ANSI/ASHRAE Addendum 135*b*-2000

Addendum to ANSI/ASHRAE Standard 135-1995

ASHRAE® STANDARD

Addendum to BACnet® – A Data Communication Protocol for Building Automation and Control Networks

Approved by the ASHRAE Standards Committee February 5, 2000; by the ASHRAE Board of Directors February 10, 2000; and by the American National Standards Institute April 25, 2000.

This standard is under continuous maintenance by a Standing Standard Project Committee (SSPC) for which the Standards Committee has established a documented program for regular publication of addenda or revisions, including procedures for timely, documented, consensus action on requests for change to any part of the standard. The change submittal form, instructions, and deadlines are given at the back of this document and may be obtained in electronic form from ASHRAE's Internet Home Page, *http://www.ashrae.org*, or in paper form from the Manager of Standards. The latest edition of an ASHRAE Standard and printed copies of a public review draft may be purchased from ASHRAE Customer Service, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. E-mail: *orders@ashrae.org*. Fax: 404-321-5478. Telephone: 404-636-8400 (worldwide), or toll free 1-800-527-4723 (for orders in U.S. and Canada).

ISSN 1041-2336

Approved American National Standard ANSI

AMERICAN SOCIETY OF HEATING, REFRIGERATING AND AIR-CONDITIONING ENGINEERS, INC.

1791 Tullie Circle, NE • Atlanta, GA 30329

SPECIAL NOTE

This American National Standard (ANS) is a national voluntary consensus standard developed under the auspices of the American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE). Consensus is defined by the American National Standards Institute (ANSI), of which ASHRAE is a member and which has approved this standard as an ANS, as “substantial agreement reached by directly and materially affected interest categories. This signifies the concurrence of more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that an effort be made toward their resolution.” Compliance with this standard is voluntary until and unless a legal jurisdiction makes compliance mandatory through legislation.

ASHRAE obtains consensus through participation of its national and international members, associated societies, and public review.

ASHRAE Standards are prepared by a Project Committee appointed specifically for the purpose of writing the Standard. The Project Committee Chair and Vice-Chair must be members of ASHRAE; while other committee members may or may not be ASHRAE members, all must be technically qualified in the subject area of the Standard. Every effort is made to balance the concerned interests on all Project Committees.

The Manager of Standards of ASHRAE should be contacted for:

a. interpretation of the contents of this Standard,
b. participation in the next review of the Standard,
c. offering constructive criticism for improving the Standard,
d. permission to reprint portions of the Standard.

ASHRAE INDUSTRIAL ADVERTISING POLICY

ASHRAE Standards and Guidelines are established to assist industry and the public by offering a uniform method of testing for rating purposes, by suggesting safe practices in designing and installing equipment, by providing proper definitions of this equipment, and by providing other information that may serve to guide the industry. The creation of ASHRAE Standards and Guidelines is determined by the need for them, and conformance to them is completely voluntary.
In referring to this Standard or Guideline and in marketing of equipment and in advertising, no claim shall be made, either stated or implied, that the product has been approved by ASHRAE.

Foreword

The purpose of this addendum is to add a number of independent substantive changes to the BACnet standard. These modifications are the result of change proposals made pursuant to the continuous maintenance procedures contained in the *Manual for Processing ASHRAE Standards* and of deliberations within Standing Standard Project Committee 135. The changes are summarized below.

135*b*-1. Inconsistencies are eliminated in the definitions of the Analog and Binary Value object types, p. 1.

135*b*-2. Any device that receives and executes UnconfirmedEventNotification service requests must support programmable process identifiers, p. 4.

135*b*-3. Modify each event-generating object type to contain the last timestamp for each acknowledgeable transition, p. 5.

135*b*-4. Modify the Notification Class object by requiring that the 'Notification Class' property be equivalent to the instance number of the Notification Class object, p. 8.

135*b*-5. Modify the Event Notification services to make the 'To State' parameter mandatory for notifications of type ACK_NOTIFICATION, p. 9.

135*b*-6. A new BACnetDeviceObjectPropertyReference production is added and its use in the Event Enrollment and Schedule object types is specified, p. 11.

135*b*-7. Add a Multi-state Value object type, p. 14.

135*b*-8. Add an Averaging object type, p. 21.

135*b*-9. Change all 'Process Identifier' properties and parameters to Unsigned32, p. 27.

135*b*-10. Change the Multi-state Input object type to correct flaws related to fault detection and reporting and achieve consistency with the new Multi-state Value object type, p. 29.

135*b*-11. Add a Protocol_Revision property to the Device object type, p. 31.

135*b*-12. The File object type is changed to allow truncation and partial deletion operations, p. 34.

135*b*-13. A new ReadRange service is added to permit reading a range of data items from a property whose datatype is a list or array of lists, p. 36.

135*b*-14. A new UTCTimeSynchronization service is introduced and related changes are made to properties in the Device object type, p. 43.

135*b*-15. Add a Trend Log object type, p. 46.

135*b*-16. The UnconfirmedCOVNotification service is extended to allow notifications without prior subscription as a means of distributing globally important data to a potentially large number of recipients, p. 57.

135*b*-17. Add eight new BACnet engineering units, p. 58.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-1995 is indicated through the use of *italics* while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout. These instances are 135*b*-7, 135*b*-8, 135*b*-13 and 135*b*-15.

135*b*-1. Inconsistencies are eliminated in the definitions of the Analog and Binary Value object types.

Analog and Binary Value objects may sometimes be used to represent "output values" in the sense that their use is intended to act as a place that may be commanded with a value that is internally used as a commanded state. In these instances it may be desirable for an AV or BV object to behave similarly to AO and BO objects in that the Present_Value property is controlled by the prioritization mechanism. In these cases, the Priority_Array and Relinquish_Default properties need to be present to implement the prioritization.

There are other instances when an AV or BV object may be used to represent a status or internally computed value, and so the objects behave more like their AI and BI counterparts.

Due to the "dual" nature of potential AV and BV behavior, it is desirable to eliminate the requirement that the Present_Value properties of these objects be writable, as was formerly the case with AV, and further to eliminate the requirement that the objects be commandable.

To achieve these ends, the decision is to make Present_Value properties for both AV and BV objects be conformance code R. At the implementor's discretion, the Present_Value may be optionally writable. If writable, the Present_Value property may or may not be in addition a commandable property. If commandable, then it must of course also be writable and the optional properties Priority_Array and Relinquish_Default must also be present.

[Change **Table 12-3**, p. 148]

Property Identifier	Property Datatype	Conformance Code
Present_Value	REAL	*R*[4]
Priority_Array	BACnetPriorityArray	O[1]
Relinquish_Default	REAL	O[1]

[1] *If Present_Value is commandable, then both of these properties shall be present.*
[4] *If Present_Value is commandable, then it is required to also be writable. This property is required to be writable when Out_Of_Service is TRUE.*

[Change **12**, p. 149ff]

12.3.4 Present_Value ~~(Commandable)~~

This property, of type REAL, indicates the current value, in engineering units, of the analog value (see 12.3.10). ~~The optional priority array must be present for the Present_Value to be commandable.~~ *Present_Value shall be optionally commandable. If Present_Value is commandable for a given object instance, then the Priority_Array and Relinquish_Default properties shall also be present for that instance. The Present_Value property shall be writable when Out_Of_Service is TRUE (see 12.3.9).*

12.3.9 Out_Of_Service

The Out-Of-Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the Present_Value of the Analog Value object is prevented from being modified by software local to the BACnet device in which the object resides. When Out_Of_Service is TRUE, the Present_Value property may ~~still~~ be written to freely. If the Priority_Array and Relinquish_Default properties are present, then writing to the Present_Value property shall be controlled by the BACnet command prioritization mechanism. See Clause 19.

12.3.11 Priority_Array

This property is a read-only array that contains prioritized commands that are in effect for this object. See Clause 19 for a description of the prioritization mechanism. If either the Priority_Array property or the Relinquish_Default property are present, then both of them shall be present. *If Present_Value is commandable, then Priority_Array and Relinquish_Default shall both be present.*

12.3.12 Relinquish_Default

This property is the default value to be used for the Present_Value property when all command priority values in the Priority_Array property have a NULL value. See Clause 19. If either the Relinquish_Default property or the Priority_Array property are present, then both of them shall be present. *If Present_Value is commandable, then Priority_Array and Relinquish_Default shall both be present.*

[Change footnotes to **Table 12-8**, p. 165]

Property Identifier	Property Datatype	Conformance Code
Present_Value	BACnetBinaryPV	R[1]
Priority_Array	BACnetPriorityArray	O[5]
Relinquish_Default	REAL	O[5]

[1] *If Present_Value is commandable, then it is required to also be writable. This property is required to be writable when Out_Of_Service is TRUE.*

[5] *If Present_Value is commandable, then both of these properties shall be present.*

[Change **12.6**, p. 165ff]

12.6.4 Present_Value ~~(Commandable)~~

This property, of type BACnetBinaryPV, reflects the logical state of the Binary Value. The logical state shall be either INACTIVE or ACTIVE. ~~The optional priority array must be present for the Present_Value to be commandable.~~ *Present_Value shall be optionally commandable. If Present_Value is commandable for a given object instance, then the Priority_Array and Relinquish_Default properties shall also be present for that instance.* The Present_Value property shall be writable when Out_Of_Service is TRUE (see 12.6.9).

12.6.9 Out_Of_Service

The Out-Of-Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the Present_Value of the Binary Value object is prevented from being modified by software local to the BACnet device in which the object resides. When Out_Of_Service is TRUE, the Present_Value property may ~~still~~ be written to freely. If the Priority_Array and Relinquish_Default properties are present, then writing to the Present_Value property shall be controlled by the BACnet command prioritization mechanism. See Clause 19.

12.6.19 Priority_Array

This property is a read-only array that contains prioritized commands that are in effect for this object. See Clause 19 for a description of the prioritization mechanism. *If either the Relinquish_Default property or the Priority_Array property are present, then both of them shall be present. If Present_Value is commandable, then Priority_Array and Relinquish_Default shall both be present and Present_Value shall be required to be writable.* ~~If one of the optional properties Priority_Array or Relinquish_Default property is present, then both of these properties shall be present and the Present_Value property is also required to be writable.~~

12.6.20 Relinquish_Default

This property is the default value to be used for the Present_Value property when all command priority values in the Priority_Array property have a NULL value. See Clause 19. *If either the Relinquish_Default property or the Priority_Array property are present, then both of them shall be present. If Present_Value is commandable, then Priority_Array and Relinquish_Default shall both be present and Present_Value shall be required to be writable.* ~~If one of the optional properties Priority_Array or Relinquish_Default property is present, then both of these properties shall be present and the Present_Value property is also required to be writable.~~

[Change **Annex C**, p. 412]

ANALOG-VALUE ::= SEQUENCE {

```
. . .
                    priority-array    [87]      BACnetPriorityArray OPTIONAL,
                    relinquish-default          [104]     REAL OPTIONAL,
. . .
```

135*b*-2. Any device that receives and executes UnconfirmedEventNotification service requests must support programmable process identifiers.

The standard currently allows multicast and broadcast addresses as destinations for the UnconfirmedEventNotification service. This may cause problems for devices that receive notifications they neither want nor expect.

This addendum requires any device that will receive and execute UnconfirmedEventNotification service requests to support programmable process identifiers. This will allow broadcast and multicast process identifiers to be assigned on a per installation basis.

[Change **13.12**, p. 248]

13.12 UnconfirmedEventNotification Service

The UnconfirmedEventNotification service is used by a notification-server to notify a remote device that an event has occurred. Its purpose is to notify recipients that an event has occurred, but confirmation that the notification was received is not required. Applications that require confirmation that the notification was received by the remote device should use the ConfirmedEventNotification service. The fact that this is an unconfirmed service does not mean it is inappropriate for notification of alarms. Events of type Alarm may require a human acknowledgment that is conveyed using the AcknowledgeAlarm service. See 13.5. Thus, using an unconfirmed service to announce the alarm has no effect on the ability to confirm that an operator has been notified. *Any device that executes this service shall support programmable process identifiers to allow broadcast and multicast 'Process Identifier' parameters to be assigned on a per installation basis.*

135*b*-3. Modify each event-generating object type to contain the last timestamp for each acknowledgeable transition.

A device is unable to acknowledge events/alarms that are generated when the device is offline or not connected to the network. The problem is due to the AcknowledgeAlarm service requiring a timestamp for the transition that is to be acknowledged and that timestamp only being available from the EventNotification service.

This proposal modifies each event-generating object to contain the last timestamp for each acknowledgeable transition.

[The following changes to tables are to add the new timestamp property. There are multiple changes shown because the footnote reference number is different in each.]

[Change **Tables 12-1**, p. 138 and **12-3**, p. 148]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Event_Time_Stamps	*BACnetARRAY [3] of BACnetTimeStamp*	*O*[3]

[Change **Tables 12-2**, p. 143 and **12-17**, p. 197]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Event_Time_Stamps	*BACnetARRAY [3] of BACnetTimeStamp*	*O*[2]

[Change **Tables 12-4**, p. 153 and **12-16**, p. 190]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Event_Time_Stamps	*BACnetARRAY [3] of BACnetTimeStamp*	*O*[5]

[Change **Table 12-6**, p. 159]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Event_Time_Stamps	*BACnetARRAY [3] of BACnetTimeStamp*	*O*[4]

[Change **Table 12-8**, p. 165]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Event_Time_Stamps	*BACnetARRAY [3] of BACnetTimeStamp*	*O*[6]

[Change **Table 12-12**, p. 181]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Event_Time_Stamps	*BACnetARRAY [3] of BACnetTimeStamp*	*R*

[Change **Table 12-18**, p. 201]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Event_Time_Stamps	*BACnetARRAY [3] of BACnetTimeStamp*	*O*[1]

[Add clauses to describe the Event_Time_Stamps in each event-generating object type (excluding the Event Enrollment object type): **12.1.26**, p. 142, **12.2.27**, p. 147, **12.3.23**, p. 152, **12.4.25**, p. 158, **12.5.29**, p. 164, **12.6.27**, p. 169, **12.13.36**, p. 196, **12.14.20**, p. 200, **12.15.21**, p. 204]

12.xx.yy Event_Time_Stamps

This optional property, of type BACnetARRAY[3] of BACnetTimeStamp, shall convey the times of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events, respectively. Time stamps of type Time or Date shall have 'FF' in each octet and Sequence number time stamps shall have the value 0 if no event notification of that type has been generated since the object was created. This property is required if intrinsic reporting is supported by this object.

[Add clause to describe the Event_Time_Stamps in the Event Enrollment object type, **12.10.17**, p. 185]

12.10.17 Event_Time_Stamps

This property, of type BACnetARRAY[3] of BACnetTimeStamp, shall convey the times of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events, respectively. Time stamps of type Time or Date shall have 'FF' in each octet and Sequence number time stamps shall have the value 0 if no event notification of that type has been generated since the object was created.

[In **21**, add entry to BACnetPropertyIdentifier, p. 376]

```
event-state                        (36),
event-time-stamps                  (130),
event-type                         (37),

…
-- see event-time-stamps           (130),
…
```

[In **ANNEX C**, change ANALOG-INPUT, p. 411, ANALOG-OUTPUT, p. 411, ANALOG-VALUE, p. 412, BINARY-INPUT, p. 412, BINARY-OUTPUT, p. 413, BINARY-VALUE, p. 413, LOOP, p. 415, MULTI-STATE-INPUT, p. 416, MULTI-STATE-OUTPUT, p. 417 by adding an optional event-time-stamps property]

```
…,
event-time-stamps        [130]  SEQUENCE OF BACnetTimeStamp OPTIONAL,
– accessed as a BACnetARRAY
}
```

[In **ANNEX C**, change EVENT-ENROLLMENT, p. 415, by adding a required event-time-stamps property]

```
…,
event-time-stamps        [130]  SEQUENCE OF BACnetTimeStamp,
– accessed as a BACnetARRAY
}
```

[Change **D.1**, p. 419, by adding these lines at the end of the example]

Property: *Event_Time_Stamps =* *((23-MAR-95,18:50:21.2),*
(-*-*,*:*:*.*),*
(23-MAR-95,19:01:34.0))

[Change **D.4**, p. 420, by adding these lines at the end of Example 1]

Property: *Event_Time_Stamps =* *((23-MAR-95,18:50:21.2),*
(-*-*,*:*:*.*),*
(21-MAR-95,01:02:03.0))

[Change **D.4**, p. 421, by adding these lines at the end of Example 2]

Property: *Event_Time_Stamps =* *((21-MAR-95,01:09:34.0),*
(23-MAR-95,19:01:34.0),
(21-MAR-95,01:11:21.2))

[Change **D.10**, p. 426, by adding these lines at the end of Example 1]

Property: *Event_Time_Stamps =* *((23-MAR-95,18:50:21.2),*
(-*-*,*:*:*.*),*
(21-MAR-95,01:02:03.0))

[Change **D.10**, p. 427, by adding these lines at the end of Example 2]

Property: *Event_Time_Stamps =* *((23-MAR-95,18:50:21.2),*
(-*-*,*:*:*.*),*
(23-MAR-95,19:01:34.0))

[Change **D.10**, p. 427, by adding these lines at the end of Example 3]

Property: *Event_Time_Stamps =* *((23-MAR-95,18:50:21.2),*
(-*-*,*:*:*.*),*
(23-MAR-95,19:01:34.0))

[Change **D.13**, p. 429, by adding these lines at the end of the example]

Property: *Event_Time_Stamps =* *((23-MAR-95,18:50:21.2),*
(-*-*,*:*:*.*),*
(23-MAR-95,19:01:34.0))

[Change **D.14**, p. 430, by adding these lines at the end of Example 1]

Property: *Event_Time_Stamps =* *((23-MAR-95,18:50:21.2),*
(-*-*,*:*:*.*),*
(23-MAR-95,19:01:34.0))

[Change **D.15**, p. 431, by adding these lines at the end of Example 1]

Property: *Event_Time_Stamps =* *((23-MAR-95,18:50:21.2),*
(-*-*,*:*:*.*),*
(21-MAR-95,01:02:03.0))

135*b*-4. Modify the Notification Class object by requiring that the ’Notification Class’ property be equivalent to the instance number of the Notification Class object.

The Notification Class object referenced by an event-generating object cannot be easily or efficiently determined by a device other than the device containing the event-generating object.

This proposal modifies the Notification Class object by requiring that the ’Notification Class’ property be equivalent to the instance number of the Notification Class object. This will have the effect that the ’Notification_Class’ property of event-generating objects will now explicitly, instead of implicitly, reference a Notification Class object.

[Change **12.16.5**]

12.16.5 Notification_Class

This property, of type Unsigned, shall indicate the numeric value of this notification class *and shall be equal to the instance number of the Notification Class object*. Event-initiating objects shall use this number to refer to this Notification Class object indirectly. ~~Implementors may choose to make the Notification_Class property’s value be equal to the instance number of the Notification Class object. This decision is a local matter.~~

135*b*-5. Modify the Event Notification services to make the ’To State’ parameter mandatory for notifications of type ACK_NOTIFICATION.

The ConfirmedEventNotification and UnconfirmedEventNotification services do not presently include the ’To State’ parameter in notifications of type ACK_NOTIFICATION. Thus, a device receiving such a notification is unable to ascertain which transition of a previously signaled event has been acknowledged.

The proposed solution is to make the ’To State’ parameter mandatory in all event notification messages, including those of type ACK_NOTIFICATION.

[Change **Table 13-7**, p. 235. Make the ’To State’ parameter mandatory instead of conditional.]

Parameter Name	Req	Ind	Rsp	Cnf
…	…	…	…	…
To State	~~C~~*M*	~~C(=)~~*M (=)*		
…	…	…	…	…

[Change **Table 13-12**, p. 248. Make the 'To State' parameter mandatory instead of conditional.]

Parameter Name	Req	Ind	Rsp	Cnf
…	…	…	…	…
To State	~~C~~*M*	~~C(=)~~*M (=)*		
…	…	…	…	…

[Change **13.7.1.13**, p. 237 and **13.12.1.13,** p. 249. Only the first clause is shown below since they are identical.]

13.7.1.13 To State

This parameter, of type BACnetEventState, shall indicate the Event_State of the object after the occurrence of the event that initiated this notification. ~~This parameter shall only be present if the 'Notify Type' parameter is EVENT or ALARM.~~

[Change **21**, p. 353]

```
ConfirmedEventNotification-Request  ::= SEQUENCE {
        processIdentifier              [0]   Unsigned,
        initiatingDeviceIdentifier     [1]   BACnetObjectIdentifier,
        eventObjectIdentifier          [2]   BACnetObjectIdentifier,
        timeStamp                      [3]   BACnetTimeStamp,
        notificationClass              [4]   Unsigned,
        priority                       [5]   Unsigned8,
        eventType                      [6]   BACnetEventType,
        messageText                    [7]   CharacterString OPTIONAL,
        notifyType                     [8]   BACnetNotifyType,
        ackRequired                    [9]   BOOLEAN OPTIONAL,
        fromState                      [10]  BACnetEventState OPTIONAL,
        toState                        [11]  BACnetEventState~~ OPTIONAL~~,
        eventValues                    [12]  BACnetNotificationParameters OPTIONAL
}
```

[Change **21**, p. 359-360]

```
UnconfirmedEventNotification-Request  ::= SEQUENCE {
          processIdentifier              [0]   Unsigned,
          initiatingDeviceIdentifier     [1]   BACnetObjectIdentifier,
          eventObjectIdentifier          [2]   BACnetObjectIdentifier,
          timeStamp                      [3]   BACnetTimeStamp,
          notificationClass              [4]   Unsigned,
          priority                       [5]   Unsigned8,
          eventType                      [6]   BACnetEventType,
          messageText                    [7]   CharacterString OPTIONAL,
          notifyType                     [8]   BACnetNotifyType,
          ackRequired                    [9]   BOOLEAN OPTIONAL,
          fromState                      [10]  BACnetEventState OPTIONAL,
          toState                        [11]  BACnetEventState~~ OPTIONAL~~,
          eventValues                    [12]  BACnetNotificationParameters OPTIONAL
}
```

135*b*-6. A new BACnetDeviceObjectPropertyReference production is added and its use in the Event Enrollment and Schedule object types is specified.

BACnet does not currently allow devices that perform scheduling or event functions to schedule objects or have events that monitor objects outside of the device. This proposal resolves the problem by defining a datatype for references to object properties in remote devices and by allowing the use of this production in the Event and Schedule objects. This datatype is also used in proposal 135*b*-15, the Trend Log object.

[Change **Table 12-12**, p. 181]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Object_Property_Reference	*BACnetDeviceObjectPropertyReference*	R
…	…	…

[Change **12.10.8**, p. 184]

12.10.8 Object_Property_Reference

This property, of type *BACnetDeviceObjectPropertyReference*, designates the particular object and property referenced by this Event Enrollment object. The algorithm specified by the Event_Type property is applied to the referenced property in order to determine the Event_State of the event.

If this property is writable, it may be restricted to only support references to objects inside of the device containing the Event Enrollment object. If the property is restricted to referencing objects within the containing device, an attempt to write a reference to an object outside the containing device into this property shall cause a Result(-) to be returned with an error class of PROPERTY and an error code of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

If this property is set to reference an object outside the device containing the Event Enrollment object, the method used for acquisition of the referenced property value for the purpose of monitoring is a local matter. The only restriction on the method of data acquisition is that the monitoring device be capable of using ReadProperty for this purpose so as to be interoperable with all BACnet devices.

[Change **Table 12-22**, p. 213]

Property Identifier	Property Datatype	Conformance Code
…	…	…
List_Of_Object_Property_Reference	List of *BACnetDeviceObjectPropertyReference*	R
…	…	…

[Change **12.18.9**, p. 215]

12.18.9 List_Of_Object_Property_Reference

This property specifies the *Device Identifiers*, Object Identifiers and Property Identifiers of the properties to be written with specific values at specific times on specific days.

If this property is writable, it may be restricted to only support references to objects inside of the device containing the Schedule object. If the property is restricted to referencing objects within the containing device, an attempt to write a reference to an object outside the containing device into this property shall cause a Result(-) to be returned with an error class of PROPERTY and an error code of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

If this property is set to reference an object outside the device containing the Schedule object, the method used for writing to the referenced property value for the purpose of controlling the property is a local matter. The only restriction on the method of writing to the referenced property is that the scheduling device be capable of using WriteProperty for this purpose so as to be interoperable with all BACnet devices.

[Add to **18.3**, p. 314, by inserting in alphabetical order and renumbering other subclauses]

OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED – *An attempt has been made to write a value to a property that would require the device to exhibit non-supported optional functionality.*

[Add to **18.6**, p. 315, by inserting in alphabetical order and renumbering other subclauses]

OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED – *The parameters of a service are such that the device would be required to exhibit non-supported optional functionality.*

[Add to **21**, p. 363]

```
operational-problem                  (25),
optional-functionality-not-supported, (45)
password-failure                     (26),
```

[Change **21** comment to the **Error** production, p. 364]

```
-- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
-- 256-65535 may be used by others subject to the procedures and constraints described
-- in Clause 23. The last enumeration used in this version is 45.
```

[Addition to **21**, p. 366 (between the productions for BACnetDestination and BACnetDeviceStatus)]

```
BACnetDeviceObjectPropertyReference  ::= SEQUENCE {
        objectIdentifier        [0]  BACnetObjectIdentifier,
        propertyIdentifier      [1]  BACnetPropertyIdentifier,
        propertyArrayIndex      [2]  Unsigned OPTIONAL      -- used only with array datatype
                                                            -- if omitted with an array then
                                                            -- the entire array is referenced
        deviceIdentifier        [3]  BACnetObjectIdentifier OPTIONAL
        }
```

[Change **21**, p. 371 (in the production for BACnetEventParameter)]

```
...
command-failure [3]     SEQUENCE {
                        time-delay                    [0]   Unsigned,
                        feedback-property-reference   [1]   BACnetDeviceObjectPropertyReference
                },

floating-limit    [4]   SEQUENCE {
                              time-delay              [0]   Unsigned,
                              setpoint-reference      [1]   BACnetDeviceObjectPropertyReference,
                              low-diff-limit          [2]   REAL,
                              high-diff-limit         [3]   REAL,
                              deadband                [4]   REAL
...
```

[Change **Annex C**, p. 415]

```
EVENT-ENROLLMENT ::= SEQUENCE {
        ...
        object-property-reference  [78]      BACnetDeviceObjectPropertyReference,
        ...
        }
```

[Change **Annex C**, p. 418]

```
SCHEDULE ::= SEQUENCE {
        ...
        list-of-object-property-references  [54] SEQUENCE OF BACnetDeviceObjectPropertyReference,
        ...
}
```

[Change **Annex D.10**, p. 426]

The following Analog Input object is assumed for the examples below. *All objects are assumed to be located in device 12.*

[Change **Annex D.10**, p. 426]

```
Property:  Event_Parameters =              (30, 65.0, 85.0, 0.25)
Property:  Object_Property_Reference = ((Device, Instance 12),(Analog Input, Instance 2), Present_Value)
Property:  Event_State =                   HIGH_LIMIT
```

[Change **Annex D.10**, p. 427]

```
Property:  Event_Parameters =              (5, 0.25)
Property:  Object_Property_Reference = ((Device, Instance 12),(Analog Input, Instance 2), Present_Value)
Property:  Event_State =                   NORMAL
```

[Change **Annex D.10**, p. 427]

```
Property:  Event_Parameters =              (30, B'0111',
                                            (B'0100', B'0010', B'0001', B'0110', B'0101', B'0011'))
Property:  Object_Property_Reference = ((Device, Instance 12),(Analog Input, Instance 2), Status_Flags)
Property:  Event_State =                   NORMAL
```

[Change **Annex D.18**, p. 433]

The following is an example of a Schedule object that is used to control a classroom during the school year. In this example, a different Schedule object is assumed to be defined for the remainder of the calendar year. The reference property of this schedule is the Present_Value of a Binary Output object, *in device 12*, that controls a rooftop unit providing conditioned air to room 208.

[Change **Annex D.18**, p. 433]

```
Property:  Exception_Schedule =    {((23-NOV-1995),(0:00,INACTIVE),10),
                                    ((HOLIDAYS,(0:00,INACTIVE),11),
                                    ((5-MAR-1996)-(7-MAR-1996),
                                    ((9:00,ACTIVE),(14:00,INACTIVE)),6)}
Property:  List_Of_Object_Property_References = ((Device, Instance 12),(Binary Output, Instance 9),
                                                  Present_Value)
Property:  Priority_For_Writing =  15
```

135*b*-7. Add a Multi-state Value object type.

The proposal has been made for the addition of a standardized Multi-state Value object type. The new object type proposed here represents a synthesis of Multi-state Input and Multi-state Output object types that is optionally commandable and optionally writable. In this regard, the Multi-state Value object type is akin to the Analog Value and Binary Value object types which are syntheses of the Analog Input and Output and Binary Input and Output object types respectively.

In the proposed clause below, the value of 'X' in the clause numbering will be replaced at the time of publication with a value placing it in the proper sequence in Clause 12, dependent on the potential addition of other object types. This addendum, for example, proposes the addition of an Averaging object type (see 135*b*-8), the acceptance of which would influence the numbering of the present clause.

12.X Multi-state Value Object Type

The Multi-state Value object type defines a standardized object whose properties represent the externally visible characteristics of a multi-state value. A "multi-state value" is a control system parameter residing in the memory of the BACnet Device. The actual functions associated with a specific state are a local matter and not specified by the protocol. For example, a particular state may represent the active/inactive condition of several physical inputs and outputs or perhaps the value of an analog input or output. The Present_Value property is an unsigned integer number representing the state. The State_Text property associates a description with each state.

The Multi-state Value object type and its properties are summarized in Table 12-X and described in detail in this subclause.

NOTE: Do not confuse the Present_Value state with the Event_State property, which reflects the offnormal state of the Multi-state Value.

Table 12-X. Properties of the Multi-state Value Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	Unsigned	R[1]
Description	CharacterString	O
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Reliability	BACnetReliability	O[2]
Out_Of_Service	BOOLEAN	R
Number_of_States	Unsigned	R
State_Text	BACnetARRAY[N] of CharacterString	O
Priority_Array	BACnetPriority	O[3]
Relinquish_Default	Unsigned	O[3]
Time_Delay	Unsigned	O[4]
Notification_Class	Unsigned	O[4]
Alarm_Values	List of Unsigned	O[4]
Fault_Values	List of Unsigned	O[4]
Event_Enable	BACnetEventTransitionBits	O[4]
Acked_Transitions	BACnetEventTransitionBits	O[4]
Notify_Type	BACnetNotifyType	O[4]
Event_Time_Stamps	BACnetARRAY[3] of BACnetTimeStamp	O[4]

[1] If Present_Value is commandable, then it is required to also be writable. This property is required to be writable when Out_Of_Service is TRUE.
[2] This property shall be required if Fault_Values is present.
[3] If Present_Value is commandable, then both of these properties shall be present.
[4] These properties are required if the object supports intrinsic reporting

12.X.1 Object_Identifier

This property, of type BACnetObjectIdentifier, is a numeric code that is used to identify the object. It shall be unique within the BACnet Device that maintains it.

12.X.2 Object_Name

This property, of type CharacterString, shall represent a name for the object that is unique within the BACnet Device that maintains it. The minimum length of the string shall be one character. The set of characters used in the Object_Name shall be restricted to printable characters.

12.X.3 Object_Type

This property, of type BACnetObjectType, indicates membership in a particular object type class. The value of this property shall be MULTISTATE_VALUE.

12.X.4 Present_Value

This property, of type Unsigned, reflects the logical state of the multi-state value. The logical state of the multi-state value shall be one of 'n' states, where 'n' is the number of states defined in the Number_of_States property. How the Present_Value is used is a local matter. The Present_Value property shall always have a value greater than zero. Present_Value shall be optionally commandable. If Present_Value is commandable for a given object instance, then the Priority_Array and Relinquish_Default properties shall also be present for that instance. The Present_Value property shall be writable when Out_Of_Service is TRUE (see 12.X.9).

12.X.5 Description

This property, of type CharacterString, is a string of printable characters whose content is not restricted.

12.X.6 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of the multi-state value. Three of the flags are associated with the values of other properties of this object. A more detailed status could be determined by reading the properties that are linked to these flags. The relationship between individual flags is not defined by the protocol. The four flags are

{IN_ALARM, FAULT, OVERRIDDEN, OUT_OF_SERVICE}

where:

IN_ALARM	Logical FALSE (0) if the Event_State property (12.X.7) has a value of NORMAL, otherwise logical TRUE (1).
FAULT	Logical TRUE (1) if the Reliability property (12.X.8) is present and does not have a value of NO_FAULT_DETECTED, otherwise logical FALSE (0).

OVERRIDDEN	Logical TRUE (1) if the point has been overridden by some mechanism local to the BACnet Device. In this context "overridden" is taken to mean that the Present_Value property is not changeable through BACnet services.
OUT_OF_SERVICE	Logical TRUE (1) if the Out_of_Service property (12.X.9) has a value of TRUE, otherwise logical FALSE (0).

12.X.7 Event_State

The Event_State property, of type BACnetEventState, is included in order to provide a way to determine if this object has an active event state associated with it. If the object supports intrinsic reporting, then the Event_State property shall indicate the event state of the object. If the object does not support intrinsic reporting then:

(a) if the Reliability property is not present, then the value of Event_State shall be NORMAL, or

(b) if the Reliability property is present and Reliability is NO_FAULT_DETECTED then Event_State shall be NORMAL, or

(c) if the Reliability property is present and Reliability is not NO_FAULT_DETECTED then Event_State shall be FAULT.

12.X.8 Reliability

The reliability property, of type BACnetReliability, provides an indication of whether the Present_Value is "reliable" as far as the BACnet Device or operator can determine and, if not, why. The Reliability property is required to be present if the Fault_Values property is present. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, MULTI_STATE_FAULT, UNRELIABLE_OTHER}.

12.X.8.1 Conditions for Generating a TO-FAULT Event

A TO-FAULT event is generated under these conditions:

(a) the Reliability property becomes not equal to NO_FAULT_DETECTED, and
(b) the TO-FAULT flag must be enabled in the Event_Enable property.

12.X.9 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the Present_Value property of the Multi-state Value object is prevented from being modified by software local to the BACnet device in which the object resides. When Out_Of_Service is TRUE the Present_Value property may still be written to freely. In addition, the Reliability property and the corresponding state of the FAULT flag of the Status_Flags property shall be decoupled when Out_Of_Service is TRUE. While the Out_Of_Service property is TRUE, the Present_Value and Reliability properties may be changed to any value as a means of simulating specific fixed conditions or for testing purposes. Other functions that depend on the state of the Present_Value or Reliability properties shall respond to changes made to these properties while Out_Of_Service is TRUE. If the Priority_Array and Relinquish_Default properties are present, then writing to the Present_Value property shall be controlled by the BACnet command prioritization mechanism. See Clause 19.

12.X.10 Number_of_States

This property, of type Unsigned, defines the number of states the Present_Value may have. The Number_Of_States property shall always have a value greater than zero.

12.X.11 State_Text

This property is a BACnetARRAY of character strings representing descriptions of all possible states of the Present_Value. The number of descriptions matches the number of states defined in the Number_of_States property. The Present_Value, interpreted as an integer, serves as an index into the array.

12.X.12 Priority_Array

This property is a read-only array that contains prioritized commands that are in effect for this object. See Clause 19 for a description of the prioritization mechanism. If either the Priority_Array property or the Relinquish_Default property are present, then both of them shall be present. If Present_Value is commandable, then Priority_Array and Relinquish_Default shall both be present.

12.X.13 Relinquish_Default

This property is the default value to be used for the Present_Value property when all command priority values in the Priority_Array property have a NULL value. See Clause 19. If either the Relinquish_Default property or the Priority_Array property are present, then both of them shall be present. If Present_Value is commandable, then Priority_Array and Relinquish_Default shall both be present.

12.X.14 Time_Delay

This property, of type Unsigned, shall specify the minimum period of time in seconds that the Present_Value must remain:

(a) equal to any one of the values in the Alarm_Values property before a TO-OFFNORMAL event is generated, or
(b) not equal to any of the values in the Alarm_Values property before a TO-NORMAL event is generated.

This property is required if intrinsic reporting is supported by this object.

12.X.15 Notification_Class

This property, of type Unsigned, shall specify the notification class to be used when handling and generating event notifications for this object. The Notification_Class property implicitly refers to a Notification Class object that has a Notification_Class property with the same value. This property is required if intrinsic reporting is supported by this object.

12.X.16 Alarm_Values

This property, of type List of Unsigned, shall specify any states the Present_Value must equal before a TO-OFFNORMAL event is generated. This property is required if intrinsic reporting is supported by this object.

12.X.16.1 Conditions for Generating a TO-OFFNORMAL Event

A TO-OFFNORMAL event is generated under these conditions:

(a) the Present_Value must equal at least one of the values in the Alarm_Values list, and
(b) the Present_Value must remain equal to the same value for a minimum period of time, specified by the Time_Delay property, and
(c) the TO-OFFNORMAL flag must be enabled in the Event_Enable property.

12.X.16.2 Conditions for Generating a TO-NORMAL Event

Once equal, the Present_Value must become not equal to any of the states in this property and not equal to any of the states in the Fault_Values property before a TO-NORMAL event is generated under these conditions:

(a) the Present_Value must remain not equal to any of the states in the Alarm_Values property for a minimum period of time, specified by the Time_Delay property, and
(b) the Present_Value must remain not equal to any of the states in the Fault_Values property, and
(c) the TO-NORMAL flag must be enabled in the Event_Enable property.

12.X.17 Fault_Values

This property, of type List of Unsigned, shall specify any states the Present_Value must equal before a TO-FAULT event is generated. If Present_Value becomes equal to any of the states in the Fault_Values list, and no physical fault has been detected for any inputs or outputs that the Present_Value represents, then the Reliability property shall have the value MULTI_STATE_FAULT. The Fault_Values property is required if intrinsic reporting is supported by this object.

12.X.18 Event_Enable

This property, of type BACnetEventTransitionBits, shall convey three flags that separately enable and disable reporting of TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events. This property is required if intrinsic reporting is supported by this object.

12.X.19 Acked_Transitions

This property, of type BACnetEventTransitionBits, shall convey three flags that separately indicate the receipt of acknowledgements for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events. These flags shall be cleared upon the occurrence of the corresponding event and set under any of these conditions:

(a) upon receipt of the corresponding acknowledgement;
(b) upon the occurrence of the event if the corresponding flag is not set in the Event_Enable property (meaning event notifications will not be generated for this condition and thus no acknowledgement is expected);
(c) upon the occurrence of the event if the corresponding flag is set in the Event_Enable property and the corresponding flag in the Ack_Required property of the Notification Class object implicitly referenced by the Notification_Class property of this object is not set (meaning no acknowledgement is expected).

This property is required if intrinsic reporting is supported by this object.

12.X.20 Notify_Type

This property, of type BACnetNotifyType, shall convey whether the notifications generated by the object should be Events or Alarms. This property is required if intrinsic reporting is supported by this object.

12.X.21 Event_Time_Stamps

This optional property, of type BACnetARRAY [3] of BACnetTimeStamp, shall convey the times of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events, respectively. Time stamps of type Time or Date shall have 'FF' in each octet and Sequence number time stamps shall have the value 0 if no event notification of that type has been generated since the object was created. This property is required if intrinsic reporting is supported by this object.

[Change **Table 13-1**, p. 218]

Add Multi-state Value to object types Binary Input, Binary Output, Binary Value, Multi-state Input, Multi-state Output.

[Change **13.2**, p. 219, final paragraph]

In the case of Multi-state Inputs *and Multi-state Values*, the Alarm_Values property…

[Change **Table 13-2**, p. 220 and **Table 13-3**, p. 221**]**

Add Multi-state Value to object types Binary Input, Binary Value, Multi-state Input.

[Change the following productions in **21**]

```
BACnetReliability ::= ENUMERATED {
. . .
          multi-state-fault          (9),

BACnetObjectType ::= ENUMERATED {
. . .
          multi-state-value          (19),

BACnetObjectTypesSupported ::= BIT STRING {
. . .
          multi-state-value          (19),
```

[Change **Annex A**, p.408]

Annex A, Standard Object Types Supported:

Multi-state Value ☐ ☐ ☐ _____ _____

[Add the following to **Annex C**, p.417]

```
MULTI-STATE-VALUE ::= SEQUENCE {
          object-identifier      [75]   BACnetObjectIdentifier,
          object-name            [77]   CharacterString,
          object-type            [79]   BACnetObjectType,
          present-value          [85]   Unsigned,
                                        -- maximum value is restricted by the number-of-states
          description            [28]   CharacterString OPTIONAL,
          status-flags          [111]   BACnetStatusFlags,
          event-state            [36]   BACnetEventState,
          reliability           [103]   BACnetReliability OPTIONAL,
          out-of-service         [81]   BOOLEAN,
          number-of-states       [74]   Unsigned,
          state-text            [110]   SEQUENCE OF CharacterString OPTIONAL,
                                        -- accessed as a BACnetARRAY
          priority-array         [87]   BACnetPriorityArray OPTIONAL,
          relinquish-default    [104]   Unsigned OPTIONAL,
          time-delay            [113]   Unsigned OPTIONAL,
          notification-class     [17]   Unsigned OPTIONAL,
          alarm-values            [7]   SEQUENCE OF Unsigned OPTIONAL,
          fault-values           [39]   SEQUENCE OF Unsigned OPTIONAL,
event-enable          [35]   BACnetEventTransitionBits OPTIONAL,
acked-transitions      [0]   BACnetEventTransitionBits OPTIONAL,
notify-type           [72]   BACnetNotifyType OPTIONAL,
event-time-stamps    [130]   SEQUENCE OF BACnetTimeStamp OPTIONAL
                             --accessed as a BACnetARRAY
}
```

[Add the following to **Annex D**, p.431, after the current **D.15** and before the current **D.16**]

D.X Example of a Multi-state Value Object

Example 1 - Unit Ventilator Operating Mode.
In this example the Present_Value is being used to indicate the operating mode for a Unit Ventilator. The actual logic used to determine and establish the present value is a local matter.

Property:	Object_Identifier =	(Multi-state Value, Instance 1)
Property:	Object_Name =	"UV39"
Property:	Object_Type =	MULTISTATE_VALUE
Property:	Present_Value =	2
Property:	Description =	"UnitVent Room 39"
Property:	Status_Flags =	{FALSE,FALSE,FALSE,FALSE}
Property:	Event_State =	NORMAL
Property:	Reliability =	NO_FAULT_DETECTED
Property:	Out_of_Service =	FALSE
Property:	Number_of_States =	4
Property:	State_Text =	("Unoccupied","Warmup","Occupied","Setback")

135*b*-8. Add an Averaging object type.

The proposed new object type provides a network-visible way to monitor the average, minimum, and maximum values attained by a sampled property of a specified object. The property may be of datatype BOOLEAN, INTEGER, Unsigned, Enumerated or REAL.

In the proposed clause below, the value of 'X' in the clause numbering will be replaced at the time of publication with a value placing it in the proper sequence in Clause 12, dependent on the potential addition of other object types.

12.X Averaging Object Type

The Averaging object type defines a standardized object whose properties represent the externally visible characteristics of a value that is sampled periodically over a specified time interval. The Averaging object records the minimum, maximum and average value over the interval, and makes these values visible as properties of the Averaging object. The sampled value may be the value of any BOOLEAN, INTEGER, Unsigned, Enumerated or REAL property value of any object within the BACnet Device in which the object resides. Optionally, the object property to be sampled may exist in a different BACnet Device. The Averaging object shall use a "sliding window" technique that maintains a buffer of *N* samples distributed over the specified interval. Every (time interval/N) seconds a new sample is recorded displacing the oldest sample from the buffer. At this time, the minimum, maximum and average are recalculated. The buffer shall maintain an indication for each sample that permits the average calculation and minimum/maximum algorithm to determine the number of valid samples in the buffer.

The Averaging object type and its properties are summarized in Table 12-X and described in detail in this subclause.

Table 12-X. Properties of the Averaging Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Minimum_Value	REAL	R
Minimum_Value_Timestamp	BACnetDateTime	O
Average_Value	REAL	R
Variance_Value	REAL	O
Maximum_Value	REAL	R
Maximum_Value_Timestamp	BACnetDateTime	O
Description	CharacterString	O
Attempted_Samples	Unsigned	W[1]
Valid_Samples	Unsigned	R
Object_Property_Reference	BACnetDeviceObjectPropertyReference	R[1]
Window_Interval	Unsigned	W[1]
Window_Samples	Unsigned	W[1]

[1] If any of these properties are written to using BACnet services, then all of the buffer samples shall become invalid, 'Attempted_Samples' shall become zero, 'Valid_Samples' shall become zero, 'Minimum_Value' shall become INF, 'Average_Value' shall become NaN and 'Maximum_Value' shall become -INF.

12.X.1 Object_Identifier

This property, of type BACnetObjectIdentifier, is a numeric code that is used to identify the object. It shall be unique within the BACnet Device that maintains it.

12.X.2 Object_Name

This property, of type CharacterString, shall represent a name for the object that is unique within the BACnet Device that maintains it. The minimum length of the string shall be one character. The set of characters used in the Object_Name shall be restricted to printable characters.

12.X.3 Object_Type

This property, of type BACnetObjectType, indicates membership in a particular object type class. The value of this property shall be AVERAGING.

12.X.4 Minimum_Value

This property, of type REAL, shall reflect the lowest value contained within the buffer window for the most recent 'Window_Samples' samples, or the actual number of samples ('Valid_Samples') if less than 'Window_Samples' samples have been taken. After a device restart, or after 'Attempted_Samples', 'Object_Property_Reference', 'Window_Samples' or 'Window_Interval' are written to using BACnet services, until a sample is taken, 'Minimum_Value' shall have the value **INF**.

12.X.5 Minimum_Value_Timestamp

This optional property, of type BACnetDateTime, indicates the date and time at which the value stored in Minimum_Value was sampled.

12.X.6 Average_Value

This property, of type REAL, shall reflect the average value contained within the buffer window for the most recent 'Window_Samples' samples, or the actual number of samples ('Valid_Samples') if less than 'Window_Samples' samples have been taken. The average shall be calculated by taking the arithmetic sum of all non-missed buffer samples and dividing by the number of non-missed buffer samples. After a device restart, or after 'Attempted_Samples', 'Object_Property_Reference', 'Window_Samples' or 'Window_Interval' are written to using BACnet services, until a sample is taken, 'Average_Value' shall have the value **NaN**.

12.X.7 Variance_Value

This optional property, of type REAL, shall reflect the variance value contained within the buffer window for the most recent 'Window_Samples' samples, or the actual number of samples ('Valid_Samples') if less than 'Window_Samples' samples have been taken. After a device restart, or after 'Attempted_Samples', 'Object_Property_Reference', 'Window_Samples' or 'Window_Interval' are written to using BACnet services, until a sample is taken, 'Variance_Value' shall have the value **NaN**.

12.X.8 Maximum_Value

This property, of type REAL, shall reflect the highest value contained within the buffer window for the most recent 'Window_Samples' samples, or the actual number of samples ('Valid_Samples') if less than 'Window_Samples' samples have been taken. After a device restart, or after 'Attempted_Samples', 'Object_Property_Reference', 'Window_Samples' or 'Window_Interval' are written to using BACnet services, until a sample is taken, 'Maximum_Value' shall have the value **-INF**.

12.X.9 Maximum_Value_Timestamp

This optional property, of type BACnetDateTime, indicates the date and time at which the value stored in Maximum_Value was sampled.

12.X.10 Description

This property, of type CharacterString, is a string of printable characters whose content is not restricted.

12.X.11 Attempted_Samples

This property, of type Unsigned, indicates the number of samples that have been attempted to be collected for the current window. The only acceptable value that may be written to this property shall be zero. If 'Attempted_Samples' is less than 'Window_Samples' then a period of time less than 'Window_Interval' has elapsed since a device restart, or 'Attempted_Samples', 'Object_Property_Reference', 'Window_Samples' or 'Window_Interval' have been written to using BACnet services. The number of missed samples in the current window can be calculated by subtracting 'Valid_Samples' from 'Attempted_Samples.' After a device restart, or after 'Attempted_Samples', 'Object_Property_Reference', 'Window_Samples' or 'Window_Interval' are written to using BACnet services, until a sample is taken, 'Attempted_Samples' shall have the value **zero**.

12.X.12 Valid_Samples

This read-only property, of type Unsigned, indicates the number of samples that have been successfully collected for the current window. This value can be used to determine whether any of the samples in the current 'Window_Interval' are missing. The number of missed samples in the current window can be calculated by subtracting 'Valid_Samples' from 'Attempted_Samples.' A result greater than zero indicates the number of samples that encountered an error when the sample was being recorded. After a device restart, or after 'Attempted_Samples', 'Object_Property_Reference', 'Window_Samples' or 'Window_Interval' are written to using BACnet services until a sample is taken, 'Valid_Samples' shall have the value **zero**.

12.X.13 Object_Property_Reference

This property, of type BACnetDeviceObjectPropertyReference, shall identify the object and property whose value is to be sampled during the 'Window_Interval'. The object referenced may be located within the device containing the Averaging object, or optionally the Averaging object may support the referencing of object properties in other devices. External references may be restricted to a particular set of BACnet devices. The referenced object property must have any of the numeric datatypes BOOLEAN, INTEGER, Unsigned, Enumerated or REAL. All sampled data shall be converted to REAL for calculation purposes. BOOLEAN FALSE shall be considered to be zero and TRUE shall be considered to be one. Enumerated datatypes shall be treated as Unsigned values. If an implementation supports writing to 'Object_Property_Reference', then If 'Object_Property_Reference' is written to using BACnet services, then all of the buffer samples shall become invalid, 'Attempted_Samples' shall become zero, 'Valid_Samples' shall become zero,, 'Minimum_Value' shall become INF, 'Average_Value' shall become NaN and 'Maximum_Value' shall become -INF.

12.X.14 Window_Interval

This property, of type Unsigned, shall indicate the period of time in seconds over which the minimum, maximum and average values are calculated. The minimum acceptable value for 'Window_Interval' shall be a local matter. Every 'Window_Interval' divided by 'Window_Samples' seconds a new sample shall be taken by reading the value of the property referenced by the 'Object_Property_Reference'. Whether the sample represents an instantaneous "snapshot" or a continuously calculated sample shall be a local matter. If 'Window_Interval' is written to using BACnet services, then all of the buffer samples shall become invalid, ’Attempted_Samples’ shall become zero, ’Valid_Samples’ shall become zero, ’Minimum_Value’ shall become INF, ’Average_Value’ shall become NaN and ’Maximum_Value’ shall become -INF.

12.X.15 Window_Samples

This property, of type Unsigned, shall indicate the number of samples to be taken during the period of time specified by the ’Window_Interval’ property. ’Window_Samples’ must be greater than zero and all implementations shall support at least 15 samples. Every ’Window_Interval’ divided by ’Window_Samples’ seconds a new sample shall be taken by reading the value of the property referenced by the ’Object_Property_Reference’. Whether the sample represents an instantaneous "snapshot" or a continuously calculated sample shall be a local matter. If ’Window_Interval’ is written to using BACnet services, then all of the buffer samples shall become invalid, ’Attempted_Samples’ shall become zero, ’Valid_Samples’ shall become zero, ’Minimum_Value’ shall become INF, ’Average_Value’ shall become NaN and ’Maximum_Value’ shall become -INF.

[Changes to ASN.1 productions in **21**]

BACnetObjectType ::= ENUMERATED {

. . .

averaging	*(18),*

BACnetObjectTypesSupported ::= BIT STRING {

. . .

averaging	*(18),*

BACnetPropertyIdentifier ::= ENUMERATED {

. . .

archive	(13),
attempted-samples	*(124),*
average-value	*(125),*
bias	(14),

. . .

maximum-output	(61),
maximum-value	*(135),*
maximum-value-timestamp	*(149),*
max-apdu-length-accepted	(62),

. . .

minimum-output	(68),
minimum-value	*(136),*
minimum-value-timestamp	*(150),*
min-pres-value	(69),

. . .

utc-offset	(119),
valid-samples	*(146),*
variance-value	*(151),*
vendor-identifier	(120),

. . .

weekly-schedule	(123),
window-interval	*(147),*
window-samples	*(148),*
-- see attempted-samples	*(124),*
-- see average-value	*(125),*
-- see maximum-value	*(135),*
-- see minimum-value	*(136),*
-- see valid-samples	*(146),*
-- see window-interval	*(147),*

```
    -- see window-samples              (148),
    -- see maximum-value-timestamp     (149),
    -- see minimum-value-timestamp     (150),
    -- see variance-value              (151),
```

-- The highest enumeration used in this version is ~~123~~ *151*.

[Add to **Annex A**, Standard Object Types Supported**]**

Averaging ☐ ☐ ☐ _____ _____

[Add to **Annex C**]

```
AVERAGING ::= SEQUENCE {
        object-identifier          [75]    BACnetObjectIdentifier,
        object-name                [77]    CharacterString,
        object-type                [79]    BACnetObjectType,
        minimum-value             [136]    REAL,
        minimum-value-timestamp   [150]    BACnetDateTime OPTIONAL,
        average-value             [125]    REAL,
        variance-value            [151]    REAL OPTIONAL,
        maximum-value             [135]    REAL,
        maximum-value-timestamp   [149]    BACnetDateTime OPTIONAL,
        description                [28]    CharacterString OPTIONAL,
        attempted-samples         [124]    Unsigned,
        valid-samples             [146]    Unsigned,
        object-property-reference  [78]    BACnetDeviceObjectPropertyReference,
        window-interval           [147]    Unsigned,
        window-samples            [148]    Unsigned
        }
```

[Add to **Annex D**]

D.X Example of an Averaging Object

The following is an example of an Averaging object that is used for determining average and maximum electrical demand. The object property it refers to is a standard analog input measuring KW. In the current period, one sample was missed.

Property:	Object_Identifier =	(Averaging, Instance 1)
Property:	Object_Name =	"FLR 12 DEMAND"
Property:	Object_Type =	AVERAGING
Property:	Minimum_Value =	2.4
Property:	Minimum_Value_Timestamp =	(16-DEC-1999,13:15:07.32)
Property:	Average_Value =	12.7
Property:	Maximum_Value =	18.8
Property:	Maximum_Value_Timestamp =	(16-DEC-1999,13:06:12.19)
Property:	Description =	"Floor 12 Electrical Demand"
Property:	Attempted_Samples =	15
Property:	Valid_Samples =	14
Property:	Object_Property_Reference =	(Analog Input, Instance 12)
Property:	Window_Interval =	900
Property:	Window_Samples =	15

[Add to **3.3**, pp. 5, 6]

3.3 Abbreviations and Acronyms Used in this Standard

. . .

INF *"Infinity", a unique binary pattern representing positive infinity (see ANSI/IEEE 754-1985)*
-INF *"Negative infinity", a unique binary pattern representing negative infinity (see ANSI/IEEE 754-1985)*

. . .

NaN *"Not a Number", a unique binary pattern representing an invalid number (see ANSI/IEEE 754-1985)*

135*b*-9. Change all ’Process Identifier’ properties and parameters to Unsigned32.

There are various places within BACnet that "process identifiers" are used, notably in the COV services, event notification services and EventEnrollment object. In every case these are currently defined as "Unsigned" meaning that (theoretically) they are unbounded in possible size. As a practical matter, it would be a beneficial to implementors if the size of these identifiers were bounded. The current proposal is to change all instances of these identifiers to Unsigned32.

[Change **Table 12-2**, p. 181]

Table 12-12. Properties of the Event Enrollment Object Type

Property Identifier	Property Datatype	Conformance Code
Process_Identifier	~~Unsigned~~ *Unsigned32*	O^2

[Change **12.10.14**, p. 185]

12.10.14 Process_Identifier

This property, of type ~~Unsigned~~ *Unsigned32*, shall …

[Change **13.5.1.2**, p. 231]

13.5.1.2 Acknowledging Process Identifier

This parameter, of type ~~Unsigned~~ *Unsigned32*, shall …

[Change **13.6.1.2**, p. 233]

13.6.1.2 Subscriber Process Identifier

This parameter, of type ~~Unsigned~~ *Unsigned32*, shall …

[Change **13.7.1.2**, p. 235]

13.7.1.2 Process Identifier

This parameter, of type ~~Unsigned~~ *Unsigned32*, shall …

[Change **13.10.12**, p. 243]

13.10.1.2 Subscriber Process Identifier

This parameter, of type ~~Unsigned~~ *Unsigned32*, shall …

[Change **13.11.1.2**, p. 246]

13.11.1.2 Subscriber Process Identifier

This parameter, of type ~~Unsigned~~ *Unsigned32*, shall …

[Change **13.12.1.2**, p. 248]

13.12.1.2 Process Identifier

This parameter, of type ~~Unsigned~~ *Unsigned32*, shall …

[Change **21**, p. 352]

```
AcknowledgeAlarm-Request ::= SEQUENCE {
        acknowledgingProcessIdentifier  [0]  ~~Unsigned~~ Unsigned32,

ConfirmedCOVNotification-Request ::= SEQUENCE {
        subscriberProcessIdentifier     [0]  ~~Unsigned~~ Unsigned32,
```

[Change **21**, p. 353]

```
ConfirmedEventNotification-Request ::= SEQUENCE {
        processIdentifier               [0]  ~~Unsigned~~ Unsigned32,
```

[Change **21**, p. 354]

```
SubscribeCOV-Request ::= SEQUENCE {
        subscriberProcessIdentifier     [0]  ~~Unsigned~~ Unsigned32,
```

[Change **21**, p. 359]

```
UnconfirmedCOVNotification-Request ::= SEQUENCE {
        subscriberProcessIdentifier     [0]  ~~Unsigned~~ Unsigned32,

UnconfirmedEventNotification-Request ::= SEQUENCE {
        processIdentifier               [0]  ~~Unsigned~~ Unsigned32,
```

[Change **21**, p. 366]

```
BACnetDestination ::= SEQUENCE {
        …
        processIdentifier                    ~~Unsigned~~ Unsigned32,
        …
```

[Change **21**, p. 379]

```
BACnetRecipientProcess ::= SEQUENCE {
        …
        processIdentifier               [1]  ~~Unsigned~~ Unsigned32,
```

135*b*-10. Change the Multi-state Input object type to correct flaws related to fault detection and reporting and achieve consistency with the new Multi-state Value object type.

The recent discussions surrounding the Multi-state Value object have exposed some issues related to "fault" detection and reporting. The resolution of those issues involves making some changes to the Multi-state Input object to correct the same flaws and provide consistency. The consensus was also to change the handling of faults so that no time delay is expected or applied using the network-visible Time_Delay property. The affected clauses are shown below.

Table 12-17. Properties of the Multi-state Input Object Type

Property Identifier	Property Datatype	Conformance Code
…		
Reliability	BACnetReliability	~~O~~*O*[2]
…		
Time_Delay	Unsigned	~~O[2]~~*O*[3]
Notification_Class	Unsigned	~~O[2]~~*O*[3]
Alarm_Values	List of Unsigned	~~O[2]~~*O*[3]
Fault_Values	List of Unsigned	~~O[2]~~*O*[3]
Event_Enable	BACnetEventTransitionBits	~~O[2]~~*O*[3]
Acked_Transitions	BACnetEventTransitionBits	~~O[2]~~*O*[3]
Notify_Type	BACnetNotifyType	~~O[2]~~*O*[3]
Event_Time_Stamps	*BACnetARRAY[3] of BACnetTimeStamp*	~~O[2]~~*O*[3]

[2] *This property shall be required if Fault_Values is present.*
~~2~~[3] These properties are required if the object supports intrinsic reporting

12.14.8 Event_State

The Event_State property, of type BACnetEventState, is included in order to provide a way to determine if this object has an active event state associated with it. If the object supports intrinsic reporting, then the Event_State property shall indicate the event state of the object. If the object does not support intrinsic reporting then ~~the value of this property shall be NORMAL.~~:

(a) if the Reliability property is not present, then the value of Event_State shall be NORMAL, or
(b) if the Reliability property is present <u>and</u> Reliability is NO_FAULT_DETECTED then Event_State shall be NORMAL, or
(c) If the Reliability property is present <u>and</u> Reliability is <u>not</u> NO_FAULT_DETECTED then Event_State shall be FAULT.

12.14.9 Reliability

The reliability property, of type BACnetReliability, provides an indication of whether the Present_Value or the operation of the physical inputs in question are "reliable" as far as the BACnet Device or operator can determine and, if not, why. *The Reliability property is required to be present if the Fault_Values property is present.* The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, NO_SENSOR, OVER_RANGE, UNDER_RANGE,OPEN_LOOP, SHORTED_LOOP, *MULTI_STATE_FAULT*, UNRELIABLE_OTHER}.

12.14.9.1 Conditions for Generating a TO-FAULT Event

A TO-FAULT event is generated under these conditions:

(a) the Reliability property becomes not equal to NO_FAULT_DETECTED, and
(b) the TO-FAULT flag must be enabled in the Event_Enable property.

12.14.15.2 Conditions for Generating a TO-NORMAL Event

Once equal, the Present_Value must become not equal to any of the states in this property *and not equal to any of the states in the Fault_Values property* before a TO-NORMAL event is generated under these conditions:

(a) the Present_Value must remain not equal to any of the values in the Alarm_Values list for a minimum period of time, specified in the Time_Delay property, and
(b) the Present_Value must remain not equal to any of the states in the Fault_Values property, and
~~(b)~~ *(c)* the TO-NORMAL flag must be enabled in the Event_Enable property.

12.14.16 Fault_Values

This property, of type List of Unsigned, shall specify any states the Present_Value must equal before a TO-FAULT event is generated. *If Present_Value becomes equal to any of the states in the Fault_Values list, and no physical fault has been detected for any inputs that the Present_Value represents, then the Reliability property shall have the value MULTI_STATE_FAULT. The Fault_Values* ~~This~~ property is required if intrinsic reporting is supported by this object.

~~**12.14.16.1 Conditions for Generating a TO-FAULT Event**~~

~~A TO-FAULT event is generated under these conditions:~~
~~(a) the Present_Value property must equal at least one of the values in the Fault_Values list, and~~
~~(b) the Present_Value must remain equal to the same value for a minimum period of time, specified in the Time_Delay property, and~~
~~(c) the TO-FAULT flag must be enabled in the Event_Enable property.~~

~~**12.14.16.2 Conditions for Generating a TO_NORMAL Event**~~

~~Once equal, the Present_Value must become not equal to any of the states in this property before a TO-NORMAL event is generated under these conditions:~~

~~(a) the Present_Value property must remain not equal to any of the values in the Fault_Values list for a minimum period of time, specified in the Time_Delay property, and~~
~~(b) the TO-NORMAL flag must be enabled in the Event_Enable property.~~

[Add following to **12**, p. 137, ahead of UNRELIABLE_OTHER]

MULTI_STATE_FAULT The Present_Value of the Multi-state object is equal to one of the states in the Fault_Values property and no other fault has been detected.
ASN.1 changes

[Add following enumeration to **21**, p. 379]

```
BACnetReliability ::= ENUMERATED {
. . .
          multi-state-fault          (9),
```

135*b*-11. Add a Protocol_Revision property to the Device object type.

As the BACnet standard evolves, it will be desirable to know what version of the standard a device was manufactured to meet. This should be a network visible identifier so that other devices may make runtime decisions based on this information.

To accomplish this, it is proposed to add a new mandatory property to the Device object type that will contain a numeric revision number. This number will act as a minor revision number to the BACnet standard. The value shall start with 0 and be ever-increasing. This value shall be determined by SSPC 135, and shall generally be incremented when substantive changes are made to the BACnet standard that affect device communication or behavior. This revision number shall revert to 0 whenever the Protocol_Version property is incremented.

A new Annex K shall record a history of changes made to this revision number along with the corresponding changes made to the standard.

The absence of this mandatory property in a device indicates that the device is, by definition, revision 0 of version 1. For this reason, the first value for the Protocol_Revision property shall be 1.

[Addition to **Table 12-11**, p. 176]

Property Identifier	Property Datatype	Conformance Code
…	…	…
Protocol_Version	Unsigned	R
Protocol_Revision	*Unsigned*	*R*
Protocol_Conformance_Class	Unsigned(1..6)	R
…	…	…

[Change **12.9.12**, p. 178]

12.9.12 Protocol_Version

This property, of type Unsigned, represents the version of the BACnet protocol supported by this BACnet Device. Every *major* revision of BACnet shall increase this version number by one. The initial release of BACnet shall be version 1

[Add a new **12.9.33**]

12.9.33 Protocol_Revision

This property, of type Unsigned, shall indicate the minor revision level of the BACnet standard. This value shall start at 1 and be incremented for any substantive change(s) to the BACnet standard that affect device communication or behavior. This value shall revert to zero upon each change to the Protocol_Version property. Changes to the values for Protocol_Version and Protocol_Revision are recorded in Annex K.

This property is required for all devices implementing BACnet Protocol_Version 1, Protocol_Revision 1 and above. Absence of this property shall indicate a device implemented to Protocol_Version 1, Protocol_Revision 0.

[Change **21**, pp. 375 and 376]

BACnetPropertyIdentifier ::= SEQUENCE {

```
        . . .
        protocol-object-types-supported      (96),
        protocol-revision                    (139),
        protocol-services-supported          (97),
        . . .
        weekly-schedule                 (123),
        -- see protocol-revision        (139),
    }
```

[Add to **Annex C** DEVICE definition, p. 414]

```
    DEVICE ::= SEQUENCE {
        . . .
        protocol-revision          [139]    Unsigned
        . . .
    }
```

[Add to **Annex D.9**, Examples of a Device Object, pp. 424 and 425]

```
    Example 1:  A "sophisticated" BACnet device.
    . . .
    Property:        Protocol_Version =              1
    Property:        Protocol_Revision =             1
    Property:        Protocol_Conformance_Class      6
    . . .
    Example 2:  A "simple" BACnet device.
    . . .
    Property:        Protocol_Version =              1
    Property:        Protocol_Revision =             1
    Property:        Protocol_Conformance_Class      6
    . . .
```

[Add new **Annex K**]

Annex K - History of Revisions (Informative)

Protocol		*Date Approved*
Version	*Revision*	*Summary of Changes to the Standard*
1	**NA**	ANSI/ASHRAE 135-1995
1	**NA**	Addendum *a* to ANSI/ASHRAE 135-1995 1. Add Annex J - BACnet/IP and supporting definitions
1	**1**	Addendum *b* to ANSI/ASHRAE 135-1995 1. Inconsistencies are eliminated in the definitions of the Analog and Binary Value object types 2. Any device that receives and executes UnconfirmedEventNotification service requests must support programmable process identifiers 3. Modify each event-generating object type to contain the last timestamp for each acknowledgeable transition 4. Modify the Notification Class object by requiring that the 'Notification Class' property be equivalent to the instance number of the Notification Class object 5. Modify the Event Notification services to make the 'To State' parameter mandatory for notifications of type ACK_NOTIFICATION 6. A new BACnetDeviceObjectPropertyReference production is added and its use in the Event Enrollment and Schedule object types is specified

		7. Add a Multi-state Value object type 8. Add an Averaging object type 9. Change all 'Process Identifier' properties and parameters to Unsigned32 10. Change the Multi-state Input object type to correct flaws related to fault detection and reporting and achieve consistency with the proposed Multi-state Value object type 11. Add a Protocol_Revision property to the Device object type 12. The File object type is changed to allow truncation and partial deletion operations 13. A new ReadRange service is added to permit reading a range of data items from a property whose datatype is a list or array of lists 14. A new UTCTimeSynchronization service is introduced and related changes are made to properties in the Device object type 15. Add a Trend Log object type 16. The UnconfirmedCOVNotification service is extended to allow notifications without prior subscription as a means of distributing globally important data to a potentially large number of recipients 17. Add eight new BACnet engineering units.

NA = Not Applicable because the Protocol_Revision property was first defined in Addendum 135 *b*

135*b*-12. The File object type is changed to allow truncation and partial deletion operations.

BACnet does not currently provide a way to truncate or delete a file except by deleting the entire file object. If an attempt is made to write from the beginning of the file or from an intermediate point, the standard does not currently specify what is to happen if the length of the new file data is less than the length of the old file data.

The proposed solution requires the File_Size property to be writable for STREAM_ACCESS files if the file size can be changed by writing to the file. Deleting a file will now be accomplished by writing 0 to File_Size. Truncating a file at octet X will be accomplished by writing X to File_Size.

A new property called Record_Count is added to the File object. This property is required to be writable for RECORD_ACCESS files if the number of records can be changed by writing to the file. It functions in a manner analogous to the File_Size property. Deleting a file is accomplished by writing 0 to Record_Count. Truncating a file at record X is accomplished by writing X to Record_Count.

Note that if the AtomicWriteFile service is used to begin writing at position X the file size does not change unless the new file data extends past the old end of the file. Thus in order to guarantee that the file ends with the new data it is necessary to first truncate the file at a position p such that $X \le p \le X+n$, where n is the number of octets or records to be written, before initiating the AtomicWriteFile service request.

This proposal also deletes the RECORD_AND_STREAM_ACCESS file type. This file type does not appear to be useful and it is difficult to define the behavior of such a file type when actions are taken to change its size.

[Change **Table 12-14**, p. 186]

Property Identifier	Property Datatype	Conformance Code
File_Size	Unsigned	R[1]
...	...	...
Record_Count	*Unsigned*	*O*[2]

[1]*If the file size can be changed by writing to the file, and File_Access_Method is STREAM_ACCESS, then this property shall be writable.*

[2] *This property shall be present only if File_Access_Method is RECORD_ACCESS. If the number of records can be changed by writing to the file, then this property shall be writable.*

[Change **12.11**, p. 186]

12.11.6 File_Size

This property, of type Unsigned, indicates the size of the file data in octets. *If the size of the file can be changed by writing to the file, and File_Access_Method is STREAM_ACCESS, then this property shall be writable.*

Writing to the File_Size property with a value less than the current size of the file shall truncate the file at the specified position. Writing a File_Size of 0 shall delete all of the file data but not the File object itself. Writing to the File_Size property with a value greater than the current size of the file shall expand the size of the file but the value of the new octets of the file shall be a local matter.

Devices may restrict the allowed values for writes to the File_Size. Specifically, devices may allow deletion of the file contents by writing a value of zero, but not necessarily allow arbitrary truncation or expansion.

12.11.10 File_Access_Method

This property, of type BACnetFileAccessMethod, indicates the type of file access supported for this object. The possible values for File_Access_Method are

{RECORD_ACCESS, STREAM_ACCESS, ~~RECORD_AND_STREAM_ACCESS~~}.

12.11.11 Record_Count

This property, of type Unsigned, indicates the size of the file data in records. The Record_Count property shall be present only if File_Access_Type is RECORD_ACCESS. If the number of records can be changed by writing to the file, then this property shall be writable.

Writing to the Record_Count property with a value less than the current size of the file shall truncate the file at the specified position. Writing a Record_Count of 0 shall delete all of the file data but not the File object itself. Writing to the Record_Count property with a value greater than the current size of the file shall expand the size of the file but the value of the new octets of the file shall be a local matter.

Devices may restrict the allowed values for writes to the File_Size. Specifically, devices may allow deletion of the file contents by writing a value of zero, but not necessarily allow arbitrary truncation or expansion.

[Change **21**. p. 372]

```
BACnetFileAccessMethod ::= ENUMERATED {
        record-access                        (0),
        stream-access                        (1)~~,~~
~~        record-and-stream-access             (2)~~}
```

[Change **21**, p. 375-376]

```
BACnetPropertyIdentifier ::= ENUMERATED {
        ...
        recipient-list                       (102),
        record-count                         (141),
        reliability                          (103),
        ...
        weekly-schedule                      (123)
-- see record-count                          (141),
        ...
        }
```

[Change **Annex C**, p.415]

```
File ::= SEQUENCE{
        ...
        record-count            [141] Unsigned OPTIONAL
        }
```

[Change **Annex D**, p. 427]

D.11 Example of a File Object

A File Object holding trend information:

Property:	File_Size=	*750*
Property:	*Record_Count =*	*150*

135*b*-13. A new ReadRange service is added to permit reading a range of data items from a property whose datatype is a list or array of lists.

This proposed service permits reading a specific range of data items from properties that are either lists or arrays of lists. In particular, the ReadRange service will facilitate the reading of records from the Trend Log object type proposed in 135b-15. The "range" of readings can be specified either by position within the list of by the use of time references.

[Existing clauses 15.8 through 15.9 shall be renumbered to 15.9 through 15.10. Existing Tables 15-13 through 15-15 shall be renumbered to 15-15 through 15-17.]

15.8 ReadRange Service

The ReadRange service is used by a client BACnet-user to read a specific range of data items representing a subset of data available within a specified object property. The service may be used with any list or array of lists property.

15.8.1 Structure

The structure of the ReadRange primitive is shown in Table 15-13. The terminology and symbology used in this table are explained in 5.6.

Table 15-13. Structure of ReadRange Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Object Identifier	M	M(=)		
Property Identifier	M	M(=)		
Property Array Index	C	C(=)		
Range	U	U(=)		
Result(+)			S	S(=)
Object Identifier			M	M(=)
Property Identifier			M	M(=)
Property Array Index			C	C(=)
Result Flags			M	M(=)
Item Count			M	M(=)
Item Data			M	M(=)
Result(-)			S	S(=)
Error Type			M	M(=)

15.8.1.1 Argument

This parameter shall convey the parameters for the ReadRange confirmed service request.

15.8.1.1.1 Object Identifier

This parameter, of type BACnetObjectIdentifier, specifies the object and property is to be read.

15.8.1.1.2 Property Identifier

This parameter, of type BACnetPropertyIdentifier, specifies the property to be read by this service. Because this service is intended to read a single property of a single object, the value of this parameter shall not be one of the special property identifiers ALL, REQUIRED, or OPTIONAL.

15.8.1.1.3 Property Array Index

If the property identified above is of datatype array of lists, this optional parameter of type Unsigned shall indicate the array index of the element of the property referenced by this service. If the property identified above is not of datatype array of lists, this parameter shall be omitted. The index value shall not be zero.

15.8.1.1.4 Range

This optional parameter shall convey criteria for the consecutive range items within the referenced property that are to be returned, as described in 15.8.2. The 'Range' parameter is shown in Table 15-14. The terminology and symbology used in this table are explained in 5.6.

Table 15-14. Structure of the 'Range' Parameter

Parameter Name	Req	Ind	Datatype
By Position	S	S(=)	
Reference Index	M	M(=)	Unsigned
Count	M	M(=)	INTEGER
By Time	S	S(=)	
Reference Time	M	M(=)	BACnetDateTime
Count	M	M(=)	INTEGER
Time Range	S	S(=)	
Beginning Time	M	M(=)	BACnetDateTime
Ending Time	M	M(=)	BACnetDateTime

15.8.1.1.4.1 By Position

The 'By Position' parameter shall indicate that the particular items to be read are referenced by an index.

15.8.1.1.4.1.1 Reference Index

The 'Reference Index' parameter specifies the index of the first (if 'Count' is positive) or last (if 'Count' is negative) item to be read.

15.8.1.1.4.1.2 Count

The absolute value of the 'Count' parameter specifies the number of records to be read. If 'Count' is positive, the record specified by 'Reference Index' shall be the first and oldest record read and returned; if 'Count' is negative the record specified by 'Reference Index' shall be the last and newest record. 'Count' may not be zero.

15.8.1.1.4.2 By Time

The 'By Time' parameter shall indicate that the particular items to be read are referenced by timestamp.

15.8.1.1.4.2.1 Reference Time

The first (if 'Count' is positive) or last (if 'Count' is negative) item to be read shall be the first item with a timestamp newer than the time specified by the 'Reference Time' parameter.

15.8.1.1.4.2.2 Count

The absolute value of the 'Count' parameter specifies the number of records to be read. If 'Count' is positive, the record specified by 'Reference Time' shall be the first and oldest record read and returned; if

'Count' is negative the record specified by 'Reference Index' shall be the last and newest record. 'Count' may not be zero.

15.8.1.1.4.3 Time Range

The 'Time Range' parameter shall indicate that the particular items to be read have timestamps within a specified range.

15.8.1.1.4.3.1 Beginning Time

The first item to be read shall be the first item with a timestamp newer than the time specified by the 'Beginning Time' parameter.

15.8.1.1.4.4.2 Ending Time

The last item to be read shall be the last item with a timestamp older than or equal to the time specified by the 'Ending Time' parameter.

15.8.1.2 Result(+)

The 'Result(+)' parameter shall indicate that the service request succeeded. A successful result includes the following parameters.

15.8.1.2.1 Object Identifier

This parameter, of type BACnetObjectIdentifier, specifies the object that was read.

15.8.1.2.2 Property Identifier

This parameter, of type BACnetPropertyIdentifier, shall identify that property that was read.

15.8.1.2.3 Property Array Index

If the property identified above is of datatype array of lists, this parameter of type Unsigned shall indicate the array index of the element of the property referenced by this service.   If the property identified above is not of datatype array of lists, this parameter shall be omitted.

15.8.1.2.4 Result Flags

This parameter, of type BACnetRangeFlags, shall convey several flags that describe characteristics of the response data:

{FIRSTITEM, LASTITEM, MOREITEMS}

The FIRSTITEM flag indicates whether this response includes the first list or array element (in the case of positional indexing), or the oldest timestamped item (in the case of time indexing).

The LASTITEM flag indicates whether this response includes the last list or array element (in the case of positional indexing), or the newest timestamped item (in the case of time indexing)

The MOREITEMS flag indicates whether more items matched the request but were not transmittable within the PDU.

15.8.1.2.5 Item Count

This parameter, of type Unsigned, represents the number of items that were returned.

15.8.1.2.6 Item Data

This parameter consists of a list of the requested data.

15.8.1.3 Result(-)

The 'Result(-)' parameter shall indicate that the service request has failed. The reason for the failure shall be specified by the 'Error Type' parameter.

15.8.1.3.1 Error Type

This parameter consists of two component parameters: (1) the 'Error Class' and (2) the 'Error Code'. See Clause 18.

15.8.2 Service Procedure

The responding BACnet-user shall first verify the validity of the 'Object Identifier', 'Property Identifier' and "Property Array Index' parameters and return a 'Result(-)' response with the appropriate error class and code if the object or property is unknown, if the referenced data is not a list or array, or if it is currently inaccessible for another reason.

If the 'Range' parameter is not present, then the responding BACnet-user shall read and attempt to return all of the available items in the list or array.

If the 'Range' parameter is present and specifies the 'By Position' parameters, then the responding BACnet-user shall read and attempt to return all of the items specified. The items specified include the item at the index specified by 'Reference Index' plus up to 'Count'-1 items following if 'Count' is positive, or up to -1-'Count' items preceding if 'Count' is negative. Array index 0 shall not be returned by this service; lists shall begin with index 1.

If the 'Range' parameter is present and specifies the 'By Time' parameters, then the responding BACnet-user shall read and attempt to return all of the items specified. If 'By Time' parameters are specified and the property values are not timestamped an error shall be returned. The items specified include the first item with a timestamp newer than 'Reference Time' plus up to 'Count'-1 items following if 'Count' is positive, or up to -1-'Count' items preceding if 'Count' is negative. Array index 0 shall not be returned by this service; lists shall begin with index 1.

If the 'Range' parameter is present and specifies the 'Time Range' parameters, then the responding BACnet-user shall read and attempt to return all of the items specified. If 'Time Range' parameters are specified and the property values are not timestamped an error shall be returned. The items specified include all items with a timestamp newer than 'Beginning Time' and less than or equal to 'Ending Time'. If any field of the 'Beginning Time' parameter contains a wildcard value, then all times shall be considered newer than 'Beginning Time'. If any fields of the 'Ending Time' parameter contains a wildcard value then all times shall be considered older than 'Ending Time'.

To avoid missing items when using chained time-based reads, items with a timestamp equal to the current time of the device shall not be included in the response. Items with the same timestamp shall be returned atomically . If items are read that match the request parameters but cannot be returned in the response, the 'Result Flags' parameter shall contain the MOREITEMS flag set to TRUE; otherwise it shall be FALSE. Remaining items may be obtained with subsequent requests specifying appropriately chosen parameters.

The returned response shall convey the number of items read and returned using the 'Item Count' parameter. The actual items shall be returned in the 'Item Data' parameter. If the returned response includes the first positional index and a 'By Position' request had been made, or the oldest timestamped item and a

'By Time' or 'Time Range' request had been made, then the 'Result Flags' parameter shall contain the FIRSTITEM flag set to TRUE; otherwise it shall be FALSE.

If the returned response includes the last positional index and a 'By Position' request had been made, or the newest timestamped item and a 'By Time' or 'Time Range' request had been made, then the 'Result Flags' shall contain the LASTITEM flag set to TRUE; otherwise it shall be FALSE.

[Add new definition in **3**, p. 5, and renumber **3.2.55** and **3.2.56** to **3.2.56** and **3.2.57**]

3.2.55 timestamp: The date and time recorded for and accompanying the record of an event or operation.

[Add to **21**, p. 350, BACnetConfirmedServiceChoice (under Object Access Services)]

```
readRange                (26),
```

[Add to **21**, p. 351, BACnet-Confirmed-Service-Request (under Object Access Services)]

```
ReadRange                [26] ReadRange-Request,
```

[Add to **21**, p. 352, BACnet-Confirmed-Service-ACK (under Object Access Services)]

```
ReadRange                [26] ReadRange-ACK,
```

[Add to **21**, pp. 355-357 under Confirmed Object Access Services]

```
ReadRange-Request ::= SEQUENCE {
        objectIdentifier        [0]  BACnetObjectIdentifier,
        propertyIdentifier      [1]  BACnetPropertyIdentifier,
        propertyArrayIndex      [2]  Unsigned OPTIONAL          -- used only with array datatype
        range                    CHOICE {
                                        byPosition      [3]  SEQUENCE {
                                                        referenceIndex  Unsigned,
                                                        count           INTEGER
                                                        },
                                        byTime          [4]  SEQUENCE {
                                                        referenceTime   BACnetDateTime,
                                                        count           INTEGER
                                                        },
                                        timeRange       [5]  SEQUENCE {
                                                        beginningTime   BACnetDateTime,
                                                        endingTime      BACnetDateTime
                                                        }
                                        } OPTIONAL
        }

ReadRange-ACK ::= SEQUENCE {
        objectIdentifier        [0]  BACnetObjectIdentifier,
        propertyIdentifier      [1]  BACnetPropertyIdentifier,
        propertyArrayIndex      [2]  Unsigned OPTIONAL,  -- used only with array datatype
        resultFlags             [3]  BACnetResultFlags,
        itemCount               [4]  Unsigned,
        itemData                [5]  SEQUENCE OF ABSTRACT-SYNTAX.&TYPE
        }
```

[Add to **21**, p. 361, BACnet-Error (under Object Access Services)]

```
ReadRange                [26] Error,
```

[Add to **21**, p. 379, in BACnet Base Types]

```
BACnetResultFlags ::= BIT STRING {
                    firstitem    (0),
                    lastitem     (1),
                    moreitems    (2)
        }
```

[Add to **Annex E**, p. 447, and renumber **E.3.8** and **E.3.9** to **E.3.9** and **E.3.10**, respectively]

E.3.8 Example of the ReadRange Service

Assumed objects:	Object_Identifier	Object_Name	Object_Type
	(Trend Log, Instance 1)	ROOM3TEMP	TREND_LOG

We wish to look at all the records for the last five minutes within a Trend Log's Log Buffer.

```
Service                          = ReadRange
 'ObjectIdentifier'      = (Trend Log, Instance 1)
 'PropertyIdentifier'            = Log_Buffer
 'Range'
        'Time Range'
        'Beginning Time'         = (23-MAR-1998, 19:52:34.0)
        'Ending Time'            = (23-MAR-1998, 19:57:34.0)
```

A typical result might be:

```
'Result Flags'          = (TRUE, TRUE, FALSE)
'Item Count'            = 2
'Item Data'             = (((23-MAR-1998, 19:54:27.0), 18.0, (FALSE,FALSE,FALSE,FALSE)),
                              ((23-MAR-1998, 19:56:27.0), 18.1, (FALSE,FALSE,FALSE,FALSE)))
```

[Add to **Annex F**, p. 475, and renumber **F.3.8** and **F.3.9** to **F.3.9** and **F.3.10**, respectively]

F.3.8 Encoding for Example E.3.8 - ReadRange Service

Example 1: Reading records from a Trend Log object.

```
X'02'                   PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=1)
X'02'                   Maximum APDU Size Accepted = 206 octets
X'01'                   Invoke ID = 1
X'1A'                   Service Choice = (n), (ReadRange-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'04800001'             Trend Log, Instance Number = 1
X'19'                   SD Context Tag 1 (Property Identifier, L=1)
X'83'                   131 (LOG_BUFFER)
X'5E'                   PD Opening Tag 5 (Time Range)
        X'A4'                   Application Tag 10 (Date, L=4)
        X'620317FF'             March 23, 1998 (Day Of Week Unspecified)
        X'B4'                   Application Tag 11, (Time, L=4)
```

```
          X’13342200’         19:52:34.0
          X’A4’               Application Tag 10 (Date, L=4)
          X’620317FF’         March 23, 1998 (Day Of Week Unspecified)
          X’B4’               Application Tag 11, (Time, L=4)
          X’13392200’         19:57:34.0
X’5F’                   PD Closing Tag 5 (Time Range)
```

Assuming the service procedure executes correctly, a complex acknowledgement is returned containing the requested data:

```
X’30’                   PDU Type = 3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X’01’                   Invoke ID=01
X’1A’                   Service ACK Choice = (n), (ReadRange-ACK)

X’0C’                   SD Context Tag 0 (Object Identifier, L=4)
X’04800001’             Trend Log, Instance Number = 1
X’19’                   SD Context Tag 1 (Property Identifier, L=1)
X’83’                   131 (LOG_BUFFER)
X’3A’                   SD Context Tag 3 (Result Flags, L=2)
X’05C0’                 1,1,0 (TRUE, TRUE, FALSE)
X’49’                   SD Context Tag 4 (Item Count, L=1)
X’02’                   2
X’5E’                   PD Opening Tag 5 (Item Data)
          X’0E’               PD Opening Tag 0 (Timestamp)
                    X’A4’               Application Tag 10 (Date, L=4)
                    X’62031701’         Monday, March 23, 1998
                    X’B4’               Application Tag 11, (Time, L=4)
                    X’13362B00’         19:54:27.0
          X’0F’               PD Closing Tag 0 (Timestamp)
          X’1E’               PD Opening Tag 1 (Log Datum)
                    X’2C’               SD Context Tag 2 (REAL, L=4)
                    X’41900000’         18.0
          X’1F’               PD Closing Tag 1 (Log Datum)
          X’2A’               SD Context Tag 2 (Status Flags, L=2)
          X’0400’             0,0,0,0 (FALSE, FALSE, FALSE, FALSE)
          X’0E’               PD Opening Tag 0 (Timestamp)
                    X’A4’               Application Tag 10 (Date, L=4)
                    X’62031701’         Monday, March 23, 1998
                    X’B4’               Application Tag 11, (Time, L=4)
                    X’13382B00’         19:56:27.0
          X’0F’               PD Closing Tag 0 (Timestamp)
          X’1E’               PD Opening Tag 1 (Log Datum)
                    X’2C’               SD Context Tag 2 (REAL, L=4)
                    X’4190CCCD’         18.1
          X’1F’               PD Closing Tag 1 (Log Datum)
          X’2A’               SD Context Tag 2 (Status Flags, L=2)
          X’0400’             0,0,0,0 (FALSE, FALSE, FALSE, FALSE)
X’5F’                   PD Closing Tag 4 (Item Data)
```

135*b*-14. A new UTCTimeSynchronization service is introduced and related changes are made to properties in the Device object type.

These changes mandate the use of Universal Time Coordinated (UTC) in a new UTCTimeSynchronization service and related properties of the Device object type. This is accomplished by having the UTCTimeSynchronization service to refer to UTC rather than "current time" and having the service procedure indicate the manner in which the 'Time' parameter is used to calculate the correct local time. Also, the Device object properties 'Local_Time' and 'Local_Date', shall become required if a device supports the execution of the TimeSynchronization or UTCTimeSynchronization services, and in addition the Device object properties 'UTC_Offset' and 'Daylight_Savings_Status' shall become required if a device supports execution of the UTCTimeSynchronization service. The following clauses represent those changes:

16.X UTCTimeSynchronization Service

The UTCTimeSynchronization service is used by a requesting BACnet-user to notify one or more remote devices of the correct Universal Time Coordinated (UTC). This service may be broadcast, multicast, or addressed to a single recipient. Its purpose is to notify recipients of the correct UTC so that devices may synchronize their internal clocks with one another.

16.X.1 Structure

The structure of the UTCTimeSynchronization service primitive is shown in Table 16-X. The terminology and symbology used in this table are explained in 5.6.

Table 16-X. Structure of UTCTimeSynchronization Service Primitive

Parameter Name	Req	Ind
Argument	M	M(=)
Time	M	M(=)

16.X.1.1 Argument

The 'Argument' parameter shall convey the parameters for the UTCTimeSynchronization service request.

16.X.1.1.1 Time

This parameter, of type BACnetDateTime, shall convey the UTC date and time.

16.X.2 Service Procedure

Since this is an unconfirmed service, no response primitives are expected. A device receiving a UTCTimeSynchronization service indication shall update its local representation of time and date by subtracting the value of the 'UTC_Offset' property of the Device object from the 'Time' parameter and taking the 'Daylight_Savings_Status' property of the Device object into account as appropriate to the locality. This change shall be reflected in the Local_Time and Local_Date properties of the Device object.

No restrictions on the use of this service exist when it is invoked at the request of an operator. Otherwise, the initiation of this service by a device is controlled by the value of the Time_Synchronization_Recipient property in the Device object. When the Time_Synchronization_Recipient list is of length zero, a device may not automatically send a TimeSynchronization request. When Time_Synchronization_Recipient list is of length one or more, a device may automatically send a TimeSynchronization request but only to the devices or addresses contained in the Time_Synchronization_Recipient list.

[Change **Table 12-11**, p. 176]

Property Identifier	Property Datatype	Conformance Code
Local_Time	Time	*O*[2,3]
Local_Date	Date	*O*[2,3]
UTC_Offset	INTEGER	*O*[3]
Daylight_Savings_Status	BOOLEAN	*O*[3]

[2] *If the device supports the execution of the TimeSynchronization service, then these properties shall be present.*
[3] *If the device supports the execution of the UTCTimeSynchronization service, then these properties shall be present.*

[Change the superscript notes for **Table 12-11**. They shall be renumbered such that the previous superscripts 2..4 become 4..6 so that the newly added superscripts 2 and 3 remain vertically ordinal.]

[Change **12.9.23**, p. 179**]**

12.9.23 UTC_Offset

The UTC_Offset property, of type INTEGER, shall indicate the number of minutes (*-780 to +780*) offset between local standard time and Universal Time Coordinated (UTC). The time zones to the west of the zero degree meridian shall be positive values, and those to the east shall be negative values. *The value of the UTC_Offset property is subtracted from the UTC received in UTCTimeSynchronization service requests to calculate the correct local standard time.*

[Change **21**, pp. 359-360]

BACnetUnconfirmedServiceChoice ::= ENUMERATED {

i-Am	(0),
i-Have	(1),
unconfirmedCOVNotification	(2),
unconfirmedEventNotification	(3),
unconfirmedPrivateTransfer	(4),
unconfirmedTextMessage	(5),
timeSynchronization	(6),
who-Has	(7),
who-Is	(8),
utcTimeSynchronization	*(9)*

}

BACnet-Unconfirmed-Service-Request ::= CHOICE {

i-Am	[0]	I-Am-Request,
i-Have	[1]	I-Have-Request,
unconfirmedCOVNotification	[2]	UnconfirmedCOVNotification-Request,
unconfirmedEventNotification	[3]	UnconfirmedEventNotification-Request,
unconfirmedPrivateTransfer	[4]	UnconfirmedPrivateTransfer-Request,
unconfirmedTextMessage	[5]	UnconfirmedTextMessage-Request,
timeSynchronization	[6]	TimeSynchronization-Request,
who-Has	[7]	Who-Has-Request,

```
        who-Is                              [8]  Who-Is-Request,
        utcTimeSynchronization              [9]  UTCTimeSynchronization-Request
        }

UTCTimeSynchronization-Request  ::= SEQUENCE {
        time                                BACnetDateTime
        }
```

[Change **21**, pp. 379-380]

```
BACnetServicesSupported ::= BIT STRING {
    …
    timeSynchronization             (32),
    who-Has                         (33),
    who-Is                          (34),
    readRange                       (35),
    utcTimeSynchronization          (36)
    }
```

135*b*-15. Add a Trend Log object type.

One of the tasks deferred by SPC 135P was to define the format of trend logs. The solution proposed here is to define a new object type, the Trend Log, that represents the network-visible attributes of such a log.

In the proposed clause below, the value of 'X' in the clause numbering will be replaced at the time of publication with a value placing it in the proper sequence in Clause 12, dependent on the potential addition of other object types.

12.X Trend Log Object Type

A Trend Log object monitors a property of a referenced object and, when predefined conditions are met, saves ("logs") the value of the property and a timestamp in an internal buffer for subsequent retrieval. The data may be logged periodically or upon a change of value. Errors that prevent the acquisition of the data, as well as changes in the status or operation of the logging process itself, are also recorded. Each timestamped buffer entry is called a trend log "record."

The referenced object may reside in the same device as the Trend Log object or in an external device. The referenced property's value may be recorded upon COV subscription or periodic poll. Where status flags are available (such as when the COVNotification or ReadPropertyMultiple services are used), they are also acquired and saved with the data.

Each Trend Log object maintains an internal, optionally fixed-size, buffer. This buffer fills or grows as log records are added. If the buffer becomes full, the least recent record is overwritten when a new record is added, or collection may be set to stop. Trend Log records are transferred as BACnetLogRecords using the ReadRange service. The buffer may be cleared by writing a zero to the Record_Count property.

Several datatypes are defined for storage in the log records. The ability to store ANY datatypes is optional. Data stored in the log buffer may be optionally restricted in size to 32 bits, as in the case of bit strings, to facilitate implementation in devices with strict storage requirements.

Logging may be enabled and disabled through the Log_Enable property and at dates and times specified by the Start_Time and Stop_Time properties. Trend Log enabling and disabling is recorded in the log buffer.

Event reporting (notification) may be provided to facilitate automatic fetching of log records by processes on other devices such as fileservers. Support is provided for algorithmic reporting; optionally, intrinsic reporting may be provided.

In intrinsic reporting, when the number of records specified by the Notification_Threshold property have been collected since the previous notification (or startup), a new notification is sent to all subscribed devices. BUFFER_READY algorithmic reporting is described in Clause 13.3.7.

In response to a notification, subscribers fetch the new records with a ReadRange service request using the Current_Notify_Time of the preceding notification as the 'Beginning Date' parameter and the Current_Notify_Time of the current notification as the 'Ending Date' parameter.

A missed notification may be detected by a subscriber if the Current_Notify_Time it received in its previous notification is older than the Previous_Notify_Time parameter of the current notification. If the ReadRange-ACK response to the ReadRange request issued under these conditions has its 'firstitem' flag set to TRUE, Trend Log records have probably been missed by this subscriber.

The acquisition of log records by remote devices has no effect upon the state of the Trend Log object itself. This allows completely independent, but properly sequential, access to its log records by all remote devices. Any remote device can independently update its records at any time.

Table 12-23. Properties of the Trend Log Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Log_Enable	BOOLEAN	W
Start_Time	BACnetDateTime	O[1,2]
Stop_Time	BACnetDateTime	O[1,2]
Log_DeviceObjectProperty	BACnetDeviceObjectPropertyReference	O[1]
Log_Interval	Unsigned	O[1,2]
COV_Resubscription_Interval	Unsigned	O
Client_COV_Increment	BACnetClientCOV	O
Stop_When_Full	BOOLEAN	R
Buffer_Size	Unsigned32	R
Log_Buffer	List of BACnetLogRecord	R
Record_Count	Unsigned32	W
Total_Record_Count	Unsigned32	R
Notification_Threshold	Unsigned32	O[3]
Records_Since_Notification	Unsigned32	O[3]
Previous_Notify_Time	BACnetDateTime	O[3]
Current_Notify_Time	BACnetDateTime	O[3]
Event_State	BACnetEventState	R
Notification_Class	Unsigned	O[3]
Event_Enable	BACnetEventTransitionBits	O[3]
Acked_Transitions	BACnetEventTransitionBits	O[3]
Notify_Type	BACnetNotifyType	O[3]
Event_Time_Stamps	BACnetARRAY[3] of BACnetTimeStamp	O[3]

[1] These properties are required to be present if the monitored property is a BACnet property.
[2] If present, these properties are required to be writable.
[3] These properties are required to be present if the object supports intrinsic reporting.

12.X.1 Object_Identifier

This property, of type BACnetObjectIdentifier, is a numeric code that is used to identify the object. It shall be unique within the BACnet Device that maintains it.

12.19.2 Object_Name

This property, of type CharacterString, shall represent a name for the Object that is unique within the BACnet Device that maintains it. The minimum length of the string shall be one character. The set of characters used in the Object_Name shall be restricted to printable characters.

12.X.3. Object_Type

This property, of type BACnetObjectType, indicates membership in a particular object type class. The value of this property shall be TREND_LOG.

12.X.4 Description

This property, of type CharacterString, is a string of printable characters whose content is not restricted.

12.X.5 Log_Enable

This property, of type BOOLEAN, indicates and controls whether (TRUE) or not (FALSE) logging is enabled. A value of FALSE overrides the time interval defined by Start_Time and Stop_Time. If logging is otherwise enabled by the Start_Time and Stop_Time properties, changes to the value of the Log_Enable property shall be recorded in the log. When the device begins operation the value TRUE shall be recorded in the log.

12.X.6 Start_Time

This property, of type BACnetDateTime, specifies the date and time at or after which logging shall be enabled by this property. If any of the fields of the BACnetDateTime contain "wildcard" values, then the conditions for logging to be enabled by Start_Time shall be ignored. If Start_Time specifies a date and time after Stop_Time then logging shall be disabled. This property must be writable if present.

12.X.7 Stop_Time

This property, of type BACnetDateTime, specifies the date and time at or after which logging shall be disabled by this property. If any of the fields of the BACnetDateTime contain "wildcard" values, then the conditions for logging to be enabled by Stop_Time shall be ignored. If Stop_Time specifies a date and time earlier than Start_Time then logging shall be disabled. This property must be writable if present.

12.X.8 Log_DeviceObjectProperty

This property, of type BACnetDeviceObjectPropertyReference, specifies the Device Identifier, Object Identifier and Property Identifier of the property to be trend logged.

If this property is writable, it may be restricted to reference only objects inside the device containing the Trend Log object. If the property is restricted to referencing objects within the containing device, an attempt to write a reference to an object outside the containing device into this property shall cause a Result(-) to be returned.

12.X.9 Log_Interval

This property, of type Unsigned, specifies the periodic interval in hundredths of seconds for which the referenced property is to be logged. If this property has the value zero then the Trend Log shall issue COV subscriptions for the referenced property. The value of this property must be non-zero if COV_Resubscription_Interval is not present. This property must be writable if present.

12.X.10 COV_Resubscription_Interval

If the Trend Log is acquiring data from a remote device by COV subscription this property, of type Unsigned, specifies the number of seconds between COV resubscriptions, if COV subscription is in effect. SubscribeCOV requests shall specify twice this lifetime for the subscription and shall specify the issuance of confirmed notifications. If COV subscriptions are in effect, the first COV subscription is issued when the Trend Log object begins operation or when Log_Enable becomes TRUE. If present, the value of this property must be non-zero. If this property is not present then COV subscription shall not be attempted.

12.9.11 Client_COV_Increment

If the Trend Log is acquiring COV data this property, of type BACnetClientCOV, specifies the increment to be used in determining that a change of value has occurred. If the referenced object and property supports COV reporting per 13.1, this property may have the value NULL; in this case change of value is determined by the criteria of 13.1.

12.X.12 Stop_When_Full

This property, of type BOOLEAN, specifies whether (TRUE) or not (FALSE) logging should cease when the buffer is full. When logging ceases, Log_Enable shall be set FALSE.

12.X.13 Buffer_Size

This property, of type Unsigned32, shall specify the maximum number of records the buffer may hold. If writable, it may not be written when Log_Enable is TRUE. The disposition of existing records when Buffer_Size is written is a local matter.

12.X.14 Log Buffer

This property is a list of up to Buffer_Size timestamped records of datatype BACnetLogRecord, each of which conveys a recorded data value, an error related to data-collection, or status changes in the Trend Log object. Each record has data fields as follows:

Timestamp	The date and time, in UTC, when the record was collected.
LogDatum	The data value read from the monitored object and property, an error encountered in an attempt to read a value, or a change in status or operation of the Trend Log object itself.
StatusFlags	The StatusFlags property of the monitored object, if present and available atomically associated with the LogDatum data value. If the StatusFlags property is not present or not available atomically associated with the data value, this item shall not be included in the log record.

The choices available for the LogDatum are listed below:

log-status	This choice represents a change in the status or operation of the Trend Log object. Whenever one of the events represented by the flags listed below occurs, except as noted, a record shall be appended to the buffer.	
	log-disabled	This flag is set whenever the Trend Log object is disabled, such as when Log_Enable is set to FALSE. Whenever the Trend Log object begins operation, this flag shall be presumed to have changed from TRUE to FALSE and a log entry shall be made.
	buffer-purged	This flag shall be set to TRUE whenever the buffer is deleted by a write of the value zero to the Record_Count property. After this value is recorded in the buffer, the subsequent immediate change to FALSE shall not be recorded.
boolean-value real-value enum-value unsigned-value signed-value bitstring-value null-value	These choices represent data values read from the monitored object and property.	

failure	This choice represents an error encountered in an attempt to read a data value from the monitored object. If the error is conveyed by an error response from a remote device the Error Class and Error Code in the response shall be recorded.
time-change	This choice represents a change in the clock setting in the device, it records the number of seconds by which the clock changed. If the number is not known, such as when the clock is initialized for the first time, the value recorded shall be zero.
any-value	This choice represents data values read from the monitored object and property.

The buffer is not network accessible except through the use of the ReadRange service with implicit reference to the Trend Log.

12.X.15 Record_Count

This property, of type Unsigned32, shall represent the number of records currently resident in the log buffer. A write of the value zero to this property shall cause all records in the log buffer to be deleted and Records_Since_Notification to be reset to zero. Upon completion, this event shall be reported in the log as the initial entry.

12.X.16 Total_Record_Count

This property, of type Unsigned32, shall represent the total number of records collected by the Trend Log object since creation. When the value of Total_Record_Count reaches its maximum possible value of $2^{32} – 1$, the next value it takes shall be zero. Once this value has wrapped to zero, its semantic value (the total number of records collected) has been lost but its use in generating notifications remains.

12.X.17 Notification_Threshold

This property, of type Unsigned32, shall specify the value of Records_Since_Notification at which notification occurs. This property is required if intrinsic reporting is supported by this object.

12.X.18 Records_Since_Notification

This property, of type Unsigned32, represents the number of records collected since the previous notification, or beginning of logging if no previous notification has occurred. This property is required if intrinsic reporting is supported by this object.

12.X.19 Previous_Notify_Time

This property, of type BACnetDateTime, represents the value that the property Current_Notify_Time had at the time of the previous notification. At the beginning of a notification operation this property is set to the value of Current_Notify_Time, then the property Current_Notify_Time is updated. If no previous notification has occurred this property shall contain all wildcard values. This property is required if intrinsic reporting is supported by this object.

12.X.20 Current_Notify_Time

This property, of type BACnetDateTime, represents the timestamp associated with the most recently collected record whose collection triggered a notification. If no notification has occurred since logging began this property shall contain all wildcard values. This property is required if intrinsic reporting is supported by this object.

12.X.21 Event_State

The Event_State property, of type BACnetEventState, is included in order to provide a way to determine if this object has an active event state associated with it. If the object supports intrinsic reporting, then the Event_State property shall indicate the event state of the object. If the object does not support intrinsic reporting, then the value of this property shall be NORMAL. The allowed states are NORMAL, and FAULT.

12.X.22 Notification_Class

This property, of type Unsigned, shall specify the notification class to be used when handling and generating event notifications for this object. The Notification_Class property implicitly refers to a Notification Class object that has a Notification_Class property with the same value. This property is required if intrinsic reporting is supported by this object.

12.X.23 Event_Enable

This property, of type BACnetEventTransitionBits, shall convey three flags that separately enable and disable reporting of TO_FAULT and TO_NORMAL events. In the context of Trend Log objects, the value of the Records_Since_Notification property becoming equal to or greater than the value of the Notification_Threshold property shall cause a NORMAL-NORMAL transition. The failure of an attempted COV subscription shall cause a TO_FAULT state transition. The TO_NORMAL transition must be enabled when intrinsic reporting is to be used; this shall be set by default. This property is required if intrinsic reporting is supported by this object.

12.X.24 Acked_Transitions

This property, of type BACnetEventTransitionBits, shall convey three flags that separately indicate the receipt of acknowledgements for TO-OFFNORMAL, TO-FAULT and TO-NORMAL events. These flags shall be cleared upon the occurrence of the corresponding event and set under any of these conditions:

(a) upon receipt of the corresponding acknowledgement;
(b) upon the occurrence of the event if the corresponding flag is <u>not set</u> in the Event_Enable property (meaning event notifications will not be generated for this condition and thus no acknowledgement is expected);
(c) upon the occurrence of the event if the corresponding flag is set in the Event_Enable property <u>and</u> the corresponding flag in the Ack_Required property of the Notification Class object implicitly referenced by the Notification_Class property of this object is <u>not set</u> (meaning no acknowledgement is expected).

12.X.25 Notify_Type

This property, of type BACnetNotifyType, shall convey whether the notifications generated by the object should be Events or Alarms. This property is required if intrinsic reporting is supported by this object.

12.X.26 Event_Time_Stamps

This optional property, of type BACnetARRAY [3] of BACnetTimeStamp, shall convey the times of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events, respectively. Time stamps of type Time or Date shall have 'FF' in each octet and Sequence number time stamps shall have the value 0 if no event notification of that type has been generated since the object was created. This property is required if intrinsic reporting is supported by this object.

[Change **12.10.5**, EventType, p. 182]

"… This parameter is an enumerated type that may have any of the following values:

{CHANGE_OF_BITSTRING, CHANGE_OF_STATE, CHANGE_OF_VALUE, COMMAND_FAILURE, FLOATING_LIMIT, OUT_OF_RANGE, BUFFER_READY}
"

[Add new Event_Type to **Table 12-13**, p. 182]

Event_Type	Event_State	Event_Parameters
...	...	...
BUFFER_READY	NORMAL	Notification_Threshold
...	...	…

[Add entry in **12.10.7**, Event_Parameters, p. 183]

Notification_Threshold	This parameter, of type Unsigned, applies to the BUFFER_READY algorithm. It specifies the value of Records_Since_Notification at which notification occurs.

[Add entry to **Table 13.2**, p. 220]

Object Type	Criteria	Event Type
...	...	...
Trend Log	If Event_State is NORMAL state and Records_Since_Notification is equal to Notification_Threshold	BUFFER_READY
...	...	…

[Add entry to **Table 13.3**, p. 221]

Object	Event Type	Notification Parameters	Referenced Object's Properties
...	...	...	...
Trend Log	BUFFER_READY	Buffer_Device Buffer_Object Previous_Notification Current_Notification	Object_Identifier Object_Identifier Previous_Notify_Time Current_Notify_Time
...	...	...	...

[Add entry to **Table 13.4**, p. 221]

Event Type	Notification Parameters	Description
...	...	...
BUFFER_READY	Buffer_Device Buffer_Object Previous_Notification Current_Notification	Object_Identifier of the device containing the buffer. Object_Identifier of the object containing the buffer. Current_Notify_Time of the previous notification sent. Current_Notify_Time of the current notification sent.
...	...	...

[Change **13.3**, p. 222]

~~Six~~ Seven event algorithms are specified in this standard because of their widespread occurrence in building automation and control systems. They are:

…

(g) BUFFER_READY

[Add new **13.3.7**]

13.3.7 BUFFER_READY Algorithm

A BUFFER_READY occurs when the number of records specified by Notification_Threshold have been entered into the log since the start of operation or the previous notification, whichever is most recent. The number of records collected is determined by the formula Total_Record_Count – Previous_Notification_Count, if Total_Record_Count is greater than or equal to Previous_Notification_Count, otherwise it is determined by the formula Total_Record_Count – Previous_Notification_Count + 2^{32}. Upon completion of the notification, Previous_Record_Count is set to the value of Total_Record_Count that caused the notifications to occur.

Previous_Notification_Count is an internal variable, of type Unsigned32, which maintains the value of Total_Record_Count at which the most recent notification took place. Upon initialization it shall be set to the value of the Total_Record_Count property in the object referenced by Object_Property_Reference.

For the purposes of event notification, the BUFFER_READY event generates TO-NORMAL transitions.

[Add to **21**, the additional BACnet Base Types]

```
BACnetClientCOV ::= CHOICE {
        real-increment        REAL,
        default-increment     NULL
        }

BACnetLogRecord ::=  SEQUENCE {
        timestamp        [0] BACnetDateTime,
        logDatum         [1] CHOICE {
                                 log-status        [0] BACnetLogStatus,
                                 boolean-value     [1] BOOLEAN,
                                 real-value        [2] REAL,
                                 enum-value        [3] ENUMERATED,  -- Optionally limited to 32 bits
                                 unsigned-value    [4] Unsigned,    -- Optionally limited to 32 bits
                                 signed-value      [5] INTEGER,     -- Optionally limited to 32 bits
                                 bitstring-value   [6] BIT STRING,  -- Optionally limited to 32 bits
                                 null-value        [7] NULL,
                                 failure           [8] Error,
                                 time-change       [9] REAL,
                                 any-value         [10] ABSTRACT-SYNTAX.&Type  -- Optional
                                 }
        statusFlags      [2] BACnetStatusFlags OPTIONAL
        }
```

```
BACnetLogStatus ::= BIT STRING {
        log-disabled            (0),
        buffer-purged           (1)
        }
```

[Add to **21**, BACnetEventParameter, p. 371]

```
buffer-ready      [7] SEQUENCE {
                        notification-threshold          [0] Unsigned,
                        previous-notification-count     [1] Unsigned32
                        },
```

[Add to **21**, BACnetNotificationParameters, p. 373]

```
buffer-ready      [7] SEQUENCE {
                        buffer-device           [0] BACnetObjectIdentifier,
                        buffer-object           [1] BACnetObjectIdentifier,
                        previous-notification   [2] BACnetDateTime,
                        current-notification    [3] BACnetDateTime
                        },
```

[Add to **21**, BACnetEventType, p. 372]

```
buffer-ready      (7),
```

[Add to **21**, BACnetObjectType, p. 374]

```
trend-log         (20),
```

[Add to **21**, BACnetObjectTypesSupported p. 374]

```
trend-log         (20),
```

[Add to **21**, BACnetPropertyIdentifier (distributed alphabetically), p. 378]

```
buffer-size                       (126),
client-cov-increment              (127),
cov-resubscription-interval       (128),
current-notify-time               (129),
log-buffer                        (131),
log-device-object-property        (132),
log-enable                        (133),
log-interval                      (134),
notification-threshold            (137),
previous-notify-time              (138),
records-since-notification        (140),
record-count                      (141),
start-time                        (142),
stop-time                         (143),
stop-when-full                    (144),
total-record-count                (145),
```

[Add comments to same production]

```
-- see buffer-size                 (126),
-- see client-cov-increment        (127),
-- see cov-resubscription-interval (128),
-- see current-notify-time         (129),
-- see log-buffer                  (131),
-- see log-device-object-property  (132),
-- see log-enable                  (133),
-- see log-interval                (134),
-- see notification-threshold      (137),
-- see previous-notify-time        (138),
-- see records-since-notification  (140),
-- see record-count                (141),
-- see start-time                  (142),
-- see stop-time                   (143),
-- see stop-when-full              (144),
-- see total-record-count          (145),
…
```

[Add new Error Code to **18.3**, p. 314, by inserting in alphabetical order and renumbering other subclauses]

NOT_COV_PROPERTY – The property is not conveyed by COV notification.

[Add new Error Code to **18.6**, p. 315, by inserting in alphabetical order and renumbering other subclauses]

COV_SUBSCRIPTION_FAILED – COV Subscription failed for some reason.

[In **21**, p. 364, add new error-codes]

```
cov-subscription-failed   (43),
not-cov-property          (44),
```

[Add to **Annex C**]

```
TREND-LOG :: = SEQUENCE {
        object-identifier              [75] BACnetObjectIdentifier,
        object-name                    [77] CharacterString,
        object-type                    [79] BACnetObjectType,
        description                    [28] CharacterString OPTIONAL,
        log-enable                     [133] BOOLEAN,
        start-time                     [142] BACnetDateTime OPTIONAL,
        stop-time                      [143] BACnetDateTime OPTIONAL,
        log-device-object-property     [132] BACnetDeviceObjectPropertyReference OPTIONAL,
        log-interval                   [134] Unsigned OPTIONAL,
        cov-resubscription-interval    [128] Unsigned OPTIONAL,
        client-cov-increment           [127] BACnetClientCOV OPTIONAL,
        stop-when-full                 [144] BOOLEAN,
        buffer-size                    [126] Unsigned,
        log-buffer                     [131] SEQUENCE OF BACnetLogRecord,
        record-count                   [141] Unsigned,
        total-record-count             [145] Unsigned32,
        notification-threshold         [137] Unsigned OPTIONAL,
        records-since-notification     [140] Unsigned OPTIONAL,
        previous-notify-time           [138] BACnetDateTime OPTIONAL,
        current-notify-time            [129] BACnetDateTime OPTIONAL,
        event-state                 [36] BACnetEventState,
        notification-class          [17] Unsigned OPTIONAL,
        event-enable                [35] BACnetEventTransitionBits OPTIONAL,
        acked-transitions           [0] BACnetEventTransitionBits OPTIONAL,
        notify-type                 [72] BACnetNotifyType OPTIONAL,
        event-time-stamps           [130]    SEQUENCE OF BACnetTimeStamp OPTIONAL
                                    --accessed as a BACnetARRAY
        }
```

[Add to **Annex D**, 'X' to be determined]

D.X Example of a Trend Log Object

The following is an example of a Trend Log object that periodically logs data from an object in a remote device and which performs buffer-ready notification via intrinsic reporting.

Property:	Object_Identifier =	(Trend Log, Instance 1)
Property:	Object_Name =	"Room 3Log"
Property:	Object_Type =	TREND_LOG
Property:	Description =	"Room 3 Temperature"
Property:	Log_Enable =	TRUE
Property:	Log_DeviceObjectProperty =	((Device, Instance 100), Analog Input, Instance 3, Present_Value)
Property:	Log_Interval =	6,000
Property:	Stop_When_Full =	FALSE
Property:	Buffer_Size =	250
Property:	Log_Buffer =	(((23-MAR-1998,12:32:33.0), 72.0,(FALSE,FALSE,FALSE,FALSE)), (23-MAR-1998,12:34:32.0),72.1, (FALSE,FALSE,FALSE,FALSE)),…)
Property:	Record_Count =	250
Property:	Total_Record_Count =	131040
Property:	Notification_Threshold =	83
Property:	Records_Since_Notification =	30
Property:	Previous_Notify_Time =	(23-MAR-1998, 17:06:32.0)
Property:	Current_Notify_Time =	(23-MAR-1998, 19:52:34.0)
Property:	Event_State =	NORMAL
Property:	Notification_Class =	1
Property:	Event_Enable =	{FALSE, TRUE, TRUE}
Property:	Acked_Transitions =	{TRUE, TRUE, TRUE}
Property:	Notify_Type =	EVENT
Property:	Event_Time_Stamps=	((23-MAR-95, 18:50:21.2), (*-*-*,*:*:*.*), (23-MAR-95, 19:01:34.0))

135*b*-16. The UnconfirmedCOVNotification service is extended to allow notifications without prior subscription as a means of distributing globally important data to a potentially large number of recipients.

[Change **13.10**, p. 243**]**

13.10.1.2 Subscriber Process Identifier

This parameter, of type Unsigned, shall convey a numeric "handle" meaningful to the subscriber. This handle shall be used to match future re-subscriptions and cancellations from the subscriber with the COV context that exists within the COV-server device and with Confirmed or UnconfirmedCOVNotifications to identify the process within the COV-client that should receive them. *The value zero is reserved for unsubscribed COV notifications as described in 13.11.*

[Change **13.11,** p. 246]

13.11 UnconfirmedCOVNotification Service

The UnconfirmedCOVNotification Service ~~is used by a COV-server to issue notifications of changes that may have occurred to the properties of a particular object to COV-clients who have used the SubscribeCOV service to request such notification.~~ is used to notify subscribers about changes that may have occurred to the properties of a particular object, *or to distribute object properties of wide interest (such as outside air conditions) to many devices simultaneously without a subscription.* Subscriptions for COV notifications are made using the SubscribeCOV service (see 13.10). *For unsubscribed notifications, the algorithm for determining when to issue this service is a local matter and may be based on a change of value, periodic updating, or some other criteria.*

13.11.1.2 Subscriber Process Identifier

This parameter, of type Unsigned, shall convey a numeric "handle" meaningful to the subscriber. This handle shall be used to identify the process within the COVclient that should receive the notification. *The value of zero is reserved for unsubscribed COV.*

13.11.1.5 Time Remaining

This parameter, of type Unsigned, shall convey the remaining lifetime of the subscription in seconds. A value of zero shall indicate an indefinite lifetime, without automatic cancellation, *or an unsubscribed notification.*

135*b*-17. Add eight new BACnet engineering units.

[Add to **21**, BACnetEngineeringUnits production, p. 367 - 370]**]**

BACnetEngineeringUnits ::= ENUMERATED {
--Area
    square-inches   *(115),*
    square-centimeters   *(116),*

--Enthalpy
    btus-per-pound   *(117),*

--Length
    centimeters   *(118),*

--Mass Flow
    pounds-mass-per-second   *(119),*

--Temperature
    delta-degrees-Fahrenheit   *(120),*
    delta-degrees-Kelvin   *(121),*

--Volumetric Flow
    cubic-feet-per-second   *(142),*
    ...
        }
-- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
-- 256-65535 may be used by others subject to the procedures and constraints described
-- in Clause 23. The last enumeration used in this version is ~~141~~*142*. ~~Enumerations 115-121~~
~~-- have been removed.~~

NOTICE

INSTRUCTIONS FOR SUBMITTING A PROPOSED CHANGE TO THIS STANDARD UNDER CONTINUOUS MAINTENANCE

This standard is maintained under continuous maintenance procedures by a Standing Standard Project Committee (SSPC) for which the Standards Committee has established a documented program for regular publication of addenda or revisions, including procedures for timely, documented, consensus action on requests for change to any part of the standard. SSPC consideration will be given to proposed changes according to the following schedule:

Deadline for receipt of proposed changes	SSPC will consider proposed changes at next
February 20	ASHRAE Annual Meeting (normally June)

Proposed changes must be submitted to the Manager of Standards (MOS) in the latest published format available from the MOS. However, the MOS may accept proposed changes in an earlier published format, if the MOS concludes that the differences are immaterial to the proposed changes. If the MOS concludes that the current form must be utilized, the proposer may be given up to 20 additional days to resubmit the proposed changes in the current format.

Specific changes in text or values are required and must be substantiated. The Manager of Standards will return to the submitter any change proposals that do not meet these requirements. Supplemental background documents to support changes submitted may be included.

FORM FOR SUBMITTAL OF PROPOSED CHANGE TO ASHRAE STANDARD UNDER CONTINUOUS MAINTENANCE

(Please type)

1. Submitter: ____
(name—type)

Affiliation: ____

Address: ______________________ City: ____________ State: ______ Zip: ________

Telephone: ______________ Fax: ____________ E-Mail: ______________

I hereby grant the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) the non-exclusive royalty rights, including non-exclusive royalty rights in copyright, in my proposals and I understand that I acquire no rights in publication of this standard in which my proposal in this or other similar analogous form is used. I hereby attest that I have the authority and am empowered to grant this copyright release.

Author's Signature: ________________________________ Date: __________

NOTE: Use a separate form for each comment, completing each section (including Sections 1 and 2) to facilitate processing.

2. Number and Year of Standard:

3. Clause (i.e., Section), Subclause or Paragraph Number, and Page Number:

4. I Propose To: (check one)	[ ] Change to read as shown	[ ] Delete and substitute as shown
	[ ] Add new text as shown	[ ] Delete without substitution

(Indicate the proposed change by showing a strikeout line through material to be deleted and underlining material to be added. After showing the text to be changed, insert a horizontal line and state the purpose, reason, and substantiation for the proposed change. Use additional pages if necessary.)

5. Proposed Change:

6. Purpose, Reason, and Substantiation Statements:
(Be brief; provide abstracts of lengthy substantiation; full text should be enclosed for reference on request by project committee members.)

[ ] Check if additional pages are attached. Number of additional pages: ______

NOTE: Use separate form for each comment. Submittals (MS Word 7 preferred) may be attached to e-mail (preferable), submitted on diskettes, uploaded to ASHRAE's ftp site, or submitted in paper form by mail or fax to ASHRAE, Manager of Standards, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305.
E-mail: change.proposal@ashrae.org. Ftp server address: ftp.ashrae.org, directory: *change.proposal*. Fax: 404-321-5478

ELECTRONIC PREPARATION/SUBMISSION OF FORM FOR PROPOSING CHANGES

An electronic version of each change, which must comply with the instructions in the Notice and the Form, is the preferred form of submittal to ASHRAE Headquarters at the address shown below. The electronic format facilitates both paper-based and computer-based processing. Submittal in paper form is acceptable. The following instructions apply to change proposals submitted in electronic form.

Use the appropriate file format for your word processor and save the file in either Microsoft Word 7 (preferred) or higher or WordPerfect 5.1 for DOS format. Please save each change proposal file with a different name (example, prop001.doc, prop002.doc, etc., for Word files—prop001.wpm, prop002.wpm, etc., for WordPerfect files). If supplemental background documents to support changes submitted are included, it is preferred that they also be in electronic form as wordprocessed or scanned documents.

Electronic change proposals may be submitted either as files (MS Word 6 preferred) attached to an e-mail (uuencode preferred), files uploaded to an ftp site, or on 3.5” floppy disk. ASHRAE will accept the following as equivalent to the signature required on the change submittal form to convey non-exclusive copyright:

Files attached to e-mail:	Electronic signature on change submittal form (as a picture; *.tif, or *.wpg), or e-mail address.
Files on disk or uploaded to ftp site:	Electronic signature on change submittal form (as a picture; *.tif, or *.wpg), listing of the submitter’s e-mail address on the change submittal form, or a letter with submitter’s signature accompanying the disk or sent by facsimile (single letter may cover all of proponent’s proposed changes).

Submit e-mail, ftp file, or disks containing change proposal files to:
Manager of Standards
ASHRAE
1791 Tullie Circle, NE
Atlanta, GA 30329-2305
E-mail: *change.proposal@ashrae.org*
Ftp server address: *ftp.ashrae.org*, logon to anonymous ftp in directory: *change.proposal*.
(Alternatively, mail paper versions to ASHRAE address or Fax: 404-321-5478.)

The form and instructions for electronic submittal to ASHRAE’s ftp site or as attachments to e-mail may be obtained from the Standards section of ASHRAE’s Home Page, *http://www.ashrae.org*, or by contacting a Standards Secretary, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. Phone: 404-636-8400. Fax: 404-321-5478. Email: *standards.section@ashrae.org*.

POLICY STATEMENT DEFINING ASHRAE'S CONCERN FOR THE ENVIRONMENTAL IMPACT OF ITS ACTIVITIES

ASHRAE is concerned with the impact of its members' activities on both the indoor and outdoor environment. ASHRAE's members will strive to minimize any possible deleterious effects on the indoor and outdoor environment of the systems and components in their responsibility while maximizing the beneficial effects these systems provide, consistent with accepted standards and the practical state of the art.

ASHRAE's short-range goal is to ensure that the systems and components within its scope do not impact the indoor and outdoor environment to a greater extent than specified by the standards and guidelines as established by itself and other responsible bodies.

As an ongoing goal, ASHRAE will, though its Standard Committee and extensive technical committee structure, continue to generate up-to-date standards and guidelines where appropriate and adopt, recommend, and promote those new and revised standards developed by other responsible organizations.

Through its *Handbook*, appropriate chapters will contain up-to-date standards and design considerations as the material is systematically revised.

ASHRAE will take the lead with respect to dissemination of environmental information of its primary interest and will seek out and disseminate information from other responsible organizations that is pertinent, as guides to updating standards and guidelines.

The effects of the design and selection of equipment and systems will be considered within the scope of the system's intended use and expected misuse. The disposal of hazardous materials, if any, will also be considered.

ASHRAE's primary concern for environmental impact will be at the site where equipment within ASHRAE's scope operates. However, energy source selection and the possible environmental impact due to the energy source and energy transportation will be considered where possible. Recommendations concerning energy selection should be made by its members.