ANSI/ASHRAE Addendum z to
ANSI/ASHRAE Standard 135-2008

# ASHRAE ADDENDA

# BACnet®—A Data Communication Protocol for Building Automation and Control Networks

Approved by the ASHRAE Standards Committee on June 26, 2010; by the ASHRAE Board of Directors on June 30, 2010; and by the American National Standards Institute on July 1, 2010.

This addendum was approved 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 may be obtained in electronic form from the ASHRAE Web site (www.ashrae.org) or in paper form from the Manager of Standards.

The latest edition of an ASHRAE Standard may be purchased on the ASHRAE Web site (www.ashrae.org) or 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 US and Canada). For reprint permission, go to www.ashrae.org/permissions.



ISSN 1041-2336

Approved American National Standard
ANSI
www.ansi.org

**American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.**
1791 Tullie Circle NE, Atlanta, GA 30329
www.ashrae.org

**ASHRAE Standing Standard Project Committee 135**
**Cognizant TC: TC 1.4, Control Theory and Application**
**SPLS Liaison: Douglas T. Reindl**



## 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 Assistant Director of Technology for Standards and Special Projects 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, or
d. permission to reprint portions of the Standard.

**[This foreword and the "rationales" on the following pages are not part of this standard. They are merely informative and do not contain requirements necessary for conformance to the standard.]**

## FOREWORD

Addendum 135*z* to ANSI/ASHRAE Standard 135-2008 contains a number of changes to the current standard. These modifications are the result of change proposals made pursuant to the ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135. The changes are summarized below.



In the following document, language added to existing clauses of ANSI/ASHRAE 135-2008 and addenda is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are added, plain type is used throughout.

**135-2008z-1 Add Event_Message_Texts**

Rationale

Clause 13.8.1.9, which concerns "Message Text" for ConfirmedEventNotification Service, states:

> "This optional parameter, of type CharacterString, shall convey a string of printable characters. This parameter may be used to convey a message to be logged or displayed, which pertains to the occurrence of the event. The content of the message is a local matter."

The message text is the only user-defined text string in BACnet that is not accessible by a BACnet property. This text is available to a client only by means of a confirmed or unconfirmed event notification message. Clients updating their alarm- and event-lists by means of one of the services (GetAlarmSummary, GetEnrollmentSummary, GetEventInformation) have no chance to receive the Message Text.

A new optional standard property is defined for this purpose. This property shall be defined for all object types supporting intrinsic reporting and for the Event Enrollment object as well.

The property shall allow for distinct message texts for TO-NORMAL, TO-OFFNORMAL and TO-FAULT transitions, respectively. The property shall be of type BACnet ARRAY[3] of CharacterString.

[Change all property tables for object types that may implement intrinsic or algorithmic reporting in Clause 12, and in all published addenda. Make the same modification to all tables, adjusting as needed for footnote numbers.]

| Property Identifier | Property Datatype | Conformance Code |
|---|---|---|
| Object_Identifier | BACnetObjectIdentifier | R |
| Object_Name | CharacterString | R |
| Object_Type | BACnetObjectType | R |
| … | … | … |
| Notify_Type | BACnetNotifyType | O[n] |
| Event_Time_Stamps | BACnetARRAY[3] of BACnetTimeStamp | O[n] |
| *Event_Message_Texts* | *BACnetARRAY[3]of CharacterString* | *O[n]* |
| …. | …. | …. |
| Profile_Name | CharacterString | O |

[n] *This property, if present, is required to be read only.*

[Add Clause 12.n.x to all object definition clauses for object types that may implement algorithmic or intrinsic reporting in **Clause 12** and in all published addenda.]

***12.n.x Event_Message_Texts***

*This optional property, of type BACnetARRAY[3]of CharacterString, shall convey the Message Text of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events, respectively. This property shall be read only. If no notification of a transition has occurred yet, an empty string shall be stored in the respective array element.*

[Change **Clause 13.8.1.9**, p. 312]
[ConfirmedEventNotification service parameter]

**13.8.1.9 Message_Text**
This optional parameter, of type CharacterString, shall convey a string of printable characters. This parameter may be used to convey a message to be logged or displayed, which pertains to the occurrence of the event. The content of the message is a local matter. *If the optional property Event_Message_Texts is present in the event generating object, the text conveyed in this Message_Text parameter shall be stored in the respective field of the Event_Message_Texts array.*

[Change **Clause 13.9.1.9**, p. 315]
[UnconfirmedEventNotification service parameter]

**13.9.1.9 Message_Text**
This optional parameter, of type CharacterString, shall convey a string of printable characters. This parameter may be used to convey a message to be logged or displayed, which pertains to the occurrence of the event. The content of the message is a local matter. *If the optional property Event_Message_Texts is present in the event generating object, the text conveyed in this Message-Texts parameter shall be stored in the respective field of the Event_Message_Texts array.*

[Change **Clause 21**, p. 465]

**BACnetPropertyIdentifier** ::= ENUMERATED {
...
event-enable (35),
*event-message-texts (351),*
event-state (36),
...
-- *see event-message-texts* *(351),*
...
}

[Add to object type encodings for all object types that may implement intrinsic or algorithmic reporting in ANNEX C and in all published addenda.]

...
*event-message-texts [351] SEQUENCE SIZE(3) OF CharacterString OPTIONAL,*
*-- accessed as a BACnetARRAY*
...

**135-2008*z*-2 Add UnconfirmedEventNotification to Automated Trend Retrieval BIBBs**

Rationale

The BIBBs for Automated Trend Retrieval mandate the use of the ConfirmedEventNotification services for sending the BUFFER_READY Event Notifications. However, there is no place in the Clause 12 Trend Log object definition, or in the Clause 13 language for intrinsic alarming, where this is mandated. In addition, other BIBBs which specify alarming behavior specify both Confirmed and Unconfirmed Event Notifications. For example, AE-N-A and AE-N-I-B, and AE-LS-A/B.

This change aims to bring the T-ATR-A/B BIBBs into line with that approach. It requires support for initiating both ConfirmedEventNotification and UnconfirmedEventNotification to claim T-ATR-B, and requires support for executing both notification types to claim T-ATR-A.

[Change **Clause K.4.4**, p. 635]

**K.4.4 BIBB - Trending-Automated Trend Retrieval-A (T-ATR-A)**

The A device responds to a notification that a trend log is ready with new data and acquires the new data from the log.

| BACnet Service | Initiate | Execute |
|---|---|---|
| ConfirmedEventNotification | | x |
| *UnconfirmedEventNotification* | | *x* |
| ReadRange | x | |

Devices claiming conformance to T-ATR-A must be able to process BUFFER_READY event notifications generated by Trend Log objects and Event Enrollment objects.

[Change **Clause K.4.5**, p. 635]

**K.4.5 BIBB - Trending-Automated Trend Retrieval-B (T-ATR-B)**

The B device notifies the A device that a trending buffer has acquired a predetermined number of data samples using the BUFFER_READY event algorithm either intrinsically in the Trend Log object or algorithmically using an Event Enrollment object.

| BACnet Service | Initiate | Execute |
|---|---|---|
| ConfirmedEventNotification | x | |
| *UnconfirmedEventNotification* | *x* | |
| ReadRange | | x |

Devices claiming conformance to T-ATR-B must support the Trend Log object.

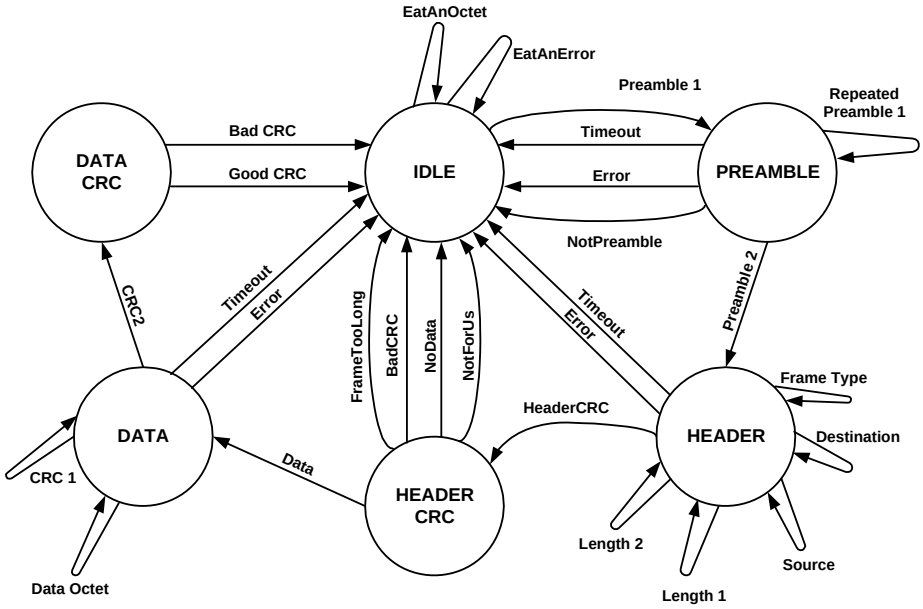
**135-2008z-3 Modify MS/TP State Machine to Ignore Data Not For Us.**

Rationale

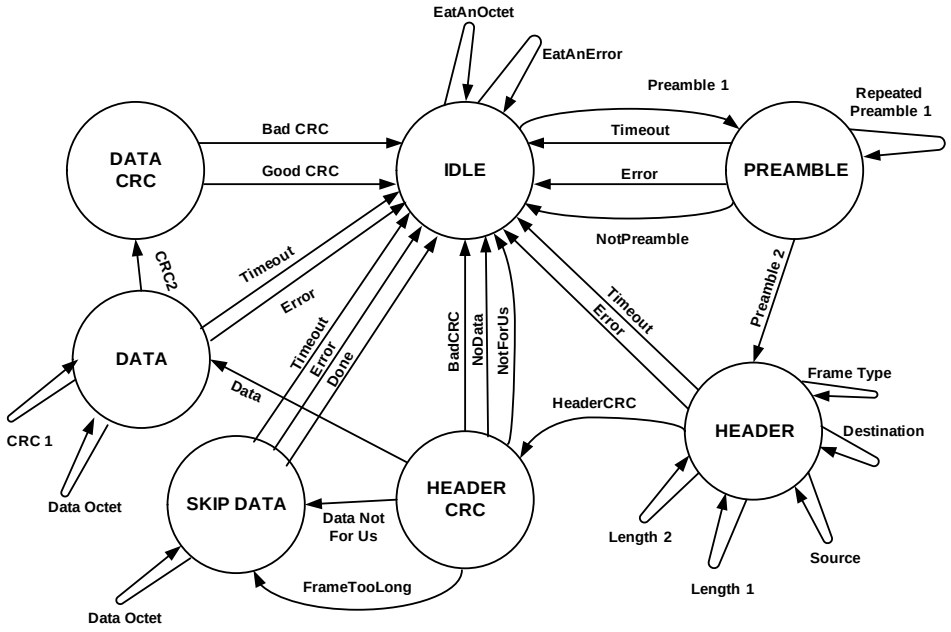
The MS/TP Receive Node State Machine changes to the IDLE state after the HeaderCRC is checked even if there is Data in the Frame to be processed if the DestinationAddress is not for us, or if the frame is too long. This causes the Data portion of the discarded frame to be parsed in the IDLE state, and in the case where the data or data coupled with the padding causes a Preamble to be seen, the state machine misses the beginning of the next valid frame.

**[**Change **Figure 9-3,** page 81]
[Make new state SKIP DATA state, as shown.]

[current figure]



[revised figure]




[Change **Clause 9.5.4.4**, page 83]

**9.5.4.4 HEADER_CRC**

In the HEADER_CRC state, the node validates the CRC on the fixed message header.

BadCRC

If the value of HeaderCRC is not X '55',

then set ReceivedInvalidFrame to TRUE to indicate that an error has occurred during the reception of a frame, and enter the IDLE state to wait for the start of the next frame.

NotForUs

If the value of the HeaderCRC is X '55' *and DataLength is zero* and the value of DestinationAddress is not equal to either TS (this station) or 255 (broadcast),

then enter the IDLE state to wait for the start of the next frame.

*DataNotForUs*

*If the value of the HeaderCRC is X '55' and DataLength is not zero and the value of DestinationAddress is not equal to either TS (this station) or 255 (broadcast),*

*then set Index to zero and enter the SKIP_DATA state to consume the data and data CRC portions of the frame.*

FrameTooLong

If the value of the HeaderCRC is X '55' and the value of DestinationAddress is equal to either TS (this station) or 255 (broadcast) and DataLength is greater than InputBufferSize,

then set ReceivedInvalidFrame to TRUE to indicate that a frame with an illegal or unacceptable data length has been received, ~~and enter the IDLE state to wait for the start of the next frame~~ *set Index to zero, and enter the SKIP_DATA state to consume the data and data CRC portions of the frame*.

NoData

If the value of HeaderCRC is X '55' and the value of DestinationAddress is equal to either TS (this station) or 255 (broadcast) and DataLength is zero,

then set ReceivedValidFrame to TRUE to indicate that a frame with no data has been received, and enter the IDLE state to wait for the start of the next frame.

Data

If the value of HeaderCRC is X'55' and the value of DestinationAddress is equal to either TS (this station) or 255 (broadcast) and DataLength is not zero and DataLength is less than or equal to InputBufferSize,

then set Index to zero; set DataCRC to X'FFFF'; and enter the DATA state to receive the data portion of the frame.

[Add new **Clause 9.5.4.7**, page 85]

**9.5.4.7 SKIP_DATA**

In the SKIP_DATA state, the node waits for the data portion of a frame to be received so that its contents can be ignored.

Timeout

If SilenceTimer is greater than $\mathbf{T_{frame\_abort}}$,

then set ReceivedInvalidFrame to TRUE to indicate that an error has occurred during the reception of a frame, and enter the IDLE state to wait for the start of the next frame.

Error

If ReceiveError is TRUE,

then set ReceiveError to FALSE; set SilenceTimer to zero; set ReceivedInvalidFrame to TRUE to indicate that an error has occurred during the reception of a frame; and enter the IDLE state to wait for the start of the next frame.

DataOctet

If ReceiveError is FALSE and DataAvailable is TRUE and Index is less than DataLength+1,

then set DataAvailable to FALSE; set SilenceTimer to zero; increment Index by 1; and enter the SKIP_DATA state.

Done

If ReceiveError is FALSE and DataAvailable is TRUE and Index is equal to DataLength+1,

then set DataAvailable to FALSE; set SilenceTimer to zero; and enter the IDLE state to wait for the start of the next frame.

[Change **Clause 9.5.2**, page 78]

**9.5.2 Variables**

A number of variables and timers are used in the descriptions that follow:

| | |
|---|---|
| **DataCRC** | Used to accumulate the CRC on the data field of a frame. |
| **DataLength** | Used to store the data length of a received frame. |
| **DestinationAddress** | Used to store the destination address of a received frame. |
| **EventCount** | Used to count the number of received octets or errors. This is used in the detection of link activity. |
| **FrameType** | Used to store the frame type of a received frame. |

**FrameCount** The number of frames sent by this node during a single token hold. When this counter reaches the value $\mathbf{N_{max\_info\_frames}}$, the node must pass the token.

**HeaderCRC** Used to accumulate the CRC on the header of a frame.

**Index** Used as an index by the Receive State Machine, up to ~~a maximum~~ *the* value of *DataLength+1*~~InputBufferSize~~.

**InputBuffer[]** An array of octets, used to store octets as they are received. InputBuffer is indexed from 0 to InputBufferSize-1. The maximum size of a frame is 501 octets. A smaller value for InputBufferSize may be used by some implementations.

. . .

**135-2008*z*-4 Add New Engineering Units**

Rationale

New engineering units are needed for new areas and markets where BACnet is being used.

[Change **Clause 21**, BACnetEngineeringUnits production, pp. 451-456]

```
BACnetEngineeringUnits ::= {
        …
    --Electrical
        ...
        decibels                    (199),
        decibels-millivolt          (200),
        decibels-volt               (201),
        ...
        millisiemens                (202),
        ...

    --Energy
        ...
        watt-hours-reactive         (203),
        kilowatt-hours-reactive     (204),
        megawatt-hours-reactive     (205),
        ...

    --Length
        kilometers                  (193),
        micrometers                 (194),
        ...

    --Mass
        ...
        grams                       (195),
        milligrams                  (196),
        ...

    --Volume
        ...
        milliliters                 (197),
        ...

    --Volumetric Flow
        ...
        milliliters-per-second      (198),
        …

    --Pressure
        ...
        millimeters-of-water        (206),
        ...

    --Other
        ...
        per-mille                   (207),
        grams-per-gram              (208),
        kilograms-per-kilogram      (209),
        grams-per-kilogram          (210),
```

*milligrams-per-gram* *(211),*
*milligrams-per-kilogram* *(212),*
*grams-per-milliliter* *(213),*
*grams-per-liter* *(214),*
*milligrams-per-liter* *(215),*
*micrograms-per-liter* *(216),*
*grams-per-cubic-meter* *(217),*
*milligrams-per-cubic-meter* *(218),*
*micrograms-per-cubic-meter* *(219),*
*nanograms-per-cubic-meter* *(220),*
*grams-per-cubic-centimeter* *(221),*
*becquerels* *(222),*
*kilobecquerels* *(223),*
*megabecquerels* *(224),*
*gray* *(225),*
*milligray* *(226),*
*microgray* *(227),*
*sieverts* *(228),*
*millisieverts* *(229),*
*microsieverts* *(230),*
*microsieverts-per-hour* *(231),*
*decibels-a* *(232),*
*nephelometric-turbidity-unit* *(233),*
*pH* *(234),*
*grams-per-square-meter* *(235),*
*minutes-per-degree-kelvin* *(236),*
...
}

-- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
-- 255-65535 may be used by others subject to the procedures and constraints described
-- in Clause 23. The last enumeration used in this version is ~~189~~*236*.

**135-2008*z*-5 Add Duplicate Segment Detection**

Rationale

A problem can start when a segment of a response is lost by the network that connects two devices. With an agreed-upon window size of 2, if the segment #2 in the response is dropped, the server times out and resends response segment 1 and 2. The client NAKs segment 1 (because it already has it), and then receives segment 2, which it ACKs (end of window). The server receives the NAK indicating segment 1, so it shifts its window and sends segments 2 and 3. The client receives the second segment #2 and NAKs it (because it already has it) and receives segment #3. The server receives the NAK indicating segment 2, so it shifts its window and sends segments 3 and 4. The client receives the second segment #3 and NAKs it and receives segment 4 and ACKs it (end of window).This process repeats itself until the whole message is finally received, and then the client tops off the sequence with an Abort-PDU because the server sent the final segment twice (as per the sequence outlined above) and the client's TSM had been completed and freed when the second final segment is received.

The change adds a new transition into each of the client and server state machines. The new state checks for the receipt of duplicate segments in the current window and discards them. The existing SegmentReceivedOutOfOrder transition is modified so that these segments would not cause that transition to occur.

[Change **Clause 5.4.1**, p 24 ]

**5.4.1 Variables And Parameters**

The following variables are defined for each instance of Transaction State Machine:

| | |
|---|---|
| **RetryCount** | used to count APDU retries |
| **SegmentRetryCount** | used to count segment retries |
| ***DuplicateCount*** | *used to count duplicate segments* |
| **SentAllSegments** | used to control APDU retries and the acceptance of server replies |
| **LastSequenceNumber** | stores the sequence number of the last segment received in order |
| **InitialSequenceNumber** | stores the sequence number of the first segment of a sequence of segments that fill a window |
| **ActualWindowSize** | stores the current window size |
| **ProposedWindowSize** | stores the window size proposed by the segment sender |
| **SegmentTimer** | used to perform timeout on PDU segments |
| **RequestTimer** | used to perform timeout on Confirmed Requests |

The following parameters are used in the description:

| | |
|---|---|
| $\mathbf{T}_{\mathbf{seg}}$ | This parameter is the length of time a node shall wait for a SegmentACK-PDU after sending the final segment of a sequence. Its value is the value of the APDU_Segment_Timeout property of the node's Device object. |
| $\mathbf{T}_{\mathbf{wait\_for\_seg}}$ | This parameter is the length of time a node shall wait after sending a SegmentACK-PDU for an additional segment of the message. Its value is equal to four times the value of the APDU_Segment_Timeout property of the node's Device object. |
| $\mathbf{T}_{\mathbf{out}}$ | This parameter represents the value of the APDU_Timeout property of the node's Device object. |

$\mathbf{N_{retry}}$ This parameter represents the value of the Number_Of_APDU_Retries property of the node's Device object.

$\boldsymbol{N_{dup}}$ *This parameter represents the number of duplicates that will be silently dropped per window before a negative segment ack is returned. This parameter shall be equal to ActualWindowSize.*

[Change **Clause 5.4.2**, p 25]

**5.4.2 ~~Function InWindow~~ *Window Query Functions***

***5.4.2.1 Function InWindow***

The function "InWindow" performs a modulo 256 compare of two unsigned eight-bit sequence numbers. All computations and comparisons are modulo 256 operations on unsigned eight-bit quantities.

function InWindow(seqA, seqB)

(a) if seqA minus seqB, modulo 256, is less than ActualWindowSize, then return TRUE
(b) else return FALSE.

Example (not normative): if ActualWindowSize is equal to 4, then
InWindow(0, 0) returns TRUE
InWindow(1, 0) returns TRUE
InWindow(3, 0) returns TRUE
InWindow(4, 0) returns FALSE
InWindow(4, 5) returns FALSE (since the modulo 256 difference 4 - 5 = 255)
InWindow(0, 255) returns TRUE (since the modulo 256 difference 0 - 255 = 1)

***5.4.2.2 Function DuplicateInWindow***

*The function "DuplicateInWindow" determines whether a value, seqA, is within the range firstSeqNumber through lastSequenceNumber, modulo 256. All computations and comparisons are modulo 256 operations on unsigned eight-bit quantities.*

*function DuplicateInWindow(seqA, firstSeqNumber, lastSequenceNumber)*

*(a) Set local variable receivedCount to lastSeqNumber minus firstSeqNumber, modulo 256.*
*(b) If receivedCount is greater than ActualWindowSize, then return FALSE.*
*(c) If seqA minus firstSeqNumber, modulo 256, is less then or equal to receivedCount, then return TRUE.*
*(d) Else return FALSE.*

*Example (not normative): if ActualWindowSize is equal to 4, then*
*DuplicateInWindow(0, 0, 1) returns TRUE*
*DuplicateInWindow(1, 0, 1) returns TRUE*
*DuplicateInWindow(2, 0, 1) returns FALSE*
*DuplicateInWindow(3, 0, 1) returns FALSE*

[Change **Clause 5.4.4.2**, p 28]

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 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; *set DuplicateCount 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.)

[Change **Clause 5.4.4.3**, p 29]

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,

then stop RequestTimer; 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; *set DuplicateCount 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.)

[Change **Clause 5.4.4.4**, p 30]

LastSegmentOfGroupReceived

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; whose 'more-follows' parameter is TRUE; and whose 'sequence-number' parameter is equal to InitialSequenceNumber plus ActualWindowSize, modulo 256,

then save the BACnet-ComplexACK-PDU segment; increment LastSequenceNumber, modulo 256; set InitialSequenceNumber to LastSequenceNumber; *set DuplicateCount to zero;* 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; restart SegmentTimer; and enter the SEGMENTED_CONF state to receive additional segments.

LastSegmentOfComplexACK_Received

If a ComplexACK PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; and whose 'more-follows' parameter is FALSE (i.e., the final segment),

then stop SegmentTimer; 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; send CONF_SERV.confirm(+) containing all of the received segments to the local application program; and enter the IDLE state.

*DuplicateSegmentReceived*

*If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not equal to LastSequenceNumber plus 1, modulo 256, and DuplicateInWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber+1 modulo 256, LastSequenceNumber) returns a value of TRUE and DuplicateCount is less than* $N_{dup}$,

*then discard the BACnet-ComplexACK-PDU segment; restart SegmentTimer; increment DuplicateCount, and enter the SEGMENTED_CONF state to receive the remaining segments.*

*TooManyDuplicateSegmentsReceived*

*If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not equal to LastSequenceNumber plus 1, modulo 256, and DuplicateInWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber+1 modulo 256, LastSequenceNumber) returns a value of TRUE and DuplicateCount is equal to* $N_{dup}$*,*

*then discard the BACnet-ComplexACK-PDU segment; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = TRUE, 'server' = FALSE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; set DuplicateCount to zero; and enter the SEGMENTED_CONF state to receive the remaining segments.*

SegmentReceivedOutOfOrder

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not equal to LastSequenceNumber plus 1, modulo 256, *and DuplicateInWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber+1 modulo 256, LastSequenceNumber) returns a value of FALSE,*

then discard the BACnet-ComplexACK-PDU segment; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = TRUE, 'server' = FALSE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; *set InitialSequenceNumber = LastSequenceNumber; set DuplicateCount to zero;* and enter the SEGMENTED_CONF state to receive the remaining segments.

[Change C**lause 5.4.5.1**, p 31]

ConfirmedSegmentedReceived

If a BACnet-Confirmed-Request-PDU whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is zero is received from the network layer and the local device supports the reception of segmented messages,

then compute ActualWindowSize based on the 'proposed-window-size' parameter of the received BACnet-Confirmed-Request-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' = TRUE, and 'actual-window-size' = ActualWindowSize; start SegmentTimer; set LastSequenceNumber to zero; set InitialSequenceNumber to zero; *set DuplicateCount to zero;* and enter the SEGMENTED_REQUEST 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-Confirmed-Request-PDU and shall be in the range 1 to 127, inclusive.)

[Change **Clause 5.4.5.2**, p 32]

LastSegmentOfGroupReceived

If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; whose 'more-follows' parameter is TRUE; and whose 'sequence-number' parameter is equal to InitialSequenceNumber plus ActualWindowSize, modulo 256,

then save the BACnet-Confirmed-Request-PDU segment; increment LastSequenceNumber, modulo 256; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = TRUE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; set InitialSequenceNumber = LastSequenceNumber; *set DuplicateCount to zero;* and enter the SEGMENTED_REQUEST state to receive the remaining segments.

LastSegmentOfMessageReceived

If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; and whose 'more-follows' parameter is FALSE (i.e., the final segment),

then save the BACnet-Confirmed-Request-PDU segment; increment LastSequenceNumber, modulo 256; stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = TRUE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; set InitialSequenceNumber = LastSequenceNumber; send CONF_SERV.indication(+) containing all of the received segments to the local application program; start RequestTimer; and enter the AWAIT_RESPONSE state.

*DuplicateSegmentReceived*

*If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not equal to LastSequenceNumber plus 1, modulo 256, and DuplicateInWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber+1 modulo 256, LastSequenceNumber) returns a value of TRUE and DuplicateCount is less than* $N_{dup}$*,*

*then discard the BACnet-Confirmed-Request-PDU segment; restart SegmentTimer; increment DuplicateCount; and enter the SEGMENTED_REQUEST state to receive the remaining segments.*

*TooManyDuplicateSegmentsReceived*

*If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not equal to LastSequenceNumber plus 1, modulo 256, and DuplicateInWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber+1 modulo 256, LastSequenceNumber) returns a value of TRUE and DuplicateCount is equal to* $N_{dup}$*,*

*then discard the BACnet-Confirmed-Request-PDU segment; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = TRUE, 'server' = TRUE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; set InitialSequenceNumber = LastSequenceNumber; set DuplicateCount to zero; and enter the SEGMENTED_REQUEST state to receive the remaining segments.*

SegmentReceivedOutOfOrder

If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not equal to LastSequenceNumber plus 1, modulo 256, *and DuplicateInWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber+1 modulo 256, LastSequenceNumber) returns a value of FALSE,*

then discard the *BACnet-Confirmed-Request*-PDU *segment*; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = TRUE, 'server' = TRUE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; set InitialSequenceNumber = LastSequenceNumber; *set DuplicateCount to zero*; and enter the SEGMENTED_REQUEST state to receive the remaining segments.

[Add a new entry to **History of Revisions**, p. 688]

**(This History of Revisions is not part of this standard. It is merely informative and does not contain requirements necessary for conformance to the standard.)**

**HISTORY OF REVISIONS**

| ***Protocol*** | | ***Summary of Changes to the Standard*** |
|---|---|---|
| ***Version*** | ***Revision*** | |
| … | … | **…** |
| 1 | 11 | **Addendum *z* to ANSI/ASHRAE 135-2008**<br>Approved by the ASHRAE Standards Committee June 26, 2010; by the ASHRAE Board of Directors June 30, 2010; and by the American National Standards Institute July 1, 2010.<br><br>1. Add Event_Message_Texts.<br>2. Add UnconfirmedEventNotification to Automated Trend Retrieval BIBBs.<br>3. Modify MS/TP State Machine to Ignore Data Not For Us.<br>4. Add New Engineering Units.<br>5. Add Duplicate Segment Detection. |

**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 effect 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, through its Standards 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 source selection should be made by its members.