-long-unsigned			×
5. sort_method	（static）	enum			×
6. sort_object	（static）	object_definition			×
7. entries_in_use	（dyn.）	double-long-unsigned	0		0
8. profile_entries	（static）	double-long-unsigned	1		1
方法		m/o			
1. reset		o			
2. capture		o			
3. reserved from previous versions		o			
4. reserved from previous versions		o			

属性说明

buffer（缓冲区）	缓冲区属性包含一组按一定顺序保存的入口，每个入口包含捕获对象的值（这些值是 GET 或 READ. request 的返回值）。结构（见下文）内捕获对象值的顺序与在属性 capture_objects 中定义的顺序一致。数组（见下文）内入口的顺序取决于规定的排序方法。缓冲区通过调用方法 capture（）填写。 compact-array or array　　entry entry ::=　　structure 　　instance specific 缺省值：　　复位后缓冲区为空。 注 1：读整个缓冲区时，仅传送"正在使用中（in use）"的入口。 注 2：如果捕获对象的值可由先前的值确定，则可以由"空数据（nulldata）"替代。（例如，如果时间可用先前的时间和捕获周期（capture_period）来计算；或者值与前面的值相等。） 选择性访问（selective access）（见 4.2）可适用于缓冲区属性（任选项）。选择性访问参数在下文中定义。

capture_objects
(捕获对象)
	用于规定分配给集的对象的捕获对象列表(寄存器、时钟和集)。即调用方法 capture ()时,这些对象的规定属性将被复制到集的缓冲区中。
	array capture_object_definition
	capture_object_definition :: = structure
	{
	class_id: long-unsigned;
	logical_name: octet-string;
	attribute_index: integer;
	data_index: long-unsigned
	}
	其中,attribute_index 是指向对象内属性的指针,attribute_index 1 指向第一个属性(即 logical_name),attribute_index 2 指向第二个属性,依此类推。attribute_index 0 指向所有公共属性。
	data_index 是选择属性特定元素的指针,data_index 1 指向属性结构的第一个元素。如果属性不是一个结构,则 data_index 没有意义。如果捕获对象是一个集的缓冲区,则 data_index 用于标识内容的集的缓冲区(即列)的捕获对象。Data_index 0 指向整个属性。

capture_period (捕获周期)	≥1:自动捕获,规定捕获周期(单位:s); 0:非自动捕获,由外部条件或事件发生的捕获事件触发。
sort_method (排序方法)	如果集不需排序,其缓冲区以"先进先出"的队列方式工作(实际上它按捕获时机排序,不需按时钟对象内的时间排序)。 如果缓冲区已满,下一次调用 capture ()方法时将把缓冲区中的第一个(最先的)入口挤出,为最新的入口腾出空间。如果集需要排序,调用 capture ()方法将新的入口存放在缓冲区的适当位置,并使移后面的所有入口,无用的入口可能被丢失。如果新的入口在最后一个入口写入缓冲区后需要写入,而且该缓冲区已满,则新的入口将不再保留。 enum 1) fifo (first in first out), 2) lifo (last in first out), 3) largest, 4) smallest, 5) nearest_to_zero, 6) farest_from_zero 缺省值: fifo
sort_object (排序对象)	如果集需要排序,则此属性规定作为排序依据的寄存器或时钟。 capture_object_definition 见上文。 缺省值: 没有作为排序依据的对象(只有 fifo 或 lifo 的 sort_method 没有作为排序依据的对象)。
entries_in_use (当前入口数)	保存在缓冲区中的入口数。调用方法 reset ()后,缓冲区中不包含任何入口,此时入口数为零。在每次调用方法 capture ()后,入口数将会加 1,直到入口数等于最大入口数(详见 profile_entries)。 double-long-unsigned 0...profile_entries 缺省值: 0
profile_entries (最大入口数)	规定要保存在缓冲区中的最大入口数。 double-long-unsigned 1...(上限受物理空间限制) 缺省值 1

对 buffer 属性进行选择访问的参数

访问选择器的值	参　数	注　释
1	范围描述器(range_descriptor)	该情况应只返回与范围描述器对应的缓冲区元素。
2	入口描述器(entry_descriptor)	该情况应只返回与入口描述器对应的缓冲区元素。

range_descriptor ::= structure

{

restricting_object	capture_object_definition	定义限定检索范围的寄存器或时钟
from_value	instance_specific	要检索的最先或最小的入口
to_value	instance_specific	要检索的最新或最大的入口
selected_values	array capture_object_definition	要检索的列的列表。如果数组为空(没有入口),所有捕获的数据被返回。否则,仅数组中规定列的数据被返回。capture_object_definition 已在上述 capture_objects 中定义。

}

entry_descriptor ::= structure

{

from_entry	double-long-unsigned	要检索的第一个入口。
to_entry	double-long-unsigned	要检索的最后一个入口。 to_entry = 0 表明可能达到的最大入口。
from_selected_value	long-unsigned	要检索的第一个值的索引号。
to_selected_value	long-unsigned	要检索的最后一个值的索引号。 to_selected_value = 0 表明可能达到的最大索引号。

}

方法说明

reset (data)) (复位)	清集的缓冲区,调用本方法后缓冲区将没有有效的入口,entries in use 归零,调用本方法不触发对捕获对象的任何附加操作,特别是它不会复位任何捕获缓冲区或寄存器。 data ::= integer(0)
capture (data) (捕获)	读取每个捕获对象的 object_attribute,并将捕获对象的值复制到缓冲区。根据 sort_method 和缓冲器的实际状态,调用本方法将生成一个新的入口或替代已经没有意义的入口。只要缓冲区的入口不满,entries_in_use 属性就会加 1。 与方法 reset ()一样,调用本方法不触发对捕获对象的任何附加操作,需要说明的是,仅有捕获对象的某些属性可能被存储,而不是全部对象。 data ::= integer(0)

某些属性更改后的对象行为特征

	对描述缓冲区静态结构的任何一个属性的任何一次更改都会自动调用一次方法 reset (),而且这次调用又会传播给所有捕获这个集的其他集。如果企图给 profile_entries 写入一个对缓冲区来说太大的值,将被拒绝。

限制

定义捕获对象时，应当避免循环(递归)引用。

用于定义优先读出数值子集的集

将 profile_entries 置为 1，则集的对象可用来定义一组优先读出值(readout-values)。在属性 capture_ objects 中，可以将这些对象和属性预先定义为用一条简单指令就可读出的。

将 capture_period 置为 1，可确保该值每秒更新一次。

5.7 时钟(Clock)(class_id：8)

时钟接口类的一个范例是处理所有与日期和时间有关的信息，包括闰年和本地时间与标准时间(格林威治时间(GMT)的时间偏差。本地时间与标准时间的偏差因季节变化而变化(例如夏季时间和冬季时间)。与外部客户机的接口基于以用年、月、日来表示的日期信息，用时、分、秒和百分之一秒表示的时间信息，以及当地时间与标准时间的偏差。

时钟接口类还可以用这种方式处理夏令时的功能，也就是说，它可依据这些属性调整当地时间与格林威治时间(GMT)的偏差值。夏令时起始时间和结束时间通常是一次设置的，内部算法依据这些设置来计算实际的切换时间。

通用化的时间概念可用图 8 表示。

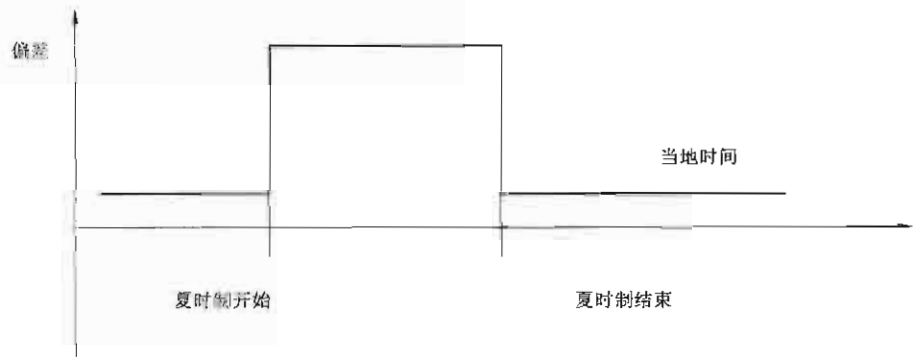

图 8 通用化的时间概念

时钟(Clock)	0..1	class_id = 8, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static)	octet-string			
2. time (dyn.)	octet-string			
3. time_zone (static)	long			
4. status (dyn.)	unsigned8			
5. daylight_savings_begin (static)	octet-string			
6. daylight_savings_end (static)	octet-string			
7. daylight_savings_deviation (static)	integer			
8. daylight_savings_enabled (static)	boolean			
9. clock_base (static)	enum			
方法	m/o			
1. adjust_time	o			
2. adjust_to_measuring_period	o			
3. adjust_to_minute	o			
4. adjust_to_preset_time	o			
5. preset_adjusting_time	o			
6. shift_time	o			

属性说明

time(时间)	包含仪表的当地日期和时间，当地时间与 GMT 的时间差和状态。见 4.4 的说明。置日期和时间时，仅指定的 date_time 的字段被改变。例如：置日期字段时不改变时间字段的内容，date_time 中所有与时间相关的八位字节都应当置为"未规定"，在写时间时 clock_status 被置位。 octet-string，格式见 4.4 中 date_time 的说明。
time_zone(时区)	本地正常时间与 GMT 时间偏差为分钟。 long。
status(状态)	此状态与从读 time 时得到的状态相同。也可见 4.4 的说明。 unsigned8，格式见 4.4 中 clock_status 的说明。
daylight_savings_begin (夏令时开始时间)	定义切换至夏令时的本地日期和时间。作为通用的定义时，允许使用通配符。 octet-string，格式见 4.4 中 date_time 的说明。
daylight_savings_end (夏令时结束时间)	同上。 octet-string，格式见 4.4 中 date_time 的说明。
daylight_savings_deviation (夏令时偏差)	包含正常时间调整到夏令时的分钟数： integer　　时差范围：达±120 min。
daylight_savings_enabled (夏令时使能)	以 TRUE 启动夏令时功能。 boolean。
clock_base(时钟基准)	定义基本定时信息的来源。 Enum(列举)　　0) 未定义 1)　　内部晶振 2)　　工频 50 Hz 3)　　工频 60 Hz 4)　　GPS 5)　　无线控制

方法说明

adjust_time (data) (调整时间)	将仪表时间置为最接近的(+/−)1 h 内时刻值(*:00、*:15、*:30、*:45)。 data ::=　　integer (0)
adjust_to_measuring_period (data) (调整测量周期)	将仪表的时间置为最接近的一个测量周期的起始点。 data ::= integer (0)
adjust_to_minute (data) (调整分)	将仪表的时间置为最接近的分钟值。 若 second-counter<30 s，则 second-counter 置为 0。 若 second-counter≥30 s，则 second-counter 置为 0，且根据需要，将 minute_counter 和所有相关时钟的值加 1。 data ::=　　integer(0)
adjust_to_preset_time (data) (调整预置时间)	本方法与 preset_adjusting_time 方法一起使用。如果仪表位于 validity_interval_start 和 validity_interval_end 之间，则把时间置为 preset_time。 data ::=　　integer(0)

preset_adjusting_time (data)
(预置调整时间)
	将时间预置为一个新值(preset_time)，同时定义一个激活新时间的有效时间间隔(validity_interval)。
	data ::=　　structure
	{
	　preset_time: octet-string;
	　validity_interval_start: octet-string;
	　validity_interval_end: octet-string
	}
	各八位字节串的格式见 4.4 中 date_time 的说明。

shift_time (data) (偏移时间)	将时间偏移 n s ($-900 \leqslant n \leqslant 900$)。 data ::= long

5.8 脚本表 (Script table) (class_id: 9)

脚本表接口类提供了通过激发一个执行方法来触发一系列操作的可能性。为达到这个目的,脚本表包含了一张脚本入口表,每个入口(脚本)表包含一个脚本标识符(script_identifier)和一系列操作说明(action_specification)。一个操作说明触发逻辑设备中一个 COSEM 对象的方法或修改一个 COSEM 对象的属性。

一个特定的脚本可以由同一逻辑设备内的其他 COSEM 对象触发,也可以由外部事件触发。

若两个脚本需要同时执行,则索引号小的首先执行。

脚本表(Script table)	0, n	class_id = 9, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. scripts (static)	octet-string array			
具体方法	m/o			
1. execute	m			

属性说明

Scripts(脚本)	规定不同的脚本,即操作列表。
	array script
	script structure
	{
	script_identifier: long-unsigned
	actions: array action_specification
	}
	脚本标识符 0(script_identifier 0)保留。若规定一个执行方法的标识符(script_identifier)为 0,将产生一个空脚本(无任何可执行的操作)。
	action_specification structure
	{
	service_id: enum
	class_id: long-unsigned
	logical_name: octet-string
	index: integer
	parameter: service specific
	}
	其中:
	service_id:定义应作用于引用对象的操作。
	1) 写属性
	2) 执行特定的方法
	index:定义(若 service_id 为 1)所选对象的哪一个属性要受到影响,或定义(若 service_id 为 2)哪一个方法要被执行。第一个属性(logical_name)的索引号为 1。第一个方法的索引号也为 1。
	注:操作说明(action_specification)只限于激活不产生任何响应的方法(从服务器到客户机)。

方法说明

execute (data) (执行)	执行参数 data 规定的脚本。 data　　long-unsigned 若 data 与脚本表中的任何一个 script_identifier 相匹配，则执行相应的 action_specification。

5.9 时间表(Schedule)(class_id：10)

时间表接口类和特定日期表接口类的一个对象负责处理设备内驾驭活动的时间和日期。下表给予概括说明并显示它们之间的相互影响。

时间表

索引	使能	操作(脚本)	switch_time	validity_window	exec_weekdays							exec_specdays				日期范围	
					Mo	Tu	We	Th	Fr	Sa	Su	S1	S2	S8	S9	begin_date	end_date
120	是	xxxx:yy	06:00	15	×	×	×	×	×	×						01-04-××	30-09-××
121	是	xxxx:yy	22:00	15	×	×	×	×	×							01-04-××	30-09-××
122	是	xxxx:yy	12:00	0						×						01-04-××	30-09-××
200	否	xxxx:yy	06:30			×	×	×	×	×						01-04-××	30-09-××
201	否	xxxx:yy	23:00		×	×	×	×	×							01-04-××	30-09-××
202	否	xxxx:yy								×						01-04-××	30-09-××

特殊日期表

索引	special_day_date(特殊日期表)	day_id(星期标识)
12	24-12-××	S1
33	25-12-××	S3
77	31-03-97	S3

时间表(Schedule)	0.0.n	class_id = 10, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. entries (static)	octet-string array			
方法	m/o			
1. enable/disable 2. insert 3. delete	o o o			

属性说明

entry(入口)	规定了在给定时间要执行的脚本，每个入口只有一个脚本能被执行。
	array schedule_table_entry
	schedule_table_entry structure
	{
	index: long-unsigned (1..9999)
	enable: Boolean
	script_logical_name: octet-string
	script_selector long-unsigned
	switch_time: octet-string
	validity_window: long-unsigned
	exec_weekdays: bit-string
	exec_specdays: bit-string
	begin_date: octet-string
	end_date: octet-string
	}
	script_logical_name:定义脚本对象的逻辑名。
	script_selector:定义了被执行的脚本 script_ identifier(脚本_标识)。
	switch_time:用通配符定义重复入口。
	octet-string 的格式按 4.4 中 time 的说明。
	validity_window 定义一个以分钟为单位的周期。在此周期内，在电源失效后确保一个入口被执行。(时间介于所定义的切换时间和实际加电时间之间。)
	0xFFFF:脚本应在任何时间都执行。
	exec_specdays 实现连接 IC 特定日期表，日期-ID。
	begin_date 和 end_date 定义了入口有效的日期周期(允许使用通配符)。格式按 4.4 中 date 的说明。

方法说明

enable/disable (data) (使能/禁止(数据))	置范围 A 入口的禁止位为真，同时使范围 B 入口使能。 data ::= structure { firstIndexA: lastIndexA: firstIndexB: lastIndexB firstIndexA 范围 A 的第一个索引是禁止的。 long-unsigned lastIndexA 范围 A 的最后一个索引是禁止的。 long-unsigned firstIndexB 范围 B 的第一个索引是使能的。 long-unsigned lastIndexB 范围 B 的最后一个索引是使能的。 long-unsigned。 firstIndexA/B<lastIndexA/B: 范围 A/B 所有入口都是禁止/使能的。 firstIndexA/B = = lastIndexA/B:一个入口是禁止/使能的。 firstIndexA/B>lastIndexA/B: 没有禁止/使能。 firstIndexA/B 及 lastIndexA/B > 9999:没有禁止/使能的入口。

insert (data)(插入)	在表中插入一个新的入口。如果该入口的索引已存在则以先的入口被新的入口所覆盖。 entry　　schedule_table_entry data corresponding to entry(数据对应于入口)

delete (data)(删除)	删除表中规定范围内的入口。
	data ::= structure
	{
	firstIndex;
	lastIndex
	}
	firstIndex　　范围的第一个索引被删除。
	long-unsigned
	lastIndex　　范围的最后一个索引被删除。
	long-unsigned
	firstIndex < lastIndex:删除给定范围内的所有入口。
	firstIndex ::= lastIndex:删除一个入口。
	firstIndex > lastIndex:没有禁止。

关于表中入口"不一致"的注:

若同一个脚本需要在一个特定的时间执行数次,则它只会执行一次。

若不同的脚本需要在同一时间执行,则执行顺序对应于"索引"顺序,最小索引的脚本首先执行。

电源故障后的恢复

电源故障后,整个时间表用于执行所有在电源故障中丢失的必要的脚本。为此,应检测在电源故障中未执行的入口。依据有效窗口属性,它们在正确的顺序下得到执行(正如它们在正常的操作中执行一样)。

处理时间变化

这里有四种不同的时间变化"操作":

a)　时间向前置;

b)　时间向后置;

c)　时间同步;

d)　夏时制。

所有这4种操作都需要按时间表与时间设置的行为,相互作用而执行不同的处理。

时间向前置

与电源故障时的处理方式相同,所有丢失的入口依据有效窗口属性执行。(制造商明确定义的)短时间置可以按时间同步处理。

注:写入"时钟"对象的"时间"属性。

时间向后置

这导致那些在重复的时间内被激活的入口重复出现。(制造商明确定义的)短时间置可以按时间同步处理。

注:写入"时钟"对象的"时间"属性。

时间同步

时间同步用来校准主时钟与本地时钟的微小偏差,算法由制造商给定,它将确保时间表的任何一个入口不会丢失或执行两次。有效窗口属性不受影响,因为所有入口应在正常操作下执行。

注:用"时钟"对象的 adjust_time 方法。

夏时制

若时钟向前置，则所有在向前置时间间隔(以及因此而丢失)的脚本将被执行。

若时钟向后置，则取消在向后置时间间隔内要重复执行的脚本。

5.10 特殊日期表(Special days table)(class_id：11)

本接口类允许定义日期，对于特殊日期将不考虑正常转换行为。接口类与"时间表"或"活动日历"类共同起作用，链接的数据项是 day_id。

特殊日期表(Special days table)	0..1	class_id = 11, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. entries (static)	octet-string array			
方法	m/o			
1. insert 2. delete	o o			

属性说明

entry(入口)	为给定日期规定一个特殊日期的日时。
	如圣诞节这样重复的特殊日期可用通配符。
	array spec_day_entry
	spec_day_entry ::= structure
	{
	index: long-unsigned
	specialday_date: octet-string
	day_id: unsigned
	}
	specialday_date 的格式按 4.4 中日期的说明。day_id 的范围应与相关的"时间表"接口类对象中的位串 exec_specdays 的长度匹配。

方法说明

insert (data)(插入)	在表中插入一个新的入口。 entry spec_day_entry 若有相同索引或相同日期的特殊日期作为日定义的日期被插入，则旧的入口将被覆盖。
delete (data)(删除)	删除表中的一个入口 index 应被删除的入口的索引 long-unsigned data ::= long-unsigned

5.11 活动日历(Activity calendar)(class_id：20)

活动日历类的一个范例是处理不同费率结构。它是仪表内预设操作的定义，并遵循以季节、星期等定义的时间表为基础的日历的传统形式。它可与更通用的时间表对象并存并可与之互换。若操作被安排在时间表对象和活动日历对象中的同一时间，则由时间表触发的操作首先执行。

在电源故障后，只有从对象活动日历中丢失的"最后操作"才会被执行(延迟)，这将确保电源启动后正确计费。若存在一个时间表对象，则活动日历丢失的"最后操作"应按时间表请求操作的顺序在恰当的时间执行。

活动日历定义了特定脚本的激活，以使得完成逻辑设备中不同的活动。脚本对象的接口和对象时间表接口相同(参见 5.9)。

如果"特殊日期表"接口类(参见 5.10)的范例可用，则相关的入口优先于有日历表选择驱动的活动日历对象。在"特殊日期表"中引用的日历表激活"活动日历"对象中 day_profile_table 的 day_schedule，"活动日历"对象中的 day_profile_table 通过 day_id 引用。

活动日历(Activity calendar)		0..1	class_id = 20，version = 0		
属性		数据类型	最小值	最大值	缺省值
1. logical_name	(static)	octet-string			
2. calendar_name_active	(static)	octet-string			
3. season_profile_active	(static)	array			
4. week_profile_table_active	(static)	array			
5. day_profile_table_active	(static)	array			
6. calendar_name_passive	(static)	octet-string			
7. season_profile_passive	(static)	array			
8. week_profile_table_passive	(static)	array			
9. day_profile_table_passive	(static)	array			
10. activate_passive_calendar_time	(static)	octet-string			
方法		操作			
1. activate_passive_calendar		1			

属性说明

被称为..._active 的属性当前是激活的，被称为..._passive 的属性通过方法 activate_passive_ calendar 激活。

calendar_name (日历名)	包含一个标识符(主要用于说明一组待对象激活的脚本)。

season_profile
(季历表)
	包含一个定义季节起始日期的列表，该列表按 season_start 排序，每个季节激活一个特定的 week_profile。
	array season
	season ::= structure
	{
	season_profile_name: octet-string
	season_start: octet-string
	week_name: octet-string
	}
	其中，
	season_profile_name：由用户定义的当前 season_profile 的名称。
	season_start：定义季节开始的时间，格式见 4.4 中 date_ time 的说明。
	注：当前季节由下一个季节的 season_start 终止。
	week_name：定义由本季节激活的 week_profile。

week_profile_table
(周历表)
	包含不同季节使用的一周中每一天与 day_profile 之间关系的数组。
	array week_profile
	week_profile ::= structure
	{
	week_profile_name: octet-string;
	monday: day_id;
	tuesday: day_id;
	wednesday: day_id;
	thursday: day_id;
	friday: day_id;
	saturday: day_id;
	sunday: day_id
	}
	day_id: unsigned
	其中
	week_profile_name 是由用户定义的标识当前 week_profile 的名称。
	Monday 定义星期一在 day_profile 中有效的日期。
	……
	Sunday 定义星期日在 day_profile 中有效的日期。

day_profile_table
(日历表)
	包含一个脚本列表和相应的活动时间,列表对每个 day_profile 按 start_time 排序。
	array day_profile
	day_profile ::= structure
	{
	day_id: unsigned;
	day_schedule: array
	(day_profile_action)
	}
	day_profile_action ::= structure
	{
	start_time: octet-string;
	script_logical_name: octet-string;
	script_selector: long-unsigned
	}
	其中:
	day_id:用户定义的当前 day_profile 的标识符。
	start_time:定义脚本执行的时间(不能使用通配符)。格式见 4.4 中 time 的说明。
	script_logical_name:定义脚本对象的逻辑名。
	script_selector:定义要执行的脚本的 script_ identifier。

activate_passive_ calendar_time(激活非活动日历时间)	定义对象自身调用方法 activate_passive_calendar 的时间。 各属性域都置为"未规定"将导致自动激活机制失效;也不允许将部分 date 和 time 字段置为"未确定"。 octet-string,格式见 4.4 中 date_time 的说明。

方法说明

activate_passive_calendar(data) (激活被动日历)	本方法用于将所有称为..._passive 的属性复制到对应的称为..._active 的属性中。 data ::= integer(0)

5.12 连接 LN (Association LN) (class_id: 15)

COSEM 逻辑设备能够在一个 COSEM 语境中用逻辑名引用建立应用连接，这种情况下，COSEM 逻辑设备通过 association LN 类的范例来建立连接。对设备所能支持的每个连接，每个 COSEM 逻辑设备有一个本接口类的范例。

连接 LN (Association LN)	0....MaxNbofAss	class_id = 15, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. object_list (static) 3. associated_partners_id 4. application_context_name 5. xDLMS_context_info 6. authentication_mechanism_name 7. LLS_secret 8. association status	octet-string object_list_type associated_partners_type application-context-name xDLMS-context-type mechanism-name octet-string enum			
方法	m/o 必选/可选			
1. reply_to_HLS_authentication 2. change_HLS_secret 3. add_object 4. remove_object	o o o o			

属性说明

logical_name (逻辑名)	标识连接 LN 对象的范例。在建立应用连接的过程中，该属性的值用于指示客户机的应用程序。

object_list
(对象列表)
	包含所有 COSEM 对象范例的列表和访问权描述符。这些 COSEM 对象在给定的连接中是"可见的"(可以被访问的)。
	object_list_type ::= array object_list_element
	object_list_element ::= structure
	{
	class_id: long-unsigned;
	version: unsigned;
	logical_name: octet-string;
	access_rights: access_right;
	}
	access_right ::= structure
	{

	attribute_access: attribute_access_descriptor;
	method_access: method_access_descriptor;
	}
	attribute_access_descriptor ::= array attribute_access_item
	attribute_access_item ::= structure
	{
	attribute_id : integer;
	access_mode : enum
	{
	no_access (0);
	read_only (1) ;
	write_only (2);
	read_and_write (3);
	}
	access_selectors : array integer OPTIONAL;
	}
	method_access_descriptor ::= array method_access_item
	method_access_item ::= structure
	{
	method_id : integer;
	access_mode : boolean;
	}
	attribute_access_descriptor 和 method_access_descriptor 包含所有实现的属性和方法。
	access_selectors 包含一组支持的选择器值的列表。
	选择性访问(selective access)(见 4.3.2)对属性 object_list 来说是可用的(可选的)。选择性访问参数在下文中定义。

associated_partners_id(连接对标识)	包含建立连接的 COSEM 服务器连接和客户机的标识符。此属性包含客户机和服务器应用程序的地址。
	associated_partners_id ::= structure
	{
	client_SAP: integer;
	server_SAP: long unsigned;
	}

application_context_name(应用语境名)	在 COSEM 环境中,应用语境是预先存在的,在建立一个应用连接过程中通过它的名称访问。该属性包含此连接的应用语境名。 application-context-name ::= OBJECT IDENTIFIER; 见 GB/T 19882.33。

xDLMS_context_info
(xDLMS 语境信息)
	包含给定的连接中有关 xDLMS 语境的所有必需信息。
	xDLMS-context-type ::= structure
	{
	conformance : bitstring(24) ;
	max_send_pdu_size : long-unsigned;
	dlms_version_number : unsigned;
	quality_of_service : integer;
	cyphering_info : octet-string
	}
	结构中 conformance 元素包含服务器支持的 xDLMS 一致性块。
	max_receive_pdu_size 元素包含客户机发送的一个 xDLMS PDU 可能的最大长度(字节数)。这一参数与 DLMS-Initiate.response pdu 的 server-max-receive-pdu-size 相同(见 GB/T 19882.33)。
	激活连接中的 max_send_pdu_size 包含服务器发送的一个 xDLMS PDU 可能的最大长度(字节数)。这一参数与 DLMS-Initiate.request pdu 的 client-max-receive-pdu-size 相同(参见 GB/T 19882.33)。
	dlms_version_number 元素包含服务器支持的 DLMS 版本号。
	quality_of_service 元素任可使用。
	激活连接中的 cyphering_info 包含 DLMS-Initiate.request pdu 的专用密钥参数(见 GB/T 19882.33)。

authentication_mechanism_name(身份验证机制名)	该属性包含为连接的身份验证机制名。 mechanism-name ::= OBJECT IDENTIFIER; 见 GB/T 19882.33。 不用身份验证时,不需要验证机制名。
LLS_secret(LLS 密码)	该属性包含 LLS 身份验证程序所需的身份验证值。
association_status (连接状态)	该属性指示由连接对象形成的当前连接状态。 association-status(连接状态):enum:(列举) (0) non-associated(未连接) (1) association-pending(待定连接) (2) associated(已连接)

当此连接对象用于建立新的连接时,对连接 LN 对象属性的 SET 操作将变成有效。

对 object_list 属性的选择性访问参数

- 如不需要进行选择性访问,(在对 object_list 属性的 GET.request(.indication)服务请求中没有 access-selection-parameters 参数),则相应的响应(确认)服务将包含 object_list 属性的所有的 object_list_element。
- 如需要对 object_list 属性进行选择性访问(存在 access-selection-parameters 参数),则其响应应包含"过滤后的"的 object_list_element,如下表所示:

访问选择器的值	服务参数	注　　释
1	NULL	除 access_right 之外的所有信息都应包含在响应中。
2	class_list	通过类访问,在这种情况下,只有 object_list 中那些 class_id 在 class_list 中的 object_list_element 将包含在响应中,它有一个与类_列表相同的类标识。不包含 access_right 信息。 class_list ::= array class_id class_id ::= long-unsigned

访问选择器的值	服务参数	注　　释
3	object_Id_list	通过对象访问，将返回 object_Id_list 中对象范例的完整信息入口。 object_Id_list　::=　array object_Id

4	object_Id	在这种情况下，将返回所请求的 COSEM 对象范例的完整信息入口。
		object_Id ::= structure
		{
		class_id: long-unsigned;
		logical_name: octet-string
		}

方法说明

reply_to_HLS_authentication (data)（回答 HLS 身份验证数据）	本方法的远程调用将客户机"密码"处理过的"呼叫 StoC"(f(StoC))返给服务器，作为调用 ACTION. request 服务的数据服务参数(见 4.7.2)。 data ::= octet-string　　客户机对呼叫的响应。 如身份验证得到认可，则服务器的响应(ACTION. confirm)包含的结果为"OK"，并且服务器也将"秘密"处理过的"呼叫 CtoS"(f(CtoS))的结果传送给客户机，这个结果包含在响应服务的 data 参数中。 data ::= octet-string 服务器对呼叫的响应。 如验证没有得到认可，则响应的结果参数包含一个"non-OK"的值，并且无任何数据返回。
change_HLS_secret (data) (更改 HLS 秘密)	更改 HLS 密码(如密钥)。 data ::= octet-string[a]　　新的 HLS 密码。
Add_object (data) (添加对象)	增加要引用的对象到 object_list 中。 data ::= object_list_element (同上)
Remove_object (data) (删除对象)	从 object_list 中删除引用对象。 data ::= object_list_element (同上)
a　"新密码"的结构取决于所实现的安全机制。"新密码"可能包含有附加的校验位(checkbit)，并可能加密。	

5.13　连接 SN(Association SN)(class_id：12)

COSEM 逻辑设备能够在一个 COSEM 语境中利用短名引用建立应用连接，这种情况下，COSEM 逻辑设备通过连接 SN 类的范例来建立连接。对设备所能支持的每个连接，每个 COSEM 逻辑设备有一个本接口类的范例。

连接 SN 对象 short_name 本身固定在 COSEM 语境中。它在 C.2 中以 0XFA00 的形式给出。

连接 SN(Association SN)	0..n	class_id = 12, version = 1		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. object_list (static)	octet-string objlist_type			
方法	m/o			
1. reserved from previous versions	o			
2. reserved from previous versions	o			
3. read_by_logicalname	o			
4. get_attributes&methods	o			
5. change_LLS_secret	o			
6. change_HLS_secret	o			
7. reserved from previous versions				
8. reply_to_HLS_authentication	o			

属性说明

object_list
(对象列表)
	包含带有 base_name (short_name)、class_id、version 和 logical_name 的所有对象的列表。base_name 是第一属性(逻辑名)的 DLMS 对象名。
	objlist_type ::= array objlist_element
	objlist_element ::= structure
	{
	base_name: long;
	class_id: long-unsigned;
	version: unsigned;
	logical_name: octet-string
	}
	选择性访问(selective access)(见 4.2)对属性 object_list 来说是可用的(可选择的)。选择性访问参数定义如下:

对 object_list 属性的选择性访问参数

访问选择器的值	参　数	注　释
1	class_id: long-unsigned	传送 object_list 中具有规定 class_id 的子集。 对本响应:data ::= objlist_type

2	structure
	{
	class_id:
	long-unsigned;
	logical_name:
	octet-string
	}
		传送 object_list 中具有规定 class_id 和 logical_name 的入口。
		对本响应:data ::= objlist_element

方法说明

read_by_logicalname(data)
(按逻辑名读取数据)
	读取所选择对象的属性。对象由其 class_id 和 logical_name 规定。
	data ::= array attribute_identification
	attribute_identification ::= structure
	{
	class_id: long-unsigned;
	logical_name: octet-string;
	attribute_index: integer
	}
	其中,attribute_index 是指向对象内属性的指针(即偏移量)。
	attribute_index 0 传送所有的属性",attribute_index 1 表传送第一属性(即 logical_name),等等。
	对本响应,data 的数据类型与属性数据类型一致。

get_attributes&methods(data)
(获取属性和方法)
	传送在实际连接中属性和方法的访问权限的信息。对象由其 class_id 和 logical_name 规定。
	data ::= array object_identification
	object_identification ::= structure
	{
	class_id: long-unsigned;
	logical_name: octet-string
	attribute_index: integer
	}
	对本响应
	data ::= array access_description
	access_description ::= structure
	{
	attributes: bit-string;
	one_attributes: bit-string;
	methods: bit-string
	}
	位串中的位置标识属性/方法编号(第一位置↔第一属性,第二位置↔第一方法),并且它的值指示属性/方法是可用的/看得见或是不可用的(清除)。

change_LLS_secret(data) (更改 LLS 秘密)	更改 LLS 密码(如密钥)。 data ::= octet-string 新的 LLS 密码。
change_HLS_secret(data) (更改 HLS 秘密)	更改 HLS 密码(如密钥)。 data ::= octet-string 新的 HLS 密码
reply_to_HLS_authentication (data) (回答 HLS 身份验证数据)	本方法由远程调用将客户机对"质询 StoC"(StoC)处理的"秘集"返回给服务器,该结果包含在客户机调用 ACTION.request 服务的 data 参数中(见 4.7.2)。 data ::= octet-string 客户机对呼叫的响应。 如身份验证得到认可,则服务器的响应(ACTION.confirm)包含的结果为"OK",并且服务器也将"秘集"处理过的"质询 CtoS"(见 CtoS)的结果返回给客户机,这个结果包含在响应服务的 data 参数中。 data ::= octet-string 服务器对呼叫的响应。 如果还没有得到认可,则响应的结果参数包含一个"non-OK"的值,并且无任何数据返回。

[a] 在当前连接下,如至少有一个属性没有读取访问权限,则通过方法对 read_by_logicalname( )对 attribute index 0 进行读取指示出错信息"访问范围冲突"(见 DL/L 790.41—2002)。

[b] "新密码"的结构依赖所实现的安全机制。"新密码"可能包含有附加的校验位(checkbit),并且可能是加密的。

5.14 SAP 分配表(SAP assignment)(class_id: 17)

接口类 SAP 分配表包含逻辑设备到其 SAP(服务访问点)的分配信息(参见 GB/T 19897.4)。

SAP 分配表(SAP Assignment)	0...1	class_id = 17, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. SAP_assignment_list (static)	octet-string asslist_type			0
方法	m/o			
1. connect_logical_device ( )	o			

属性说明

SAP_assignment_list
(SAP 分配列表)
	包含物理设备中所有逻辑设备及其 SAP 地址的列表。
	asslist_type ::= array asslist_element
	asslist_element ::= structure
	{
	SAP: instance specific; (见下面的注)
	logical_device_name: octet-string
	}
	注：实际可以由"更低的"通信层完成一个有关面向连接的三层体系结构的详细资料见 GB/T 19882.x。

方法说明

connect_logical_device(data) (连接逻辑设备)	连接一个逻辑设备到一个 SAP。连接到 SAP 0 将切断设备与 SAP 的连接。多个的设备不能连接到一个 SAP 上(SAP 0 除外)。 data ::= asslist_element

5.15 寄存器监视器(Register monitor)(class_id: 21)

本接口类可定义脚本集(见 5.6)，在一个被监视的寄存器类型对象(数据、寄存器、扩展寄存器、需量寄存器等等)的一个属性值越过一组门槛值时执行。

在同一逻辑设备中，寄存器监视器接口类需要有一个脚本表接口类的范例。

寄存器监视器(Register monitor)	1...n	class_id = 21, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. thresholds (static) 3. monitored_value (static) 4. actions (static)	octet-string array value_definition array			
方法	m/o			

属性说明

threshold(门槛值)	提供与引用寄存器的属性进行比较的门槛值。 array threshold threshold: instance specific threshold 的数据类型与引用对象监视属性的数据类型一致。

monitored_value(监视值)	定义一个对象的哪一个属性要进行监视，只有简单数据类型的值允许监视。
	value_definition ::= structure
	{
	class_id: long unsigned;
	logical_name: octet string;
	attribute_index: integer
	}

action(操作)	定义引用对象的监视属性超越相应的门槛值时要执行的脚本。属性"action"与属性"threshold"有相同数量的元素，action_item 的顺序与门槛的顺序一致(见上文)。
	array action_set
	action_set ::= structure
	{
	action_up: action_item;
	action_down: action_item
	}
	其中，
	action_up 定义监视寄存器的属性数据超越门槛值上限时的操作
	action_item ::= structure
	{
	script_logical_name: octet-string;
	script_selector: long-unsigned
	}

5.16 实用表(Utility tables)(class_id: 26)

实用表类的一个范例封装着 ANSI C12.19 表的数据。

在本接口类的定义中，每张"表"表述一个范例，特定范例由其逻辑名标识。

公用表(Utility tables)	0..n	class_id = 26, version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. table_ID (static) 3. length 4. buffer	octet-string long-unsigned double-long-unsigned octet-string			
方法	m/o			

属性说明

table_ID(表标识)	表号，表号的规定与 ANSI 标准一样，既可以是标准表也可以是制造商的表。
Length(长度)	表缓冲区中的八位字节数。
Buffer(缓冲区)	表的内容。 选择性访问(Selective access)(见 4.2)对 buffer 属性是可用的(可选择的)，选择性访问参数定义如下：

对 buffer 属性的选择性访问参数

访问选择器	参 数	注 释
1	offset_access	对表按偏移量和计数进行访问使用 offset_selector 作为参数数据。
2	index_access	对表按元素标识和元素数目进行访问使用 index_ selector 作为参数数据。

offset_selector ::= structure

{

Offset	double-longunsigned	访问区起始位置的偏移量(八位字节数)，偏移量相对于表的开始。
Count	long-unsigned	请求或传送的八位字节数。

}

index_selector ::= structure

{

Index	array of long-unsigned	层次结构表中标识元素索引的序列。
Count	long-unsigned	请求或传送的元素数目。

index_selector ::= structure

{

Index	array of long-unsigned	在表中标识元素索引的序列。
Count	long-unsigned	被请求的或被翻译的元素号。计数值大于1将返回多个元素。 在给定的请求语境中，计数值为0，表示层次结构的表中开始于选择点的整个子树。

5.17 单操作时间表(Single action schedule)(class_id：22)

许多应用程序请求仪表内周期性操作，这些操作并不需要与费率连接(活动日历或时间表)。

单操作时间表(Single action schedule)	0..n	class_id = 22，version = 0		
属性	数据类型	最小值	最大值	缺省值
1. logical_name (static) 2. executed_script (static) 3. type (static) 4. execution_time (static)	octet-string script enum array			
服务	m/o			

属性说明

executed_script
(运行脚本)
	包含脚本表的逻辑名和要执行的脚本的选择器。
	script Structure
	{
	Script_logical_name　　octet-string
	Script_selector　　long_unsigned
	}
	逻辑名和脚本选择器定义被执行的脚本。

type(类型)	enum 1) execution_time 的大小=1；允许日期用通配符。 2) execution_time 的大小=n；所有时间值相同；不允许日期用通配符 3) execution_time 的大小=n；所有时间值相同；允许日期用通配符 4) execution_time 的大小=n；时间值可能不同；不允许日期用通配符 5) execution_time 的大小=n；时间值可能不同；允许日期用通配符

execution_time
(执行时间)
	规定脚本执行的时刻(time of day)。
	array　(octet-string，octet-string)
	两个八位字节串包含时间和日期，在该数组中，structure {
	time：octet-string；
	date：octet-string
	}
	时间和日期的数据格式见4.4的定义。
	"时间"不允许使用通配符；秒和百分之一秒字段应置为零。

6 接口类的维护

如 6.2 所述，对接口类作任何修改之后，旧的或不用的接口类版本都应记录到附录 E 中。

本部分制定前的接口类版本仅由 DLMS UA 保存。

6.1 新接口类

DLMS UA 保留作为唯一接口类管理员的权力。

6.2 接口类的新版本

对现有的接口类做任何修改都将产生一个新的版本(version ::= version+1)，并且应进行存档。修改已有的接口类需遵循以下规则：

a) 可增加新的属性和方法。

b) 已有的属性和方法可能失效，但失效的属性和方法的索引不应被其他属性和方法再使用。

c) 如果不能满足上述规则，则应建立新的接口类。

d) COSEM 接口类的新版本由 DLMS UA 管理。

6.3 接口类的撤消

除连接对象和逻辑设备名对象外，仪表内任何其他接口类的实例并非都是必需的。因此，即使是未使用的接口类也不应从标准中撤消。应保留它们以确保与已有的可能实现相兼容。

附 录 A
（规范性附录）
与接口类相关的协议

每一个通信装置及协议需要规定一些设置参数用于相应的操作。

A.1 IEC 本地端口启动（类_标：19）

代替该接口类规定的对于使用 GB/T 19897.1 通信的操作参数，若干端口需要组态，逻辑名定义如 D.1.1.11。

IEC 本地端口启动		0...0	class_id=19 version=0		
属性		数据类型	最小值	最大值	缺省值
1. logical_name	(static)	八位字节串			
2. defaut_mode	(static)	enum			
3. defaut_baud	(static)	enum			
4. prop_baud	(static)	enum			
5. response_time	(static)	enum			
6. device_addr	(static)	八位字节串			
7. pass_p1	(static)	八位字节串			
8. pass_p2	(static)	八位字节串			
9. pass_p3	(static)	八位字节串			
具体方法		m.0			

属性描述

默认_模式	定义所使用的仪表端口协议 Enum (0) 协议按照 GB/T 19897.1(模式 A...E) (1) 协议按照 GB/T 19897.4，使用所有的列举数 enum，该类中的其他属性是不适用的。
默认_波特	定义开启序列的波特率 Enum (0) 300 波特 (1) 600 波特 (2) 1 200 波特 (3) 2 400 波特 (4) 4 800 波特 (5) 9 600 波特 (6) 19 200 波特 (7) 38 400 波特 (8) 57 600 波特 (9) 115 200 波特

推荐_波特	定义仪表推荐的波特率 enum (0) 300 波特 (1) 600 波特 (2) 1 200 波特 (3) 2 400 波特 (4) 4 800 波特 (5) 9 600 波特 (6) 19 200 波特 (7) 38 400 波特 (8) 57 600 波特 (9) 115 200 波特
响应_时间	定义接受一个请求和传送一个响应之间的最短时间 enum (0)20 ms (1) 200 ms
装置_地址	装置地址依据 GB/T 19897.1 的八位字串
密码_P1	密码 1 依据 GB/T 19897.1 的八位字串
密码_P2	密码 2 依据 GB/T 19897.1 的八位字串
密码_W5	密码 W5 用于国家应用保留的八位字串

A.2 PSTN 调制解调器配置(类_标:27)

一个 PSTN 调制解调器配置对象储存与调制解调器初始化的数据有关,该调制解调器用来从/向一个装置传送数据。需配置几个调制解调器.逻辑_名定义如 D.1.1.2。

PSTN 调制解调器配置	0....n	class_id=27,version=0		
属性	数据类型	最小值	最大值	缺省值
1.逻辑_名 (静态) 2.通信_速度 (静态) 3.初始化_字符串 (静态) 4.调制解调器_概要 (静态)	八位字节字符串 enum 数组 数组	0	9	5
具体服务	m/o			

属性_描述

通信_速度	装置和调制解调器之间的通信速度,不一定是 WAN 通信速度。 Enum:(0) 300 波特 (1) 600 波特 (2) 1 200 波特 (3) 2 400 波特 (4) 4 800 波特 (5) 9 600 波特 (6) 19 200 波特 (7) 38 400 波特 (8) 57 600 波特 (9) 115 200 波特

初始化_字符串	该数据包含所有为正确配置调制解调器所需初始化指令，包括特殊调制解调器性能配置。
	初始化_字符串::= 数组
	{
	初始化_字符串_元素
	}
	初始化_字符串_元素::= 结构
	{
	请求：八位字符字符串
	响应：八位字符字符串
	}
	若该数组包含多于一个初始化字符串元素，则它们在接受一个匹配已定义响应的响应之后传送到调制解调器。
	
	注释：
	假定调制解调器已预先配置，接受初始化_字符串。
	若无须初始化，则初始化字符串为空。

调制解调器_概要	该数据定义从 Hayes 标准指令/响应到调制解调器具体字符串的映射。
	调制解调器_概要::= 数组
	{
	调制解调器_概要_元素
	}
	调制解调器_概要_元素::= 八位字节字符串
	调制解调器_概要数组务必包含调制解调器所需按以下顺序的字符串：
	元素 0：　确定
	元素 1：　连接
	元素 2：　响铃
	元素 3：　无载体
	元素 4：　错误
	元素 5：　连接 1 200
	元素 6：　无拨号音
	元素 7：　忙
	元素 8：　无响应
	元素 9：　连接 600
	元素 10：　连接 2 400
	元素 11：　连接 4 800
	元素 12：　连接 9 600
	元素 13：　连接 14 400
	元素 14：　连接 28 800
	元素 15：　连接 36 600
	元素 16：　连接 56 000

A.3 PSTN 自动响应(类_标:28)

一个 PSTN 自动响应对象和调制解调器之间数据传送管理的数据有关，该接口用来响应接入呼叫请求。若干个调制解调器需要配置。逻辑_名定义如 D.1.1.4。

PSTN 自动响应	0...n	class_id=28,version=0		
属性	数据类型	最小值	最大值	缺省值
1. 逻辑_名 （静态） 2. 模式 （静态） 3. 收听_窗口 （静态） 4. 状态 （动态） 5. 呼叫请求_次数 （静态） 6. 响铃_次数 （静态）	八位字节字符串 enum 数组 enum unsigned 响铃_次数_类型			
具体服务	m/o			

属性描述

模式	定义装置自动响应时 PSTN 线路的工作模式。 模式::= enum (0) 线路 dedicatedto 装置 (1) 允许共享线路管理和有限呼叫请求次数。一旦达到呼叫请求次数，窗口状态变为无效，直到下一个起始日期，而不管呼叫请求的结果。 (2) 允许共享线路管理和有限成功呼叫请求次数。一旦达到成功通信次数，窗口状态变为无效，直到下一个起始日期。 (200...255) 制造商具体模式

收听_窗口	包含起始和结束瞬时，当窗口有效（起始瞬时），窗口无效（结束瞬时）。日期格式是 UTC。起始_日期中定义周期。例如，当没有指定日期（等于 255）这意味着我们有一个日常线路管理，每天，每月(……窗口管理可以得到定义。
	收听_窗口::= 数组 窗口_元素
	窗口_元素::= 结构
	{
	起始_时间 UTC
	结束_时间 UTC
	}

状态	定义窗口的状态 状态::= enum (0) 失效：装置拒绝无新的呼叫请求接入。该状态在下一个收听窗口起始时自动复位成有效。 (1) 有效：装置响应下一个接入呼叫请求。 (2) 锁定：该值可由装置自动设置或由一个具体客户端设置，当该客户端已完成读取对话并想要在窗口持续期结束之前将线路交换用户。该状态在下一个收听窗口起始时自动复位成有效。
呼叫请求_次数	该次数用于在模式 1 和模式 2 中引用。 当设为 0，意味着无限。

响铃_次数	定义仪表连接到调制解调器之前的响铃次数。判别两种情况：窗口内响铃次数由属性"收听_窗口"定义，和"收听_窗口"以外的响铃次数。
	响铃_次数_类型:= 结构
	{
	响铃_次数_窗口内 unsigned
	(0=窗口内无连接)
	响铃_次数_窗口外 unsigned
	(0=窗口外无连接)
	}

A.4 PSTN 自动拨号(类_标:29)

一个 PSTN 自动拨号对象储存装置和调制解调器之间为实现自动拨号进行的数据传送控制的数据。需配置几个调制解调器。逻辑_名定义如 D.1.1.3。

PSTN 自动拨号		0....n	class_id=29,version=0		
属性		数据类型	最小值	最大值	缺省值
1. 逻辑_名	(静态)	八位字节字符串			
2. 模式	(静态)	enum			
3. 重复	(静态)	unsigned			
4. 重复_延迟	(静态)	long-unsigned			
5. 呼叫请求_窗口	(静态)	数组			
6. 电话_列表	(静态)	数组			
具体服务		n/a			

属性描述

模式	定义装置是否能够自动拨号 模式::= enum (0) 无自动拨号 (1) 任何时间允许自动拨号。 (2) 在呼叫请求窗口有效时间内允许自动拨号。 (3) 在呼叫请求窗口有效时间内允许"正常"自动拨号;任何时间允许"警告"启动自动拨号 (200...255) 制造商具体模式。
重复	在拨号不成功情况下最大尝试次数。
重复_延迟	时间延迟,以秒计时直到再次重复一个不成功拨号尝试。重复_延迟=0 意味着未确定延迟。

呼叫请求_窗口	包含起始和结束日期_时间标志,在窗口有效(起始瞬时),或无效(结束瞬时)。日期格式是 UTC。窗口_日期暗中定义周期。例如::当没有指定日期(等于 255)这意味着我们有一个日常线路管理。每天,每月……窗口管理可以得到定义。
	呼叫请求_窗口::= 数组 窗口_元素
	窗口_元素::= 结构
	{
	起始_时间 UTC
	结束_时间 UTC
	}

电话_列表	包含电话数目,装置调制解调器需要在特定情况下呼叫。不包含数组入口和环境之间的连接。
	电话_列表::= 数组
	{
	电话_数目
	}
	电话_数目::= 八位字节字符串

A.5 IEC HDLC 启动(类_标:23)

HDLC 启动事例包含所有建立通信通道所需的数据,依据 GB/T 19897.4 若干个通信通道需要配置。逻辑_名定义如 D.1.1.13。

IEC HDLC 启动		0...n	class_id=23,version=0		
属性		数据类型	最小值	最大值	缺省值
1.逻辑_名	(静态)	八位字节字符			
2.通信_速度	(静态)	串	0	9	5
3.传输_窗口_大小	(静态)	enum	1	7	1
4.接收_窗口_大小	(静态)	unsigned	1	7	1
5.最大_传输_信息_长度	(静态)	unsigned	32	128	128
6.最大_接收_信息_长度	(静态)	unsigned	32	128	128
7.字符间_暂停时间	(静态)	unsigned	20	1 000	30
8.休止_暂停时间	(静态)	long-unsigned	0		120
9.装置_地址	(静态)	long-unsigned long-unsigned	16	0x3ffd	0
具体服务		m/o			

属性描述

通信_速度	装置和调制解调器之间的通信速度,不一定是 WAN 上的通信速度。 Enum: (0) 300 波特 (1) 600 波特 (2) 1 200 波特 (3) 2 400 波特 (4) 4 800 波特 (5) 9 600 波特 (6) 19 200 波特 (7) 38 400 波特 (8) 57 600 波特 (9) 115 200 波特 该通信速度可被覆盖。若另一个协议的特殊模式通过装置的 HDLC 模式。
传输_窗口_大小	装置或系统在需要从对应工作站得到确认之前可以传输窗体的最大数目,在登录期间允许一个较小的值。
接收_窗口_大小	装置或系统在需要向对应工作站发送确认之前可以接收窗体的最大数目,在登录期间允许一个较小的值。
最大_传输_信息_长度	装置能够传输的最大信息字段长度,在登录期间允许一个较小的值。
最大_接收_信息_长度	装置能够接收的最大信息字段长度,在登录期间允许一个较小的值。
字符间_暂停时间	定义时间,以毫秒计时,当从主工作站上接收到任何字符,装置将已接收到的数据看作一个已完成的窗体。
休止_暂停时间	定义时间,以秒计时,当从主工作站接收到任何窗体,装置将进行一次断开。 当该值设为 0,表明休止_暂停时间无法操作。

装置_地址	包含装置的物理装置地址： 在单精度比特寻址中： 0x00：　　无_工作站地址 0x01...0x0f　　预留 0x10...0x7d　　可用地址空间 0x7e：　　“呼叫请求”装置地址 0x7f：　　广播地址 在双精度比特寻址中： 0x0000　　无_工作站地址 0x0001...0x000f　　预留 0x0010...0x3ffd　　可用地址空间 0x2fff：　　“呼叫请求”装置地址 0x3ff1　　发送地址

A.6 IEC 双绞线(1)启动(类_标:24)

IEC 双绞线(1)启动对象包含所有启动通信通道所需数据。依据草案(GB/T 19897.2—2005 需配置几个通信通道。逻辑_名定义如 D.1.1.13)。

IEC 双绞线(1)启动	0.0.[illegible]	class_id=24,version=0		
属性	数据类型	最小值	最大值	缺省值
1.逻辑_名　(静态) 2.二级地址　(静态) 3.一级_地址_列表　(静态) 4.tabi_列表　(静态) 5.致命_错误　(动态)	八位字节字符串 八位字节字符串 一级_地址_列表_类型 tabi_列表_类型 enum			
具体服务	m/o			

属性描述

二级_地址	二级_地址储存对应真实设备的二级工作站的 ADS(comp. GB/T 19897.2—2005)。 八位字节字符串(大小(6))

一级_地址_列表	一级_地址_列表储存 ADP 列表或一级工作站物理地址，实际设备(二级工作站)的各个逻辑装置均已程序化(comp. GB/T 19897.2—2005)。
	一级_地址_列表_类型::= 数组
	{
	一级_地址_元素
	}
	一级_地址_元素::= 八位字节字符串(大小(1))

tabi_列表	tabi_列表代表 TAB(i)列表，当实际设备(二级工作站)已程序化，在出现“忽略工作站呼叫请求”情况下(comp. GB/T 19897.2—2005)。 Tabi_列表_类型::= 数组 tabi_元素 Tabi_元素::= 整型 8

致命_错误	致命错误代表如 GB/T 19897.2—2005 所述协议致命错误之一的最后一次出现。 该变量的初始默认值是 00H。各个致命错误均表示出来。 Enum (0) 无_错误 (1) t-EP-1F (2) t-EP-2F (3) t-EL-4F (4) t-EL-5F (5) eT-1F (6) eT-2F (7) e-EP-3F (8) e-EP-4F (9) e-EP-5F (10) e-EL-2F

附 录 B
（规范性附录）
数据模型和协议

此数据模型使用通用构架模块来定义计量设备的复杂功能。此模型不涉及内部具体执行内容，它仅提供了一种类似于在接口处获得仪表功能性的视角。

该通信协议定义了数据如何被访问和交换的，下列图形是对它的说明：

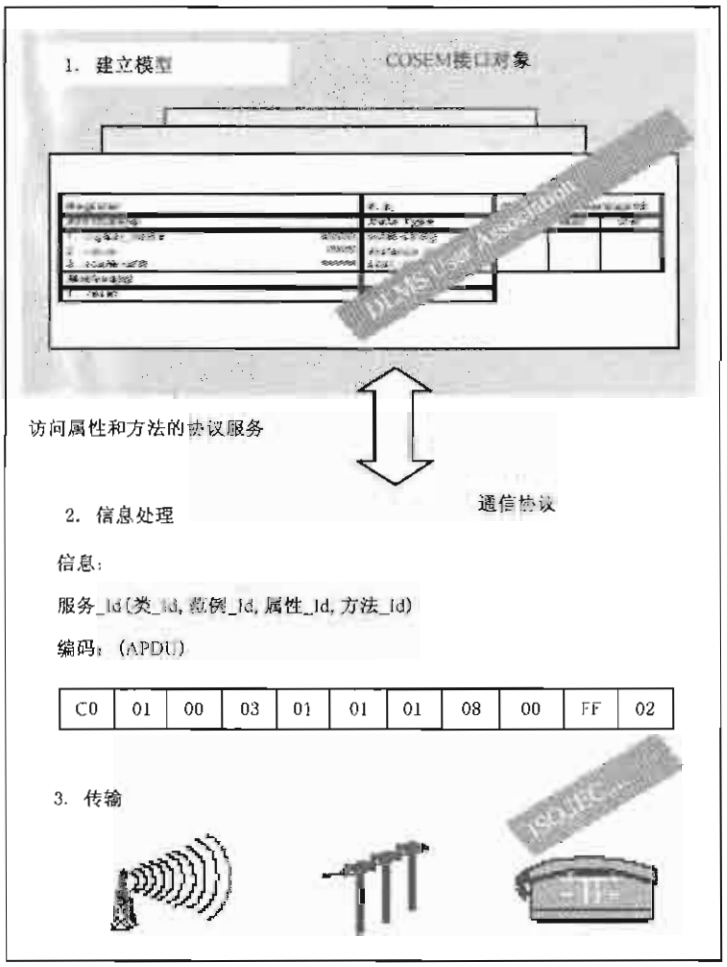

图 B.1 COSEM 的三步处理协议

- COSEM 规范规定了计量领域中具体的接口类，仪表的功能性由这些称之为 COSEM 接口对象的接口类实例来定义，这些实例在本部分中定义。标识 COSEM 接口对象的逻辑名在 GB/T 19882.31 中定义；
- COSEM 接口对象的属性和方法可以通过应用层报文服务来访问和使用。
- 低层通信协议传输这些信息。

附 录 C
（规范性附录）
使用短名访问属性和方法

COSEM 对象的属性和方法可以通过两种不同方法访问。

利用 COSEM 逻辑名：在这种情况下，COSEM 对象的属性和方法可以通过 COSEM 对象范例所属标识符访问。

属性访问是：

类_标，“逻辑_名”属性值，属性_索引。

方法访问是：

类_标，“逻辑_名”属性值，方法_索引。

属性_索引用作所要求属性的标识符。

方法_索引用作所要求方法的标识符。

利用缩略名：这种类型的访问方式用于简单装置中，在这种情况下，COSEM 对象的各个属性和方法由一个 13 bit 整数标识。缩略名的用法与 DLMS 命名变量名的语法相同。

C.1 分配缩略名准则

数据 类_标=1，版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
值	X+8	
具体方法		

寄存器 类_标=3，版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
值	X+8	
标尺_单元	X+16	
具体方法		
复位	X+40	

扩展寄存器 类_标=4,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
值	X+8	
标尺_单元	X+16	
状态	X+24	
俘获_时间	X+32	
具体方法		
复位	X+56	

用量寄存器 类_标=5,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
当前_平均_值	X+8	
最终_平均_值	X+16	
标尺_单元	X+24	
状态	X+32	
俘获_时间	X+40	
当前_起始_时间	X+48	
周期	X+56	
周期_数目	X+64	
具体方法		
复位	X+88	
下一_周期	X+96	

寄存器激活 类_标=6,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
寄存器_分配	X+8	
屏蔽_列表	X+16	
有效_屏蔽	X+24	
具体方法		
增加_寄存器	X+48	
增加屏蔽	X+56	
删除屏蔽	X+64	

概要类 类_标=7,版本=1	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
缓冲区	X+8	利用参数化访问实现选择性访问缓冲区
俘获_对象	X+16	
俘获_周期	X+24	
排序_方法	X+32	
排序_对象	X+40	
使用_入口	X+48	
概要_入口	X+56	
具体方法		
复位		
俘获	X+88	
	X+96	

时钟 类_标=8,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
时间	X+8	
时区	X+16	
状态	X+24	
白天_保存_开始	X+32	

白天_保存_结束	X+40	
白天_保存_偏差	X+48	
白天_保存_允许	X+56	
时钟_基点	X+64	
具体方法		
校准_时间	X+96	
校准_到_测量_周期	X+104	
校准_到_分	X+112	
校准_到_当前_时间	X+120	
预设_校准_时间	X+128	
转换_时间	X+136	

命令表 类_标=9,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
命令	X−8	
具体方法		
执行	X+32	

时间表 类_标=10,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
入口	X+8	
具体方法		
允许/禁止	X+32	
插入	X+40	
删除	X+48	

具体日期表 类_标=11,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
入口	X+8	
具体方法		
插入	X+32	
删除	X+40	

活动日历 类_标=20,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
日历_名_有效	X+8	
季节_概要_有效	X+16	
星期_概要_有效	X+24	
日期_概要_表_有效	X+32	
日历_名_无效	X+40	
季节_概要_表_无效	X+48	
星期_概要_无效	X+56	
日期_概要_表_无效	X+64	
激活_无效_日历_时间	X+72	
具体方法		
激活_无效_日历	X+96	

连接 SN 类_标＝12，版本＝1	短名	注释
属性		
逻辑_名	X::＝0×FA00	X 是对象基本_名
对象_列表	X＋8	利用参数化访问实现选择性访问对象_列表
具体方法		
按逻辑名_读取	X＋48	这种方法可以用参数化访问功能
取得_属性和方法	X＋56	这种方法可以用参数化访问功能
变更_LLS_密码	X＋64	
变更_HLS_密码	X＋72	
响应_HLS_验证	X＋80	
连接 SN 对象的基名对应于当前连接是 0xFA00		

SAP 赋值 类_标＝17，版本＝0	逻辑名	注释
属性		
逻辑_名	X	X 是对象基本_名
SAP_赋值_列表	X＋8	
具体方法		
连接_逻辑_装置	X＋32	

寄存器监视器 类_标＝21，版本＝0	逻辑名	注释
属性		
逻辑_名	X	X 是对象基本_名
阈值	X＋8	
监视_值	X＋16	
事件	X＋24	
具体方法		

公用表 类_标=26,版本=0	缩略名	注释
属性		
逻辑_名	X	X是对象基本_名
表_标	X+8	
长度	X+16	
缓冲区	X+24	
具体方法		

单事件时间表 类_标=22,版本=0	缩略名	注释
属性		
逻辑_名	X	X是对象基本_名
执行_命令	X+8	
类型	X+16	
执行_时间	X+24	
具体方法		

IEC光学端口启动 类_标=19,版本=0	缩略名	注释
属性		
逻辑_名	X	X是对象基本_名
默认_模式	X+8	
默认_波特	X+16	
推荐_波特	X+24	
响应_时间	X+32	
装置_地址	X+40	
密码_P1	X+48	
密码_P2	X+56	
密码_W5	X+64	
具体方法		

PSTN调制解调器配置 类_标=27,版本=0	缩略名	注释
属性		
逻辑_名	X	X是对象基本_名
通信_速度	X+8	
初始化_字符串	X+16	
调制解调器_概要	X+24	
具体方法		

PSTN 自动响应 类_标＝28，版本＝0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
模式	X＋8	
收听_窗口	X＋16	
状态	X＋24	
呼叫请求_次数	X＋32	
响铃_次数	X＋40	
具体方法		

PSTN 自动拨号 类_标＝29，版本＝0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
模式	X＋8	
重复	X＋16	
重复_延迟	X＋24	
呼叫请求_窗口	X＋32	
电话_列表	X＋40	
具体方法		

IEC HDLC 启动 类_标＝23，版本＝0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
通信_速度	X＋8	
传输_窗口_大小	X＋16	
接收_窗口_大小	X＋24	
最大_传输_信息_长度	X＋32	

最大_接收_信息_长度	X+40	
字符间_暂停时间	X+48	
休止_暂停时间	X+56	
装置_地址	X+64	
具体方法		

IEC 双绞线(1)启动 类_标=24,版本=0	缩略名	注释
属性		
逻辑_名	X	X 是对象基本_名
二级_地址	X+8	
一级_地址_列表	X+16	
Tabi_列表	X+24	
致命_错误	X+32	
具体方法		

C.2 为特殊 COSEM 对象保留基本_名

为准许访问装置，提供按缩略_名访问，有些缩略_名保留作为特殊 COSEM 对象的基本_名。保留名的范围是从 0xFA00 到 0xFFF8。

定义以下具体基本_名：

基本_名 (对象名)	COSEM 对象
0Xfa00	短名 SN
0XFB00	命令者(准备"发送_接收命令")
0XFC00	SAP 分配
0XFD00	寄存器对象包含"COSEM 逻辑装置名"在属性"值"中

附 录 D
（规范性附录）
标准与 OBIS 关系

OBIS 标识系统作为 COSEM 逻辑_名的一个基础。COSEM 对象命名系统由基本准则（第 4 章）确定。实际数据项的标识由 GB/T 19882.31 确定。

以下章节中规定了在 COSEM 环境中定义的使用。

所有代码并未明确列出，但预留了制造商代码范围（128..255）。

D.1 数据项映射到 COSEM 对象和属性

本章确定 OBIS 标识的用法和其映射到 COSEM 特定接口类对象和它们的属性。

D.1.1 抽象 COSEM 对象

本条包含未直接连接到能量类型的数据项的定义。

值集合 C 抽象对象（A=0）	
0	常规用途 COSEM 对象
1	IC"时钟"COSEM 对象
2	COSEM 对象 IC"PSTN"调制解调器配置和相关 IC-s
10	IC"命令表" COSEM 对象
11	IC"具体日期表" COSEM 对象
12	IC"日历时间表" COSEM 对象
13	IC"活动日历" COSEM 对象
14	IC"寄存器激活" COSEM 对象
15	IC"单事件时间表" COSEM 对象
20	IC"IEC 光学端口启动" COSEM 对象
21	标准读出定义
22	IC"IEC HDLC 启动" COSEM 对象
23	IC"IEC 双绞线（1）启动" COSEM 对象
40	IC"协同 SN/LN" COSEM 对象
41	IC"SAP 分配" COSEM 对象
42	COSEM 逻辑装置名
65	IC"公用表" COSEM 对象
128...175	制造商具体 COSEM 相关抽象对象
保留所有其他代码。	

D.1.1.1 时钟

该 COSEM 对象控制物理装置的系统时钟，是接口类"时钟"的一个范例。

时钟				OBIS 标识			
	IC	A	B	C	D	E	F
时钟对象	时钟	0	×	1	0	×	0×FF

值集合 E 的用法：

若只有一个对象范例，则值 E 为 0。

若在同一物理装置中有多于一个对象范例，则其值集合 E 将从 0 开始计数范例到所需最大值。

D.1.1.2 PSTN 调制解调器配置

该 COSEM 对象确定和控制装置性能，通过一个 PSTN 调制解调器实现通信，是接口类"PSTN 调制解调器配置"的一个范例。

PSTN 调制解调器配置	OBIS 标识						
	IC	A	B	C	D	E	F
PSTN 调制解调器配置对象	PSTN 调制解调器配置 开	0	×	2	0	0	0×FF

值集合 B 的用法：

若在同一物理装置中有多于一个对象范例，则其值集合 B 将计数通信通道。

D.1.1.3 PSTN 自动拨号

该 COSEM 对象确定和控制装置性能，利用一个 PSTN 调制解调器实现自动拨号，是接口类"PSTN 自动拨号"的一个范例。

PSTN 自动拨号	OBIS 标识						
	IC	A	B	C	D	E	F
PSTN 自动拨号对象	PSTN 自动拨号	0	×	2	1	0	255

值集合 B 的用法：

若在同一物理装置中有多于一个对象范例，则其值集合 B 将计数通信通道。

D.1.1.4 PSTN 自动响应

该 COSEM 对象确定和控制装置性能，通过一个 PSTN 调制解调器实现自动响应，是接口类"PSTN 自动响应"的一个范例。

PSTN 自动响应	OBIS 标识						
	IC	A	B	C	D	E	F
PSTN 自动响应对象	PSTN 自动响应	0	×	2	2	0	255

值集合 B 的用法：

若在同一物理装置中有多于一个对象范例，则其值集合 B 将计数通信通道。

D.1.1.5 脚本表

这些 COSEM 对象控制装置性能。

预定义几个接口类"命令表"的范例，以隐藏命令形式出现，只有 action() 方法可以访问。

下表仅包含列表命令的"标准"范例标识，执行这些命令的具体范例使用值集合 D 中非零值。

命令表对象	OBIS 标识						
	IC	A	B	C	D	E	F
全耗仪表复位[1]	命令表	0	×	10	0	0	255
MDI 复位/计费周期结束[1]	命令表	0	×	10	0	1	255
计费 命令表	命令表	0	×	10	0	100	255
启动测试模式[1]	命令表	0	×	10	0	101	255
启动正常模式[1]	命令表	0	×	10	0	102	255
设置输出信号	命令表	0	×	10	0	103	255
发送命令表	命令表	0	×	10	0	105	255
[1] 启动这些命令通过调用 action( ) 方法到对应命令对象的标识符 1。							

计费命令表规定了通过标准化应用程序，如何调用特定 TARIFF 环境的启动，进入计费的入口点。

发送命令表允许标准化应用程序扩展正常需求功能进入点。

D.1.1.6 特殊日期表

该 COSEM 对象确定和控制装置性能，通过特殊日期日历实现时钟控制，是接口类"特殊日期表"的一个范例。

特殊日期表	OBIS 标识						
	IC	A	B	C	D	E	F
特殊日期表对象	特殊日期表	0	×	11	0	0	255

D.1.1.7 时间表

该 COSEM 对象通过有序方法确定和控制装置性能，是接口类"时间表"的一个范例。

时间表	OBIS 标识						
	IC	A	B	C	D	E	F
时间表对象	时间表	0	×	12	0	×	255

值集合 E 的用法：

若只有一个对象范例，则值 E 为 0。

若在同一物理装置中有多于一个对象范例，则其值集合 E 将从 0 开始计数范例到所需最大值。

D.1.1.8 活动日历

该 COSEM 对象通过基于日历的方法规定和控制装置性能，是接口类"活动日历"的一个范例。

活动日历	OBIS 标识						
	IC	A	B	C	D	E	F
活动日历对象	活动日历	0	×	13	0	0	255

D.1.1.9 寄存器激活

该 COSEM 对象用于处理不同的费率结构，这是一个接口类"寄存器激活"的范例。

寄存器激活	OBIS 标识						
	IC	A	B	C	D	E	F
寄存器激活对象	寄存器激活	0	×	14	0	0	255

D.1.1.10 单操作时间表

该 COSEM 对象控制装置的状态，预定义一个接口类范例"单操作时间表"。下表仅包含所列"单操作时间表"的"标准"范例标识。执行这些接口类具体范例应使用数组 D 中非零值。

单操作时间表	OBIS 标识						
	IC	A	B	C	D	E	F
记帐结束时间表	单事件时间表	0	×	15	0	0	255

D.1.1.11 IEC 本地端口建立

依据 GB/T 19897.1 中的本地端口通信参数，该 COSEM 对象规定和控制着装置状态，这是接口类"IEC 本地端口建立"的一个范例。

IEC 本地端口建立	OBIS 标识						
	IC	A	B	C	D	E	F
IEC 光学端口建立对象	IEC 本地端口建立	0	×	20	0	0	255
IEC 电气端口建立对象	IEC 本地端口建立	0	×	20	0	1	255

数组 B 的用法：

若在同一物理装置中有多于一个对象范例，则数组 B 将计数通信通道。

D.1.1.12 标准读出集

定义一组 COSEM 对象，如 GB/T 19887.1(模式 A 到 D)所示，携带标准读数。

标准读数	OBIS 标识						
	IC	A	B	C	D	E	F
常规本地端口读数	集	0	0	21	0	0	255
常规显示读数	集	0	0	21	0	1	255
备用显示读数	集	0	0	21	0	2	255
服务显示读数	集	0	0	21	0	3	255
可组态仪表数据列表	集	0	0	21	0	4	255
附加读数集 1	集	0	0	21	0	5	255
.....							
附加读数集 n	集	0	0	21	0	N	255

标准读数对象也可与一个能量类型以及一个通道有关，见 GB/T 19882.31。

D.1.1.13 HDLC 建立

该 COSEM 对象在连接协调时刻，利用 HDLC 协议规定和控制装置的状态。这是接口类"IEC HDLC 建立"的一个范例。

IEC HDLC 建立	OBIS 标识						
	IC	A	B	C	D	E	F
IEC HDLC 建立对象	IEC HDLC 建立	0	×	22	0	0	0×FF

数组 B 的用法：

若在同一物理装置中有多于一个对象范例，则其值集合 B 将计数通信通道。

D.1.1.14 IEC 双绞线(1)建立

依据 GB/T 19897.2 中的通信参数，该 COSEM 对象规定和控制着装置状态，这是接口类"IEC 双绞线(1)建立"的一个范例。

IEC 双绞线(1)建立	OBIS 标识						
	IC	A	B	C	D	E	F
IEC 双绞线(1)建立对象	IEC 双绞线(1)建立	0	×	23	0	0	255

数值组 B 的用法：

若在同一物理装置中有多于一个对象范例，则数组 B 将计数通信通道。

D.1.1.15 连接对象

一系列 COSEM 对象用来标识物理装置内的连接对象。

协同对象	OBIS 标识						
	IC	A	B	C	D	E	F
当前连接	连接 n LN/SN	0	0	40	0	0	255
连接，范例 1	连接 n LN/SN	0	0	40	0	1	255
……							
连接，范例 n	连接 n LN/SN	0	0	40	0	N	255

D.1.1.16 SAP 分配

一个 COSEM 对象用来传递物理装置内 SAP 分配信息。

SAP 分配对象	OBIS 标识						
	IC	A	B	C	D	E	F
当前物理装置 SAP 分配	SAP 分配	0	0	41	0	0	255

D.1.1.17 COSEM 逻辑装置名

每个 COSEM 逻辑装置由其逻辑名在全世界范围内标识。

若用短名寻址，COSEM 逻辑装置名总是可以在相同位置找到。这是一个基址为 0xfd00 的数据或寄存器对象。

COSEM 逻辑装置名	OBIS 标识						
	IC	A	B	C	D	E	F
COSEM 逻辑装置名	数据[a]	0	0	42	0	0	0×FF
[a] 即使类"数据"无用，类"寄存器"(使用标尺=0，单位=255)仍可以使用。							

D.1.1.18 实用表

以下概括了利用逻辑名 OBIS 编码的 ANSI 公用表范例编码。

a) 利用 0 值确定抽象对象。

b) 范例表组。

c) 利用值 63 标示实用表的特殊定义。

d) 表组选择器。

e) 组内表标号。

f) 利用值 255 用于当前费率周期。

实用表	OBIS 标识						
	IC	A	B	C	D	E	F
标准表 0—127	实用表	0	×	65	0	n	255
标准表 128—255	实用表	0	×	65	1	n	255
.....							
标准表 1920—2047	实用表	0	×	65	15	n	255
制造商表 0—127	实用表	0	×	65	16	n	255
制造商表 128—255	实用表	0	×	65	17	n	255
.....							
制造商表 1920—2047	实用表	0	×	65	31	n	255
标准挂起表 0—127	实用表	0	×	65	32	n	255
标准挂起表 128—255	实用表	0	×	65	33	n	255
.....							
标准挂起表 1920—2047	实用表	0	×	65	47	n	255
Mfg 挂起表 0—127	实用表	0	×	65	48	n	255
Mfg 挂起表 128—255	实用表	0	×	65	49	n	255
.....							
Mfg 挂起表 1920—2047	实用表	0	×	65	63	n	255

D.1.1.19 装置标识

一系列 COSEM 对象用来传递装置标识号。可由制造商确定编号(制造商编号)或由用户确定。

不同标识号是接口类"数据"的范例,数据类型是八位字串。

若使用一个以上标识号,则允许将它们整和进一个接口类"通用集"范例。在这种情况下,所俘获对象是装置标识数据对象,俘获周期是 1,有实际值。排序方法是 FIFO,集的入口限定到 1。

装置标识	OBIS 标识						
	IC	A	B	C	D	E	F
装置标识 1 对象(制造商号)	数据[a]	0	×	96	1	0	255
.....							
装置标识 10 对象	数据[a]	0	×	96	1	9	255
装置标识的对象	通用集	0	×	96	1	255	255

[a] 即使类"数据"无用,类"寄存器"(使用 标尺=0,单位=255)仍可以使用。

D.1.1.20 参数变化

一组简单 COSEM 对象描述装置配置历史。所有的值由接口类"数据"传递。

参数变化	OBIS 标识						
	IC	A	B	C	D	E	F
配置程序变更数对象	数据[a]	0	×	96	2	0	0×FF
配置程序变化最后日期对象	数据[a]	0	×	96	2	1	0×FF
时间开关程序变化最后日期对象	数据[a]	0	×	96	2	2	0×FF
纹波控制接收机程序变化最后日期对象	数据[a]	0	×	96	2	3	0×FF
保护配置程序变化数[b]	数据[a]	0	×	96	2	10	0×FF

[a] 即使类"数据"无用,类"寄存器"(使用 标尺=0,单位=255)仍可以使用。

[b] 保护配置表示需要打开表壳才能修改的特征。

D.1.1.21 输入/输出控制信号

这些 COSEM 对象确定和控制装置输入/输出线路状态。状态由接口类"数据"一个范例规定。

D.1.1.22 内部状态

这些 COSEM 对象规定了控制信号的内部状态和内部运行状态。

用于这些对象的接口类是采用八位字串类型的"数据",这些数据类型以位图形式传递二进制信息。

D.1.1.23 电源故障

可利用各种可能方式表示来自装置电源故障监控的值。对事件的简单计数由 COSEM 对象接口类"数据"及 unsigned 或 long unsigned 数据类型来表示。若有更复杂的信息需要表示,则应采用 COSEM 对象接口类"寄存器"或"通用集"。

D.1.1.24 错误值

一组 COSEM 对象用来传递装置错误指示。

不同错误值是接口类"数据"的范例,数据类型是八位字节串。

若使用一个以上的编号,则允许将它们整和进一个接口类"通用集"范例。在这种情况下,所捕获对象是装置标识数据对象,捕获周期=1 且具有实效值,排序方法是 FIFO,集的入口限于 1。

错误值		OBIS 标识					
	IC	A	B	C	D	E	F
错误 1 对象	数据	0	0	97	97	0	0×FF
.....							
错误 10 对象	数据	0	0	97	97	0	0×FF
错误集对象	通用集	0	0	97	97	0	0×FF
a 即使类"数据"无属"类"寄存器"(使用 和尺=0,单位=无)的可以使用							

错误编码对象也可与一个能量类型以及一个通道有关,见 GB/T 19882.31。

D.1.2 有关电气能量 COSEM 对象

D.1.2.1 数值组 D 定义

由数值组 D 定义的不同值,不是表述先前的费率周期,并按如下方式建模:

- 累积值由接口类"寄存器"范例的 COSEM 对象表述。
- 最大值和最小值由接口类"通用集"范例 COSEM 对象按最大值和最小值排序方法表示,深度依据执行确定,捕获对象依据执行确定。单一最大值或最小值可由接口类"扩展寄存器"范例的 COSEM 对象交替表述。
- 当前和最终平均值是 COSEM 对象的各自的属性,该对象是接口类"需量寄存器"的范例,并使用当前值的 OBIS 编码作为逻辑名。
- 时间积分值由接口类"寄存器"或"扩展寄存器"范例的 COSEM 对象表述。

D.1.2.2 前期费率周期数据

COSEM 将若干费率周期的值或值列表作为集来处理。

数值组 F 具有一个介于 1 和 99 之间的值,101 直接访问前期费率周期的数据是可行的(见 GB/T 19882.31 数值组 F)。附加在保存值的入口深度 1 及包含的保存时间标签由接口类"通用集" COSEM 对象管理。

注意:无任何前期有效值用于当前和最终平均类型值。

数值组 F 具有一个介于 102 和 126 之间的值,该数据由接口类"通用集"范例 COSEM 对象表述,这是适用于控制属性的值。

D.1.2.3 电气标识号

不同电气标识编码是接口类"数据"的范例，并采用八位字节串数据类型。

若使用一个以上的编号，则允许将它们整和进一个接口类"通用集"范例。在这种情况下，所捕获对象是电气标识数据对象，捕获周期是1且具有实效值，排序方法是FIFO，集的入口限于1。

电气标识	OBIS标识						
	IC	A	B	C	D	E	F
电气标识1对象	数据[a]	1	×	0	0	0	0×FF
.....							
电气标识10对象	数据[a]	1	×	0	0	9	0×FF
电气标识的对象	集	1	×	0	0	0×FF	0×FF

[a] 即使类"数据"无用，类"寄存器"（使用标尺=0，单位=255）仍可以使用。

D.1.2.4 记帐周期入口

这些值由接口类"数据"范例表述，数据类型是unsigned。

与入口有关的历史数据	OBIS标识						
	IC	A	B	C	D	E	F
费率周期计数器对象	数据[a]	1	×	0	1	0	0×FF
有效记账周期数据实效目对象	数据[a]	1	×	0	1	1	0×FF

[a] 即使类"数据"无用，类"寄存器"（使用标尺=0，单位=255）仍可以使用。

在表述先前费率周期数据的COSEM对象中，先前数据值的时间标签是捕获对象的一部分，该值与一个通道有关。

D.1.2.5 程序入口

这些值由接口类"数据"范例表述，数据类型是unsigned或long unsigned。

程序入口	OBIS标识						
	IC	A	B	C	D	E	F
配置程序编号对象	数据[a]	1	×	0	2	0	0×FF
时间转换程序编号对象	数据[a]	1	×	0	2	2	0×FF
RCR程序编号对象	数据[a]	1	×	0	2	3	0×FF

[a] 即使类"数据"无用，类"寄存器"（使用标尺=0，单位=255）仍可以使用。

程序入口也与一个通道相关，见GB/T 19882.31。

注：对于完整的列表，见GB/T 19882.31—2005表12。

D.1.2.6 输入和输出脉冲常数，额定值

这些值由接口类"寄存器"范例表述。

D.1.2.7 费率

这些值由接口类"数据"范例表述，以及可用的数据类型。

D.1.2.8 阈值

这些值由接口类"寄存器监视器"范例表述，通过定义受控寄存器，以及阈值本身和超出阈值时所完成的操作。

D.1.2.9 测量-寄存器-周期值，时间入口

这些值由接口类"数据"或"寄存器"范例表述。

时间入口	OBIS 标识						
	IC	A	B	C	D	E	F
时钟同步方法	数据[a]	1	×	0	9	0	0×FF
[a] 即使类"数据"无用.类"寄存器"(使用 标尺=0、单位=255)仍可以使用。							

同步方法：列举

(0) 非同步
(1) 调整到刻
(2) 调整到测量周期
(3) 调整到分钟
(4) 保留
(5) 调整到预设时间
(6) 移动时间

D.1.2.10 正向功率的测量算法

这些值由接口类"数据"范例表述。

正向功率的测量算法	OBIS 标识						
	IC	A	B	C	D	E	F
测量算法	数据	1	×	0	11	1	255

测量算法：列举

(0) 非指定
(1) 仅使用电压和电流的基本量
(2) 使用电压和电流的谐波量
(3) 仅使用电压和电流的直流量
(4) 使用电压和电流的直流量和谐波量

D.1.2.11 正向电能的测量算法

这些值由接口类"数据"范例表述。

正向电能的测量算法	OBIS 标识						
	IC	A	B	C	D	E	F
测量算法	数据	1	×	0	11	2	255

测量算法：列举。

用于正向功率联合值的类似列举在 D.1.2.10 中描述。

D.1.2.12 反向功率的测量算法

这些值由接口类"数据"范例表述。

反向功率的测量算法	OBIS 标识						
	IC	A	B	C	D	E	F
测量算法	数据	1	×	0	11	3	255

测量算法：列举

(0) 无定义
(1) 每相的反向功率(和)采用基本的相电压和相电流计算
(2) 三相反向功率采用三相视在功率和三相正向功率计算
(3) 反向功率(和)采用相视在功率和相正向功率计算

D.1.2.13 反向电能的测量算法

这些值由接口类"数据"范例表述。

反向电能的测量算法	OBIS 标识						
	IC	A	B	C	D	E	F
测量算法	数据	1	×	0	11	4	255

测量算法：列举。

用于反向功率联合值的类似列举在 D.1.2.12 中描述。

D.1.2.14 视在功率的测量算法

这些值由接口类"数据"范例表述。

视在功率的测量算法	OBIS 标识						
	IC	A	B	C	D	E	F
测量算法	数据	1	×	0	11	5	255

测量算法：列举。

(0) 无定义

(1) $S = U \times I$，其中电压：基本量；电流：基本量

(2) $S = U \times I$，其中电压：基本量；电流：所有谐波

(3) $S = U \times I$，其中电压：基本量；电流：所有谐波和直流部分

(4) $S = U \times I$，其中电压：所有谐波；电流：基本量

(5) $S = U \times I$，其中电压：所有谐波；电流：所有谐波

(6) $S = U \times I$，其中电压：所有谐波；电流：所有谐波和直流部分

(7) $S = U \times I$，其中电压：所有谐波和直流部分；电流：基本量

(8) $S = U \times I$，其中电压：所有谐波和直流部分；电流：所有谐波

(9) $S = U \times I$，其中电压：所有谐波和直流部分；电流：所有谐波和直流部分

(10) $S = \sqrt{P^2 + Q^2}$，其中 P：电压和电流基本量；Q：电压和电流基本量。

(11) $S = \sqrt{P^2 + Q^2}$，其中 P：电压和电流中所有谐波量，Q：电压和电流基本量。P 和 Q 为三相参量。

(12) $S = \sqrt{P^2 + Q^2}$，其中 P：电压和电流中所有谐波量和直流部分，Q：电压和电流基本量。P 和 Q 为三相参量。

(13) $S = \sqrt{P^2 + Q^2}$，其中 P：电压和电流基本量，Q：电压和电流基本量。P 和 Q 为单相参量。

(14) $S = \sqrt{P^2 + Q^2}$，其中 P：电压和电流中所有谐波量，Q：电压和电流基本量。P 和 Q 为单相参量。

(15) $S = \sqrt{P^2 + Q^2}$，其中 P：电压和电流中所有谐波量和直流部分，Q：电压和电流基本量。P 和 Q 为单相参量。

D.1.2.15 视在电能的测量算法

这些值由接口类"数据"范例表述。

视在电能的测量算法	OBIS 标识						
	IC	A	B	C	D	E	F
测量算法	数据	1	×	0	11	6	255

测量算法：列举。

用于视在功率联合值的类似列举在 D.1.2.14 中描述。

D.1.2.16 功率因数计算的测量算法

这些值由接口类"数据"范例表述。

功率因数计算的测量算法	OBIS 标识						
	IC	A	B	C	D	E	F
测量算法	数据	1	×	0	11	7	255

测量算法：列举。

(0) 无定义

(1) 功率因数：取决于基本电压和电流的矢量，也可以通过基本视在功率和正向功率，或者其他相应的算法

(2) 功率因数真值，功率因数是由含有谐波的电压乘以电流，也可以通过含有谐波的视在功率和正向功率计算

D.2 OBIS 标识的译码

为辨别同一接口类的不同范例，它们必须具有相同的逻辑_名。COSEM 逻辑_名是在 OBIS 中定义的(见 GB/T 19882.31)。

在 COSEM 环境中，OBIS 编码由作一个八位字节串[6]。每一个八位字节包含相应 OBIS 数值组的 unsigned 值，编码无标记。

如果一个数值组少于 6 的数值组标识，所有未使用数值组需充填值 255。

包含二进制编码值 A 的八位字节 1，在右边四位(A=1,2,...,8)，左边 4 位包含标识系统信息。左边 4 位设为 0，表示 OBIS 标识系统(版本 1)用作逻辑_名。

使用标识系统	八位字节 1 左边 4 位(MSB 左)
OBIS，GB/T 19882	0 0 0 0
预留	0 0 0 1 ... 1 1 1 1

在所有数值组中，一定的选择用法已充分定义，并为未来使用预留空间。数值组 A 用所有可能值充分定义各自预留值。在数值组 B 到 F 中，非定义数值组项存于 127 以上。

编码空间 128～254(0xFE)的使用表述了制造商具体编码。若数值组 B 到 F 中一个高于 127 值被使用，则整个编码都作为制造商特定编码，甚至其他数值组(除数值组 A 外)都无须具有任何由此标准确定的意义。

附 录 E
（资料性附录）
接口类早期版本

先于本标准颁布的接口类版本的定义可以从 DLMS 用户联合会获得。

——通用集(class_id = 7，版本 = 0)

——连接短名(class_id = 12，版本 =0)

参 考 文 献

[1] IEC 61334-6:2000 Distribution automation using distribution line carrier systems - Part 6: A-XDR encoding rule.

[2] ITU Recommendation X.217:1995 Information technology—Open systems Interconnection—service definition for the association control service element.

[3] ITU Recommendation X.227:1995 Information technology—Open systems Interconnection—Connection-oriented protocol for the association control service element: Protocol specification.

[4] IEEE 754:1985 IEEE Standard for Binary Floating-Point Arithmetic.

中 华 人 民 共 和 国
国 家 标 准
自动抄表系统
第 3-2 部分：应用层数据交换协议 接口类
GB/T 19882.32—2007/IEC 62056-62:2002
*
中国标准出版社出版发行
北京复兴门外三里河北街 16 号
邮政编码：100045
网址 www.spc.net.cn
电话：68523946 68517548
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 4.75 字数 136 千字
2008 年 1 月第一版 2008 年 1 月第一次印刷
*
书号：155066·1-30433 定价 46.00 元

GB/T 19882.32-2007

GB/T 19882.32—2007/IEC 62056-62:2002