INTERNATIONAL STANDARD

IEC 60870-6-802

2002

AMENDMENT 1
2005-03

Amendment 1

Telecontrol equipment and systems –

Part 6-802:
Telecontrol protocols compatible with ISO standards and ITU-T recommendations – TASE.2 Object models

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE N

For price, see current catalogue

FOREWORD

This amendment has been prepared by IEC technical committee 57: Power systems management and associated information exchange.

The text of this amendment is based on the following documents:

FDIS	Report on voting
57/740/FDIS	57/745/RVD

Full information on the voting for the approval of this amendment can be found in the report on voting indicated in the above table.

The committee has decided that the contents of this amendment and the base publication will remain unchanged until the maintenance result date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

Page 81

Add, at the end of Annex A, the following new Annex:

Annex B
(normative)

Supplemental object models

B.1 General

This annex expands the number of data objects modelled by TASE.2. It is designed to follow the existing IEC 60870-6-802 document format. This will allow easy integration into the primary document during a maintenance release.

The primary purpose of Telecontrol Application Service Element (TASE.2) is to transfer data between control systems and to initiate control actions. Data is represented by object instances. This part of IEC 60870-6 proposes object models from which to define object instances. The object models represent objects for transfer. The local system may not maintain a copy of every attribute of an object instance.

The object models presented herein are specific to "control centre" or "utility" operations and applications; objects required to implement the TASE.2 protocol and services are found in IEC 60870-6-503. Since needs will vary, the object models presented here provide only a base; extensions or additional models may be necessary for two systems to exchange data not defined within this standard.

It is by definition that attribute values (i.e. data) are managed by the owner (i.e. source) of an object instance. The method of acquiring the values is implementation dependent; therefore accuracy is a local matter.

The notation of the object modelling used for the objects specified in Clause B.2 is defined in IEC 60870-6-503. It should be noted that this part of IEC 60870-6 is based on the TASE.2 services and protocol. To understand the modelling and semantics of this part of IEC 60870-6, some basic knowledge of IEC 60870-6-503 is recommended.

Clause B.2 describes the control centre-specific object models and their application. They are intended to provide information to explain the function of the data.

Clause B.3 defines a set of MMS type descriptions for use in exchanging the values of instances of the defined object models. It is important to note that not all attributes of the object models are mapped to types. Some attributes are described simply to define the processing required by the owner of the data and are never exchanged between control centres. Other attributes are used to determine the specific types of MMS variables used for the mapping, and therefore do not appear as exchanged values themselves. A single object model may also be mapped onto several distinct MMS variables, based on the type of access and the TASE.2 services required.

Clause B.4 describes the mapping of instances of each object type MMS variables and named variable lists for implementing the exchange.

Clause B.5 describes device-specific codes and semantics to be used with the general objects.

Clause B.6 is the standards conformance table.

B.2 Object Models (see clause 5 of this part of IEC 60870-6)

B.2.1 General

Object models are required for various functions within a system. This Clause delineates abstract object models based on functionality. Object models within one functional area may be used in another functional area.

B.2.2 Supervisory control and data acquisition

The object models in this Clause are derived from the historical perspective of Supervisory Control and Data Acquisition (SCADA) systems. The following text presents the context within which the object models are defined.

Fundamental to SCADA systems are two key functions: control and indication. The control function is associated with the output of data whereas the indication function is associated with the input of data.

The previous identified functions within SCADA systems are mapped to point equipment (point). The primary attribute of a point is the data value. SCADA systems define three types of data for points: analog, digital and state.

The association of one or more points together is used to represent devices. For example, a breaker device may be represented by a control point and an indication point. The control point represents the new state that one desires for the breaker device. The indication point represents the current state of the breaker device. For SCADA to SCADA data exchange (e.g. control centre to control centre, control centre to SCADA master etc.), additional data is often associated with point data. Quality of point data is often exchanged to define whether the data is valid or not. In addition, for data that may be updated from alternate sources, quality often identifies the alternate source. Select-Before-Operate control is associated with Control Points for momentary inhibiting access except from one source. Two other informative data values are: time stamp and change of value counter. The time stamp, when available, details when a data value last changed. The change of value counter, when available, details the number of changes to the value.

From the context presented, the primary object models required are: Indication Point, and Control Point. The attributes Point Value, Quality, Select-Before-Operate, Time Stamp, and Change of Value Counter are required to meet the desired functionality for data exchange. The Indication Point and Control Point models may be logically combined to a single model to represent a device which implements a control function with a status indication as to its success/failure. The combined logical model will result in the same logical attributes, and map onto the same MMS types as the independent models.

B.2.3 IndicationPoint object

An IndicationPoint object represents an actual input point.

Object: **IndicationPoint** (Read Only)
Key Attribute: PointName
Attribute: PointType (REAL, STATE, DISCRETE, STATESUPPLEMENTAL)
Constraint PointType=REAL
Attribute: PointRealValue
Constraint PointType=STATE
Attribute:PointStateValue
Constraint PointType=DISCRETE
Attribute: PointDiscreteValue
Constraint PointType= STATESUPPLEMENTAL
Attribute:PointStateSupplementalValue
Attribute: QualityClass: (QUALITY, NOQUALITY)
Constraint: QualityClass = QUALITY
Attribute: Validity (VALID, HELD, SUSPECT, NOTVALID)
Attribute: CurrentSource (TELEMETERED, CALCULATED, ENTERED, ESTIMATED)
Attribute: NormalSource (TELEMETERED, CALCULATED, ENTERED, ESTIMATED)
Attribute: NormalValue (NORMAL,ABNORMAL)
Attribute: TimeStampClass: (TIMESTAMP, TIMESTAMPEXTENDED, NOTIMESTAMP)
Constraint: TimeStampClass = TIMESTAMP
Attribute: TimeStamp
Attribute: TimeStampQuality: (VALID, INVALID)
Constraint: TimeStampClass = TIMESTAMPEXTENDED
Attribute: TimeStampExtended
Attribute: TimeStampQuality: (VALID, INVALID)
Attribute: COVClass: (COV, NOCOV)
Constraint: COVClass = COV
Attribute: COVCounter

PointName
The PointName attribute uniquely identifies the object.

PointType
The PointType attribute identifies the type of input point, and must be one of the following: REAL, STATE, DISCRETE, STATESUPPLEMENTAL.

PointRealValue
The current value of the IndicationPoint, if the PointType attribute is REAL.

PointStateValue
The current value of the IndicationPoint, if the PointType attribute is STATE.

PointDiscreteValue
The current value of the IndicationPoint, if the PointType attribute is DISCRETE.

PointStateSupplementalValue
The current value of the IndicationPoint, if the PointType attribute is STATESUPPLEMENTAL.

QualityClass
The QualityClass has the value QUALITY if the object instance has any of the quality attributes (Validity, CurrentSource, or NormalValue), and takes the value NOQUALITY if none of the attributes are present.

Validity
The Validity attribute specifies the validity or quality of the PointValue data it is associated with. These are based on the source system's interpretation as follows:

Validity	Description
VALID	Data value is valid
HELD	Previous data value has been held over. Interpretation is local
SUSPECT	Data value is questionable. Interpretation is local
NOTVALID	Data value is not valid

CurrentSource
The CurrentSource attribute specifies the current source of the PointValue data it is associated with as follows:

CurrentSource	Description
TELEMETERED	The data value was received from a telemetered site
CALCULATED	The data value was calculated based on other data values
ENTERED	The data value was entered manually
ESTIMATED	The data value was estimated (State Estimator, etc.)

NormalSource
The NormalSource attribute specifies the normal source of the PointValue data it is associated with as follows:

NormalSource	Description
TELEMETERED	The data value is normally received from a telemetered site
CALCULATED	The data value is normally calculated based on other data values
ENTERED	The data value is normally entered manually
ESTIMATED	The data value is normally estimated (State Estimator, etc.)

NormalValue
The NormalValue attribute reports whether value of the PointValue attribute is normal. Only one bit is set, it is defined as follows:

NormalValue	Description
NORMAL	The point value is that which has been configured as normal for the point
ABNORMAL	The point value is not that which has been configured as normal for the point

TimeStampClass
The TimeStampClass attribute has the value TIMESTAMP or TIMESTAMPEXTENDED if the IndicationPoint is time stamped, and has the value NOTIMESTAMP if the IndicationPoint contains no TimeStamp attribute.

TimeStamp
The TimeStamp attribute provides a time stamp (with a minimum resolution of one second) of when the value (attribute PointRealValue, PointStateValue, PointDiscreteValue, or PointStateSupplementalValue) of the IndicationPoint was last changed. It is set at the earliest possible time after collection of the IndicationPoint value from the end device.

TimeStampExtended

The TimeStampExtended attribute provides a time stamp (with a resolution of one millisecond) of when the value (attribute PointRealValue, PointStateValue, PointDiscreteValue, or PointStateSupplementalValue) of the IndicationPoint was last changed. It is set at the earliest possible time after collection of the IndicationPoint value from the end device.

TimeStampQuality
The TimeStampQuality attribute has the value VALID if the current value of the TimeStamp attribute contains the time stamp of when the value was last changed, and has the value INVALID at all other times.

COVClass
The COVClass (**C**hange **O**f **V**alue Counter) attribute has the value COV if the IndicationPoint contains a COVCounter attribute, otherwise it has the value NOCOV.

COVCounter
The COVCounter attribute specifies the number of times the value (attribute PointRealValue, PointStateValue, PointDiscreteValue, or PointStateSupplementalValue) of the IndicationPoint has changed. It is incremented each time the owner sets a new value for the IndicationPoint.

B.3 MMS types for object exchange (see clause 6 of this part of IEC 60870-6)

B.3.1 General

This Clause defines the MMS types to be used within TASE.2 for exchanging standard objects. The mapping of the objects onto these types is defined in Clause B.4. The MMS type definitions are defined in terms of ASN.1 value notation, following the MMS grammar for Data as defined in ISO/IEC 9506-2.

Throughout this Clause, all field widths specified are maximum field widths. The process of ASN.1 encoding used within MMS may reduce the actual transmitted widths to the minimum required to represent the value being transmitted.

B.3.2 Supervisory control and data acquisition types - IndicationPoint type descriptions

The following foundation types are referenced in complex IndicationPoint Type Descriptions:

Data_Real floating-point: { format-width 32, exponent-width 8 }

Data_State bit-string:

```
    {
        State_hi[0],
        State_lo[1],
        Validity_hi[2],
        Validity_lo[3],
        CurrentSource_hi[4],
        CurrentSource_lo[5],
        NormalValue[6],
        TimeStampQuality[7]
    }
```

Data_Discrete integer {width 32 }

Data_StateSupplemental bit-string:

```
    {
        State_hi[0],
        State_lo[1],
        Tag_hi[2],
        Tag_lo[3],
        ExpectedState_hi[4],
        ExpectedState_lo[5],
        Reserved[6],
        Reserved[7]
    }
```

Data_Flags bit-string:

```
    {
        unused[0],
        unused[1],
        Validity_hi[2],
        Validity_lo[3],
        CurrentSource_hi[4],
        CurrentSource_lo[5],
        NormalValue[6],
        TimeStampQuality[7]
    }
```

Data_TimeStamp GMTBasedS

Data_TimeStampExtended **TimeStampExtended**

COV_Counter unsigned { width 16 }

The following complex types are used in transferring IndicationPoint object values:

```
Data_RealQ STRUCTURE
{
        COMPONENT Value           Data_Real,
        COMPONENT Flags           Data_Flags
}
Data_StateQ STRUCTURE

{
        COMPONENT Value           Data_State,
        COMPONENT Flags           Data_Flags
}


Data_DiscreteQ STRUCTURE
{
        COMPONENT Value           Data_Discrete,
        COMPONENT Flags           Data_Flags
}
Data_StateSupplementalQ STRUCTURE

{
        COMPONENT Value           Data_StateSupplemental,
        COMPONENT Flags           Data_Flags
}
Data_RealQTimeTag STRUCTURE
{
        COMPONENT Value           Data_Real,
        COMPONENT TimeStamp       Data_TimeStamp,
        COMPONENT Flags           Data_Flags

}

Data_StateQTimeTag STRUCTURE
{
        COMPONENT TimeStamp       Data_TimeStamp,
        COMPONENT Flags           Data_State
}
Data_DiscreteQTimeTag STRUCTURE
{
        COMPONENT Value           Data_Discrete,
        COMPONENT TimeStamp       Data_TimeStamp,
        COMPONENT Flags           Data_Flags
}
Data_ StateSupplementalQTimeTag STRUCTURE
{
        COMPONENT Value           Data_StateSupplemental,
        COMPONENT TimeStamp       Data_TimeStamp,
        COMPONENT Flags           Data_Flags
}
```

```
Data_RealExtended STRUCTURE
{
        COMPONENT Value           Data_Real,
        COMPONENT TimeStamp       Data_TimeStamp,
        COMPONENT Flags           Data_Flags,
        COMPONENT COV             COVCounter
}
Data_StateExtended STRUCTURE
{
        COMPONENT TimeStamp       Data_TimeStamp,
        COMPONENT Flags           Data_State,
        COMPONENT COV             COVCounter
}
Data_DiscreteExtended STRUCTURE
{
        COMPONENT Value           Data_Discrete,
        COMPONENT TimeStamp       Data_TimeStamp,
        COMPONENT Flags           Data_Flags,
        COMPONENT COV             COVCounter
}
Data_StateSupplementalExtended STRUCTURE
{
        COMPONENT Value           Data_StateSupplemental,
        COMPONENT TimeStamp       Data_TimeStamp,
        COMPONENT Flags           Data_Flags,
        COMPONENT COV             COVCounter
}
Data_RealQTimeTagExtended STRUCTURE
{
        COMPONENT Value           Data_Real,
        COMPONENT TimeStamp       Data_TimeStampExtended,
        COMPONENT Flags           Data_Flags
}

Data_StateQTimeTagExtended STRUCTURE
{
        COMPONENT TimeStamp       Data_TimeStampExtended,
        COMPONENT Flags           Data_State
}
Data_DiscreteQTimeTagExtended STRUCTURE
{
        COMPONENT Value           Data_Discrete,
        COMPONENT TimeStamp       Data_TimeStampExtended,
        COMPONENT Flags           Data_Flags
}
Data_State_SupplementalQTimeTagExtended STRUCTURE
{
        COMPONENT Value           Data_StateSupplemental,
        COMPONENT TimeStamp       Data_TimeStampExtended,
        COMPONENT Flags           Data_Flags
}
```

```
IndicationPointConfig STRUCTURE
{
        COMPONENT PointType         integer { width 8, range 0 .. 2 },
        COMPONENT QualityClass      integer { width 8, range 0 .. 1 },
        COMPONENT NormalSource      integer { width 8, range 0 .. 3 },
        COMPONENT TimeStampClass    integer { width 8, range 0 .. 1 },
        COMPONENT COVClass          integer { width 8, range 0 .. 1 }
}
```

B.4 Mapping of Object Models to MMS Types (see clause 7 of this part of IEC 60870-6)

This Clause defines the mapping of each object attributes onto MMS. In general, most objects are represented by one or more MMS Named Variables of the predefined TASE.2 types from Clause B.3.

PointName
Maps to an MMS variable identifier (either VMD specific or Domain specific)

PointType
Used in selecting the named type of the variable. If COVClass is NOCOV, the type of the MMS variable is selected according to the following criteria:

PointType	QualityClass	TimeStampClass	Map to type:
REAL	NOQUALITY	NOTIMESTAMP	**Data_Real**
STATE	NOQUALITY	NOTIMESTAMP	**Data_State**
DISCRETE	NOQUALITY	NOTIMESTAMP	**Data_Discrete**
STATE SUPPLEMENTAL	NOQUALITY	NOTIMESTAMP	**Data_StateSupplemental**
REAL	QUALITY	NOTIMESTAMP	**Data_RealQ**
STATE	QUALITY	NOTIMESTAMP	**Data_StateQ**
DISCRETE	QUALITY	NOTIMESTAMP	**Data_DiscreteQ**
STATE SUPPLEMENTAL	QUALITY	NOTIMESTAMP	**Data_StateSupplementalQ**
REAL	QUALITY	TIMESTAMP	**Data_RealQTimeTag**
STATE	QUALITY	TIMESTAMP	**Data_StateQTimeTag**
DISCRETE	QUALITY	TIMESTAMP	**Data_DiscreteQTimeTag**
STATE SUPPLEMENTAL	QUALITY	TIMESTAMP	**Data_StateSupplementalQTimeTag**
REAL	QUALITY	TIMESTAMPEXTENDED	**Data_RealQTimeTagExtended**
STATE	QUALITY	TIMESTAMPEXTENDED	**Data_StateQTimeTagExtended**
DISCRETE	QUALITY	TIMESTAMPEXTENDED	**Data_DiscreteQTimeTagExtended**
STATE SUPPLEMENTAL	QUALITY	TIMESTAMPEXTENDED	**Data_StateSupplementalQTimeTag Extended**

If COVClass is COV, the following criteria are used:

PointType	Map to type:
REAL	**Data_RealExtended**
STATE	**Data_StateExtended**
DISCRETE	**Data_DiscreteExtended**
STATESUPPLEMENTAL	**Data_StateSupplementalExtended**

The PointType attribute may optionally be mapped to the **PointType** component of an MMS named variable of type **IndicationPointConfig** with the following interpretation: 0=STATE, 1=DISCRETE, 2=REAL, 3=StateSupplemental.

PointRealValue
If present, maps to either the value of an MMS variable of type **Data_Real** (if QualityClass and TimeStampClass are NOQUALITY, NOTIMESTAMP) or to the **Value** COMPONENT of the MMS variable.

PointStateValue
If present, maps to either the value of an MMS variable of type **Data_State** (if QualityClass and TimeStampClass are NOQUALITY, NOTIMESTAMP) or to bits **State_hi** and **State_lo** of the **Flags** COMPONENT of the MMS variable.

PointDiscreteValue
If present, maps to either the value of the MMS variable of type **Data_Discrete** (if QualityClass and TimeStampClass are NOQUALITY, NOTIMESTAMP) or to the **Value** COMPONENT of the MMS variable.

PointStateSupplementalValue
If present, maps to either the value of the MMS variable of type **Data_StateSupplemental** (if QualityClass and TimeStampClass are NOQUALITY, NOTIMESTAMP) or to the **Value** COMPONENT of the MMS variable.

QualityClass

Used in selecting the named type of the variable (see above). The QualityClass attribute may also be optionally mapped to the **QualityClass** component of an MMS Named Variable of type **IndicationPointConfig** with the following interpretation: NOQUALITY=0, QUALITY=1.

Validity
If present, maps to bits 2 and 3 (**Validity_hi**, **Validity_lo**) of the **Flags** COMPONENT with the following values: VALID = 0, HELD=1, SUSPECT=2, NOTVALID=3.

CurrentSource
If present, maps to bits 4 and 5 (**CurrentSource_hi**, **CurrentSource_lo**) of the **Flags** COMPONENT with the following values: TELEMETERED=0, CALCULATED=1, ENTERED=2, ESTIMATED=3.

NormalSource
The NormalSource attribute may be optionally mapped to the **NormalSource** component of an MMS Named Variable of type **IndicationPointConfig** with the following interpretation: TELEMETERED=0, CALCULATED=1, ENTERED=2, ESTIMATED=3.

NormalValue
If present, maps to bit 6 (**NormalValue**) of the **Flags** COMPONENT with the following values: NORMAL=0, ABNORMAL=1.

TimeStampClass
Used in selecting the named type of the variable (see above). The TimeStampClass attribute may also be optionally mapped to the **TimeStampClass** component of an MMS Named Variable of type **IndicationPointConfig** with the following interpretation: NOTIMESTAMP=0, TIMESTAMP=1, TIMESTAMPEXTENDED=2.

TimeStamp
If present, maps to the **TimeStamp** COMPONENT.

TimeStampQuality
If present, maps to bit 7 (**TimeStampQuality**) of the **Flags** COMPONENT with the following values: VALID=0, INVALID=1.

COVClass
Used in selecting the named type of the variable (see above). The COVClass attribute may also be optionally mapped to the **COVClass** component of an MMS named variable of type **IndicationPointConfig** with the following interpretation: NOCOV=0, COV=1.

COVCounter
If present, maps to an MMS variable of type **COV_Counter**.

B.5 Use of supervisory control objects (see clause 8 of this part of IEC 60870-6)

B.5.1 General

The supervisory control object models (IndicationPoint and ControlPoint) are generic in nature in that more than one type of device can be represented with these object models. This Clause provides the allowable uses of these object models to represent real devices. However, it is recognized that this list may not be exhaustive. If a new device is defined in the future that requires different semantics (i.e., interpretations) that cannot be mapped into the existing list, then implementers can add new semantics as long as they do not conflict with the existing semantics assigned to values in this section.

B.5.2 Use of IndicationPoint model

The IndicationPoint model is used to represent arbitrary data input from devices such as status points (PointType=STATE, PointType=STATESUPPLEMENTAL, or PointType=DISCRETE), analog points (PointType=REAL) and counter values (PointType=DISCRETE), and Transformer step positions (PointType=DISCRETE).

PointType STATE and STATESUPPLEMENTAL are recommended for status points (single or double) with up to three states whereas PointType DISCRETE is recommended for status points with more than three states. The following PointValue values of type STATE are used to represent specific device positions:

00	01	10	11	Device
Between	Tripped	Closed	Invalid	Disconnector
Between	Off	On	Invalid	Disconnector
Invalid	Off	On	Invalid	Breaker
Invalid	Auto	Manual	Invalid	
Invalid	Normal	Alarm	Invalid	
Invalid	Local	Remote	Invalid	
Invalid	Raise	Lower	Invalid	
Invalid	Not Ready	Ready	Invalid	
Invalid	Offline	Available	Invalid	