

BACnet Errata
ANSI/ASHRAE STANDARD 135-2004
A Data Communication Protocol for Building Automation and Control Networks

May 3, 2007

This document lists all known *errata* to ANSI/ASHRAE Standard 135-2004 as of the above date. Each entry is cited first by clause, then page number, except where an erratum covers more than one clause. The outside back cover marking identifying the first printing of Standard 135-2004 is “86443 PC 6/04” and “86443 PC 2/06” for the reprint. Items 17 through 21 have been added since the previously published errata sheet dated February 7, 2007 was distributed. Items 12 through 21 have been added since the previously published errata sheet dated December 1, 2005 was distributed and these items are the only errata applicable to the 2/06 reprint. Errata noted in the previous list dated 12/1/05 have been corrected in the 2/06 reprint.

- 1) 5.4.4.2, p.27-28: There are five text locations where the RequestTimer is stopped. But in the client state machine’s state SEGMENTED_REQUEST the RequestTimer is never running, only the SegmentTimer is active. The following changes fix this.

5.4.4.2 SEGMENTED_REQUEST

In the SEGMENTED_REQUEST state, the device waits for a BACnet-SegmentACK-PDU for one or more segments of a BACnet-Confirmed-Request-PDU.

...

SimpleACK_Received

If a BACnet-SimpleACK-PDU is received from the network layer and SentAllSegments is TRUE, then

stop ~~RequestTimer~~SegmentTimer; send CONF_SERV.confirm(+) to the local application program; and enter the IDLE state.

UnsegmentedComplexACK_Received

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is FALSE and SentAllSegments is TRUE,

then stop ~~RequestTimer~~SegmentTimer; send CONF_SERV.confirm(+) to the local application program; and enter the IDLE state.

SegmentedComplexACK_Received

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is zero and this device supports segmentation and SentAllSegments is TRUE,

then stop ~~RequestTimer~~SegmentTimer; compute ActualWindowSize based on the 'proposed-window-size' parameter of the received BACnet-ComplexACK-PDU and on local conditions; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = FALSE, and 'actual-window-size' = ActualWindowSize; start SegmentTimer; set LastSequenceNumber to zero; set InitialSequenceNumber to zero; and enter the SEGMENTED_CONF state to receive the remaining segments. (The method used to determine ActualWindowSize is a local matter, except that the value shall be less than or equal to the 'proposed-window-size' parameter of the received BACnet-ComplexACK-PDU and shall be in the range 1 to 127, inclusive.)

ErrorPDU_Received

If a BACnet-Error-PDU is received from the network layer and SentAllSegments is TRUE,

then stop ~~RequestTimer~~*SegmentTimer*; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

RejectPDU_Received

If a BACnet-Reject-PDU is received from the network layer and SentAllSegments is TRUE,

then stop ~~RequestTimer~~*SegmentTimer*; send REJECT.indication to the local application program; and enter the IDLE state. = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE; and enter the IDLE state.

- 2) Although the standard states that the MS/TP token is passed 50 times between Poll For Master cycles, the Master Node state machine actually passes the token 52 times. The following changes fix this.

Clause 9.5.2, page 79:

9.5.2 Variables

...

TokenCount The number of tokens received by this node. When this counter reaches the value N_{poll} , the node polls the address range between **TS** and **NS** for additional master nodes. **TokenCount** is set to ~~zero~~ *one* at the end of the polling process.

Clause 9.5.6.5, pp.89-90:

9.5.6.5 DONE_WITH_TOKEN

...

SoleMaster

If FrameCount is greater than or equal to $N_{max_info_frames}$ and TokenCount is less than $N_{poll} - 1$ and SoleMaster is TRUE,

then there are no other known master nodes to which the token may be sent (true master-slave operation). Set FrameCount to zero, increment TokenCount, and enter the USE_TOKEN state.

SendToken

If FrameCount is greater than or equal to $N_{max_info_frames}$ and TokenCount is less than $N_{poll} - 1$ and SoleMaster is FALSE, or if NS is equal to $(TS+1)$ modulo $(N_{max_master}+1)$,

then increment TokenCount; call SendFrame to transmit a Token frame to NS; set RetryCount and EventCount to zero; and enter the PASS_TOKEN state. (The comparison of NS and TS+1 eliminates the Poll For Master if there are no addresses between TS and NS, since there is no address at which a new master node may be found in that case).

SendMaintenancePFM

If FrameCount is greater than or equal to $N_{max_info_frames}$ and TokenCount is greater than or equal to $N_{poll} - 1$ and $(PS+1)$ modulo $(N_{max_master}+1)$ is not equal to NS,

then set PS to $(PS+1)$ modulo $(N_{max_master}+1)$; call SendFrame to transmit a Poll For Master frame to PS; set RetryCount to zero; and enter the POLL_FOR_MASTER state.

ResetMaintenancePFM

If FrameCount is greater than or equal to $N_{max_info_frames}$ and TokenCount is greater than or equal to $N_{poll} - 1$ and $(PS+1)$ modulo $(N_{max_master}+1)$ is equal to NS, and SoleMaster is FALSE,

then set PS to TS; call SendFrame to transmit a Token frame to NS; set RetryCount, **TokenCount**, and EventCount to zero; *set TokenCount to one*; and enter the PASS_TOKEN state.

SoleMasterRestartMaintenancePFM

If FrameCount is greater than or equal to $N_{\text{max_info_frames}}$, TokenCount is greater than or equal to $N_{\text{poll}} - 1$, $(PS+1)$ modulo $(N_{\text{max_master}}+1)$ is equal to NS, and SoleMaster is TRUE,

then set PS to $(NS + 1)$ modulo $(N_{\text{max_master}}+1)$; call SendFrame to transmit a Poll For Master to PS; set NS to TS (no known successor node); set RetryCount, ~~TokenCount~~, and EventCount to zero; *set TokenCount to one*; and enter the POLL_FOR_MASTER state to find a new successor to TS.

- 3) 9.5.6.3, p.88: Clause 9.5.6.3 has been misinterpreted as stating that frame types not expecting a response should all be sent before any frame types that do expect a response. For a router to MS/TP, this violates the ordering of frames specified in clause 6.5.4. The following correction clarifies this.

9.5.6.3 USE_TOKEN

In the USE_TOKEN state, the node is allowed to send one or more data frames. These may be BACnet Data frames or proprietary frames.

NothingToSend

If there is no data frame awaiting transmission,

then set FrameCount to $N_{\text{max_info_frames}}$ and enter the DONE_WITH_TOKEN state.

SendNoWait

If ~~there is a~~ *the next* frame awaiting transmission ~~that~~ is of type Test_Response, BACnet Data Not Expecting Reply, or a proprietary type that does not expect a reply,

then call SendFrame to transmit the frame; increment FrameCount; and enter the DONE_WITH_TOKEN state.

SendAndWait

If ~~there is a~~ *the next* frame awaiting transmission ~~that~~ is of type Test_Request, BACnet Data Expecting Reply, or a proprietary type that expects a reply,

then call SendFrame to transmit the data frame; increment FrameCount; and enter the WAIT_FOR_REPLY state.

- 4) 12.11.4, p.179: BACKUP_IN_PROGRESS was not added to the Device object's System_Status values list when Backup & Restore (Clause 19.1) was adopted.

12.11.4 System_Status

This property, of type BACnetDeviceStatus, reflects the current physical and logical status of the BACnet Device. The values that may be taken on by this property are

{ OPERATIONAL, OPERATIONAL_READ_ONLY, DOWNLOAD_REQUIRED, DOWNLOAD_IN_PROGRESS, ~~NON_OPERATIONAL~~, NON_OPERATIONAL, BACKUP_IN_PROGRESS }.

The exact meaning of these ~~states~~ *states*, except for BACKUP_IN_PROGRESS, in a given device and their synchronization with other internal operations of the device or the execution of BACnet services by the device are local matters and are not defined by this standard.

- 5) Figure 13-11, p. 267, shows a form of the Event Enrollment object that no longer exists. The figure should look like the following:

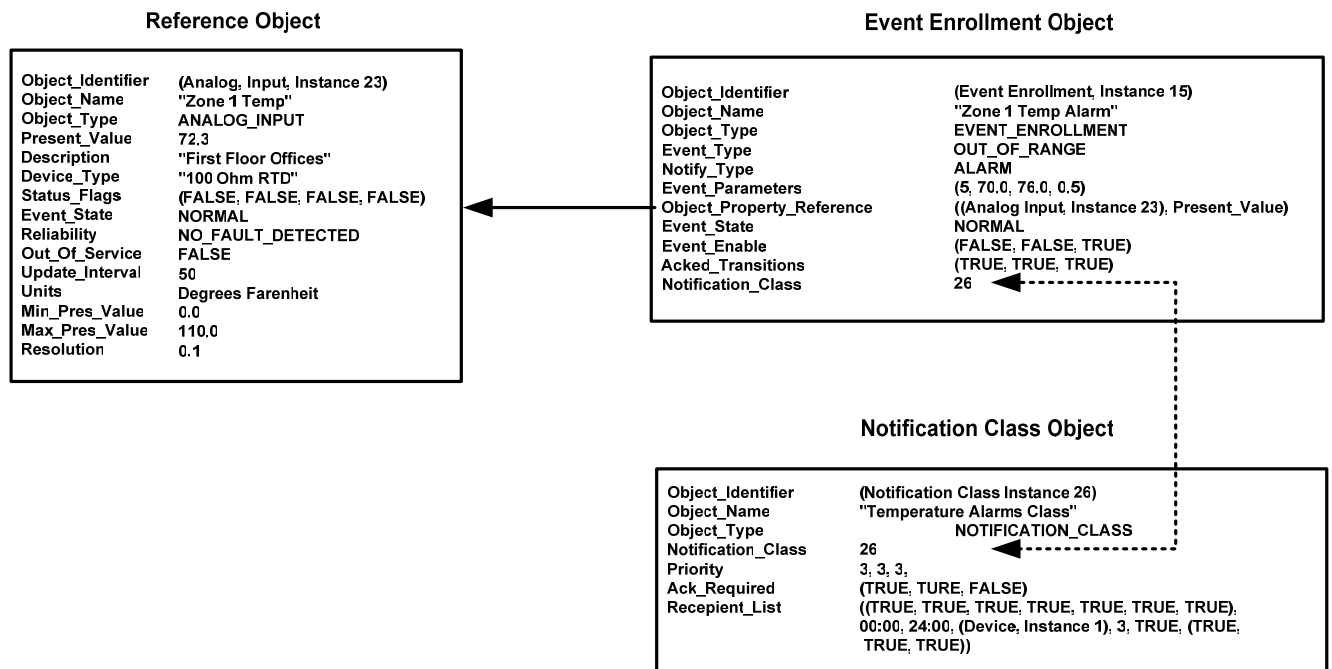


Figure 13-11. Example of an Event Enrollment.

- 6) 13.9.1.7, p. 278, refers to a property that was removed from the Event Enrollment object:

13.9.1.7 Priority

This parameter, of type Unsigned8, shall specify the priority of the event that has occurred. The priority is specified by the Priority property of the Notification Class ~~or Event Enrollment~~ objects associated with the event. The possible range of priorities is 0-255. A lower number indicates a higher priority. The priority and the Network Priority (see 6.2.2) are associated as defined in Table 13-5.

- 7) 19.2.1.1, p.363: incorrectly states that the Present_Value properties of the Analog Value, Binary Value and Multi-state Value objects are commandable by definition.

19.2.1.1 Commandable Properties

...

The designated ~~property~~ *properties* of ~~these~~ *the Analog Output, Binary Output and Multi-state Output* objects ~~is~~ *are* commandable (prioritized) by definition. *The designated properties of the Analog Value, Binary Value and Multi-state Value objects may optionally be commandable.* Individual vendors, however, may decide to apply prioritization to any of the vendor-specified properties. These additional commandable properties shall have associated Priority_Array and Relinquish_Default properties with appropriate names. See 23.3

- 8) 19.2.1.1, p.363: Multi-State Value is not correctly capitalized.

BACnetCOVSubscription, p.409, several incorrect leading uppercase characters:

```

BACnetCOVSubscription ::= SEQUENCE {
  Rrecipient                [0] BACnetRecipientProcess,
  MmonitoredPropertyReference [1] BACnetObjectPropertyReference,
  IissueConfirmedNotifications [2] BOOLEAN,
  TtimeRemaining            [3] Unsigned,
  COVcovIncrement           [4] REAL OPTIONAL -- used only with monitored
                                          -- properties with a datatype of REAL
}

```

BACnetDateRange, p.409, incorrect leading uppercase character on startDate:

```

SstartDate    Date,

```

BACnetDeviceObjectPropertyValue, p.410, incorrect hyphen on arrayIndex:

```

arrayIndex    [3] Unsigned OPTIONAL,

```

BACnetEventParameter, p.416, extended choice, incorrectly written double and reference parameters:

```

extended[9] SEQUENCE {
  vendorId                [0] Unsigned,
  extendedEventType       [1] Unsigned,
  parameters              [2] SEQUENCE OF CHOICE {
    null    NULL,
    real    REAL,
    integer Unsigned,
    boolean BOOLEAN,
    double  DOUBLE, Double,
    octet   OCTET STRING,
    bitstring BIT STRING,
    enum    ENUMERATED,
    reference [0] BACnetDeviceObjectPropertyReference
  }
}

```

BACnetEventType, p.417, nested comment (to be removed):

```

-- complex-event-type(6), see comment below

```

BACnetLogRecord, p.419, missing comma after logDatum:

```

logDatum    [1] CHOICE {
  log-status [0] BACnetLogStatus,
  ...
  any-value  [10] ABSTRACT-SYNTAX.&Type -- Optional
}
statusFlags [2] BACnetStatusFlags OPTIONAL

```

BACnetNotificationParameters, p.420, extended choice, incorrectly written double parameter:

```

extended [9] SEQUENCE {
  ...
  parameters [2] SEQUENCE OF CHOICE {
    ...
    boolean BOOLEAN,
    double DOUBLE, Double,
  }

```

BACnetPropertyValue, p.429, incorrect leading uppercase character:

```

BACnetPropertyValue ::= SEQUENCE {
  PpropertyIdentifier [0] BACnetPropertyIdentifier,

```

14) Annex C, p.453 and p.463: The event-time-stamps property of the Accumulator and Pulse Converter objects is missing the comment:

-- accessed as a BACnetARRAY

15) In Clause 12.4.9, Out-Of-Service should be Out_Of_Service.

16) The Standard defines and lists the Event Parameters in terms of the BACnet defined EventType and context tag declarations. EXTENDED and UNSIGNED_RANGE are missing in the prose in one place, and in two tables of Event Parameters in Clause 12.

Add UNSIGNED_RANGE to the enumerations in Clause 12.12.5 (Event Enrollment), p.185:

12.12.5 Event_Type

This property, of type BACnetEventType, indicates the type of event algorithm that is to be used to detect the occurrence of events and report to enrolled devices. 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, CHANGE_OF_LIFE_SAFETY, EXTENDED, UNSIGNED_RANGE}.

Add the following entries to **Table 12-15**, p. 186:

...		
EXTENDED	Any BACnetEventState	Vendor_Id Extended_Event_Type Parameters
UNSIGNED_RANGE	NORMAL HIGH_LIMIT LOW_LIMIT	Time_Delay Low_Limit High_Limit

Change the unnumbered table in **Clause 12.12.7**, p. 187:

Deadband,
High_Diff_Limit,
Low_Diff_Limit,
High_Limit (REAL),

These parameters, of type REAL, apply to the FLOATING_LIMIT and OUT_OF_RANGE event algorithms. Their use is described in the algorithms for these types in Clause 13.

Low_Limit (REAL)

Mode_Property_Reference

This parameter, of type BACnetDeviceObjectPropertyReference, applies to the CHANGE_OF_LIFE_SAFETY algorithm. It identifies the object and property that provides the operating mode of the referenced object providing life safety functionality (normally the Mode property). This parameter may reference only object properties that are of type BACnetLifeSafetyMode.

Vendor_Id

This parameter, of type Unsigned16, is a vendor identification code, assigned by ASHRAE, which is used to distinguish proprietary extensions to the protocol. See clause 23.

Extended_Event_Type

This parameter, of type Unsigned, is a value selected by the organization indicated by Vendor_Id, which specifies the interpretation of the Parameters parameter.

Parameters

This parameter consists of a set of data values whose interpretation is specified by the combination of Vendor_Id and Extended_Event_Type. The set of data values constitutes the set of input parameters for the proprietary algorithm.

High_Limit (Unsigned),

Low_Limit (Unsigned)

These parameters, of type Unsigned, apply to the UNSIGNED_RANGE event algorithm. Their use is described in the algorithm for these types in Clause 13.

Change the punctuation as follows in **Clause 21**, p. 415:

```

BACnetEventParameter ::= CHOICE {
    ...
    extended    [9] SEQUENCE {
        vendorId vendor-id                [0] Unsigned16,
        extendedEventType extended-event-type [1] Unsigned,
        parameters [2] SEQUENCE OF CHOICE {

```

```

BACnetNotificationParameters ::= CHOICE {
    ...
    extended    [9] SEQUENCE {
        vendorId vendor-id                [0] Unsigned16,
        extendedEventType extended-event-type [1] Unsigned,
        parameters [2] SEQUENCE OF CHOICE {

```

16) Although BACnet/IP (as defined in Annex J) is a very popular data link option, it not consistently mentioned in other parts of the standard that discuss data link layers.

Change Clause 4.1, p. 9:

BACnet is based on a four-layer collapsed architecture that corresponds to the physical, data link, network, and application layers of the OSI model as shown in Figure 4-2. The application layer and a simple network layer are defined in the BACnet standard. BACnet provides ~~five~~ *six* options that correspond to the OSI data link and physical layers. Option 1 is the logical link control (LLC) protocol defined by ISO 8802-2 Type 1, combined with the ISO 8802-3 medium access control (MAC) and physical layer protocol. ISO 8802-2 Type 1 provides unacknowledged connectionless service only. ISO 8802-3 is the international standard version of the well-known "Ethernet" protocol. Option 2 is the ISO 8802-2 Type 1 protocol combined with ARCNET (ATA/ANSI 878.1). Option 3 is a Master-Slave/Token-Passing (MS/TP) protocol designed specifically for building automation and control devices as part of the BACnet standard. The MS/TP protocol provides an interface to the network layer that looks like the ISO 8802-2 Type 1 protocol and controls access to an EIA-485 physical layer. Option 4, the Point-To-Point protocol, provides mechanisms for hardwired or dial-up serial, asynchronous communication. Option 5 is the LonTalk protocol. *Option 6, BACnet/IP, permits BACnet devices to use standard Internet Protocols (UDP and IP) as a virtual data link layer.* Collectively these options

provide a master/slave MAC, deterministic token-passing MAC, high-speed contention MAC, dial-up access, star and bus topologies, and a choice of twisted-pair, coax, or fiber optic media. The details of these options are described in Clauses 7 through 11 *and Annex J*.

Change **Figure 4-2**, p. 9:

BACnet Layers					Equivalent OSI Layers	
BACnet Application Layer					Application	
BACnet Network Layer					Network	
ISO 8802-2 (IEEE 8802.3) Type 1		MS/TP	PTP	LonTalk	BVLL	
ISO 8802-3 (IEEE 802.3)	ARCNET	EIA-485	EIA-232		UDP/IP	
					Data Link	
					Physical	

Change Table **6-2**, p. 51:

Table 6-2. BACnet DADR and SADR encoding rules based upon data link layer technology

BACnet Data Link Layer	DLEN	SLEN	Encoding Rules
ISO 8802-3 ("Ethernet"), as defined in Clause 7	6	6	Encoded as in their MAC layer representations
ARCNET, as defined in Clause 8	1	1	Encoded as in their MAC layer representations
MS/TP, as defined in Clause 9	1	1	Encoded as in their MAC layer representations
LonTalk domain wide broadcast	2	2	The encoding for the SADR is shown in Figure 6-3
LonTalk multicast	2	2	
LonTalk unicast	2	2	The encoding for the DADR is shown in Figure 6-4
LonTalk, unique Neuron_ID	7	2	
<i>BACnet/IP, as defined in Annex J</i>	6	6	<i>Encoded as specified in J.1.2</i>

17) Table 12-15, p.186 (Event Enrollment object type).The description of the Change of Value event type in Clause 13.3.3 **CHANGE_OF_VALUE Algorithm**, p.260, notes only a single state, Normal, and only Normal-Normal event transitions. Table 12-15 incorrectly lists an Event_State of OFFNORMAL. Remove the reference as shown.

Table 12-15. Event_Types, Event_States, and their Parameters

Event_Type	Event_State	Event_Parameters
CHANGE_OF_BITSTRING	NORMAL OFFNORMAL	Time_Delay Bitmask List_Of_Bitstring_Values
CHANGE_OF_STATE	NORMAL OFFNORMAL	Time_Delay List_Of_Values
CHANGE_OF_VALUE	NORMAL OFFNORMAL	Time_Delay Bitmask Referenced_Property_Increment
COMMAND_FAILURE	NORMAL OFFNORMAL	Time_Delay Feedback_Property_Reference
FLOATING_LIMIT	NORMAL HIGH_LIMIT LOW_LIMIT	Time_Delay Setpoint_Reference Low_Diff_Limit High_Diff_Limit Deadband
OUT_OF_RANGE	NORMAL HIGH_LIMIT LOW_LIMIT	Time_Delay Low_Limit High_Limit Deadband
BUFFER_READY	NORMAL	Notification_Threshold
CHANGE_OF_LIFE_SAFETY	NORMAL OFFNORMAL LIFE_SAFETY_ALARM	Time_Delay List_Of_Alarm_Values List_Of_Life_Safety_Alarm_Values Mode_Property_Reference

18) 12.17.28, p.211. The Loop object's Priority_For_Writing property description identifies the wrong property for the reference to the controlled object property. Correct the clause as shown.

12.17.28 Priority_For_Writing

Loop objects may be used to control the commandable property of an object. This property, of type Unsigned, provides a priority to be used by the command prioritization mechanism. It identifies the particular priority slot in the Priority_Array of the ~~Controlled_Variable_Reference~~ *Manipulated_Variable_Reference* that is controlled by this loop. It shall have a value in the range 1-16.

19) 13.4, p. 266 and Figure 13-11, p.267. The Recipient and associated properties were removed from the Event Enrollment object in 2003, but some references in Clause 13 were not also removed. Remove the references as shown.

13.4 Alarm and Event Occurrence and Notification

...

Intrinsic object-generated events, and events generated by Event Enrollment objects, may be controlled by a Notification Class object that defines their handling options. ~~Event Enrollment objects, may alternatively specify single recipients to receive notifications without special handling.~~

...

Event Enrollment objects and Notification Class objects specify the destination devices for notification messages using BACnetRecipients. The recipients may be individual devices, groups of devices with a common multicast address, or all devices reachable by a broadcast address. If a broadcast is used, the scope may be limited to all devices on a single network or it may be extended to encompass all devices on a BACnet internetwork. See Clause 6.

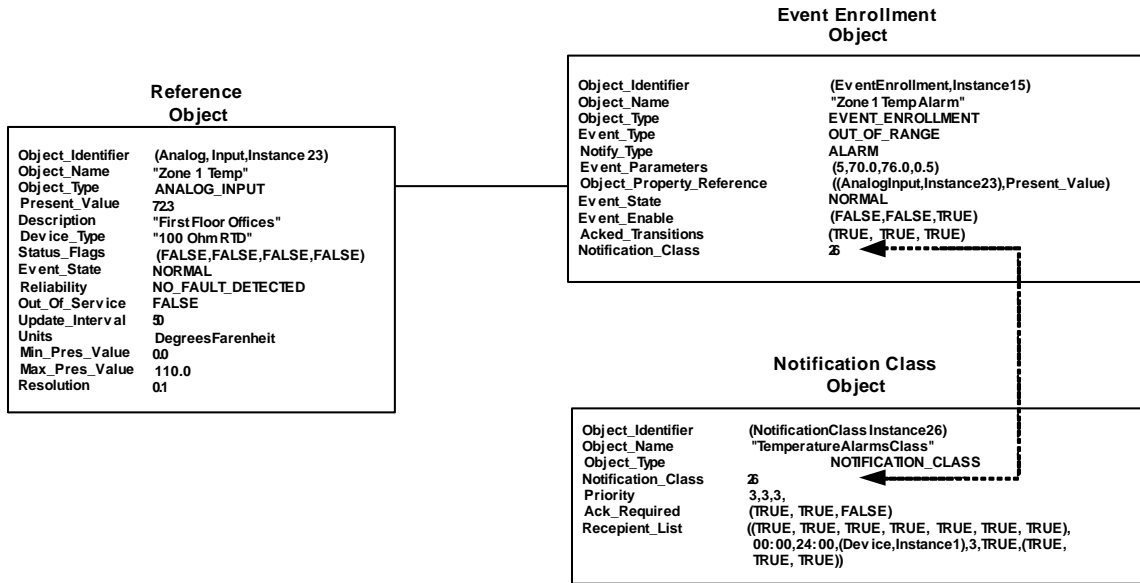


Figure 13-11. Example of an Event Enrollment.

20) 18.8.1, p. 357 and 18.9.1, p.358. Clause 18.8.1 describes BUFFER_OVERFLOW in terms of an input buffer, but in Clause 5.4.5.3 uses the term BUFFER_OVERFLOW for both input and output overflows. Remove the explicit "input" references.

18.8.1 BUFFER_OVERFLOW:- An input buffer capacity has been exceeded.

18.9.1 BUFFER_OVERFLOW:- An input buffer capacity has been exceeded.

21) Clause 12.12.7, p.187. A table describes the fields in the BACnetEventParameter construct. In two of the entries, the fields are described as type BACnetObjectPropertyReference but should be BACnetDeviceObjectPropertyReference, per Addendum 135-1995b-6. Change table entries as shown.

Feedback_Property_Reference This parameter, of type BACnetDeviceObjectPropertyReference, applies to the COMMAND_FAILURE algorithm. It identifies the object and property that provides the feedback to ensure that the commanded property has changed value. This property may reference only object properties that have enumerated values or are of type BOOLEAN.

Setpoint_Reference This parameter, of type BACnetDeviceObjectPropertyReference, applies to the FLOATING_LIMIT event algorithm. It indicates the setpoint reference for the reference property interval.