ANSI/ASHRAE Addendum f to
ANSI/ASHRAE Standard 135.1-2009

ASHRAE ADDENDA

Method of Test for Conformance to BACnet®

Approved by the ASHRAE Standards Committee on January 29, 2011; by the ASHRAE Board of Directors on February 2, 2011; and by the American National Standards Institute on February 3, 2011.

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

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, 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 135.1*f* to ANSI/ASHRAE Standard 135.1-2009 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.

135.1-2009*f*-1. Clarify Tests for Ack Notification Timestamps, p. 2.
135.1-2009*f*-2. Add new Database_Revision tests, p. 15.
135.1-2009*f*-3. Update CreateObject Service tests , p. 18.
135.1-2009*f*-4. Update DeleteObject Service tests, p. 24.

In the following document, language added to existing clauses of ANSI/ASHRAE 135.1-2009 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.

135.1-2009*f*-1. Clarify Tests for Ack Notification Timestamps.

Rationale
The language in the acknowledge alarm service is unclear as to which timestamp parameter is to be sent in the resulting ack notifications. Upon study, it was confirmed that the timestamp that is to be included is the time that the Acknowledgment service is executed. An erratum has been issued to clarify this in Standard 135-2008, and the following proposed change fixes this issue and other issues in the tests of Standard 135.1.

Addendum 135.1-2009*f*-1

[Change **Clause 7.3.1.11,** p. 42**]**

Test Steps:

1. WAIT (Time_Delay + **Notification Fail Time**)
2. VERIFY Event_State = NORMAL
3. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)
4. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
5. IF (X is writable) THEN
   WRITE X = (a value that is OFFNORMAL)
   ELSE
   MAKE (X have a value that is OFFNORMAL)
6. WAIT (Time_Delay)
7. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process ID),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the event-generating object configured for this test),
'Time Stamp' =	(any valid time stamp),
'Notification Class' =	(the class corresponding to the object being tested),
'Priority' =	(the value configured to correspond to a TO-OFFNORMAL transition),
'Event Type' =	(any valid event type),
'Notify Type' =	~~EVENT | ALARM~~*(the notify type configured for this event)*,
'AckRequired' =	TRUE~~| FALSE~~,
'From State' =	NORMAL,
'To State' =	OFFNORMAL,
'Event Values' =	(values appropriate to the event type)

8. *TRANSMIT BACnet-SimpleACK-PDU*
9. VERIFY Event_State = OFFNORMAL
10. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
11. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
12. IF (X is writable) THEN
    WRITE X = (a value that is NORMAL)
    ELSE
    MAKE (X have a value that is NORMAL)
13. WAIT (Time_Delay)
14. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process ID),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the event-generating object configured for this test),
'Time Stamp' =	(any valid time stamp),
'Notification Class' =	(the class corresponding to the object being tested),
'Priority' =	(the value configured to correspond to a TO-NORMAL transition),
'Event Type' =	(any valid event type),

'Notify Type' = ~~EVENT | ALARM~~*(the notify type configured for this event)*,
'AckRequired' = TRUE~~ | FALSE~~,
'From State' = OFFNORMAL,
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)

15. *TRANSMIT BACnet-SimpleACK-PDU*
16. VERIFY Event_State = NORMAL
17. VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)
18. VERIFY Status_Flags = (FALSE, FALSE, ?,?)
19. IF (the event-triggering object can be placed into a fault condition) THEN~~ (~~
20. MAKE (the event-triggering object change to a fault condition)
~~19. WAIT (Time_Delay)~~
21. BEPORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the value configured to correspond to a TO-FAULT transition),
'Event Type' = (any valid event type),
'Notify Type' = ~~EVENT | ALARM~~*(the notify type configured for this event)*,
'AckRequired' = TRUE~~ | FALSE~~,
'From State' = NORMAL,
'To State' = FAULT,
'Event Values' = (values appropriate to the event type)
22. *TRANSMIT BACnet-SimpleACK-PDU*
23. VERIFY Event_State = FAULT
24. VERIFY Acked_Transitions = (FALSE, FALSE, FALSE)
25. VERIFY Status_Flags = (~~FALSE~~*TRUE*, TRUE, ?, ?)
26. MAKE (the event-triggering object change to a normal condition)
~~25. WAIT (Time_Delay)~~
27. BEPORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the value configured to correspond to a TO-NORMAL transition),
'Event Type' = (any valid event type),
'Notify Type' = ~~EVENT | ALARM~~*(the notify type configured for this event)*,
'AckRequired' = TRUE~~ | FALSE~~,
'From State' = FAULT,
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)
28. *TRANSMIT BACnet-SimpleACK-PDU*
29. VERIFY Event_State = NORMAL
30. VERIFY Acked_Transitions = (FALSE, FALSE, FALSE)
31. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
32. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' in step ~~20~~*21*),
'Event Object Identifier' = (the 'Event Object Identifier' in step ~~20~~*21*),
'Event State Acknowledged' = FAULT,
'Time Stamp' = (the 'Time Stamp in step ~~20~~*21*),
'Time of Acknowledgment' = (~~the current time~~*the TD's current time*)

33. RECEIVE BACnet-SimpleACK-PDU

34. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step ~~20~~*21*),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step ~~20~~*21*),
'Time Stamp' = *(the current time or sequence number)*~~(the 'Time Stamp' in step 20)~~,
'Notification Class' = (the 'Notification Class' in step ~~20~~*21*),
'Priority' = (the 'Priority' in step ~~20~~*21*),
'Event Type' = (the 'Event Type' in step ~~20~~*21*),
'Notify Type' = ACK_NOTIFICATION,
'To State' = FAULT
ELSE
BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step ~~20~~*21*),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step ~~20~~*21*),
'Time Stamp' = *(the current time or sequence number)* ~~(the 'Time Stamp' in step 20)~~,
'Notification Class' = (the 'Notification Class' in step ~~20~~*21*),
'Priority' = (the 'Priority' in step ~~20~~*21*),
'Event Type' = (the 'Event Type' in step ~~20~~*21*),
'Notify Type' = ACK_NOTIFICATION

35. *TRANSMIT BACnet-SimpleACK-PDU*

36. VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)
~~}~~

37. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' in step *27*~~13~~),
'Event Object Identifier' = (the 'Event Object Identifier' in step *27*~~13~~),
'Event State Acknowledged' = NORMAL,
'Time Stamp' = (the 'Time Stamp in step *27*~~13~~),
'Time of Acknowledgment' = (~~the current time~~*the TD's current time*)

38. RECEIVE BACnet-SimpleACK-PDU

39. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step *27*~~13~~),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step *27*~~13~~),
'Time Stamp' = *(the current time or sequence number)*~~(the 'Time Stamp' in step 13)~~,
'Notification Class' = (the 'Notification Class' in step *27*~~13~~),
'Priority' = (the 'Priority' in step *27*~~13~~),
'Event Type' = (the 'Event Type' in step *27*~~13~~),
'Notify Type' = ACK_NOTIFICATION,
'To State' = NORMAL
ELSE
BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step *27*~~13~~),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step *27*~~13~~),
'Time Stamp' = *(the current time or sequence number)* ~~(the 'Time Stamp' in step 13)~~,
'Notification Class' = (the 'Notification Class' in step *27*~~13~~),

'Priority' = (the 'Priority' in step *27*~~13~~),
'Event Type' = (the 'Event Type' in step *27*~~13~~),
'Notify Type' = ACK_NOTIFICATION

40. TRANSMIT BACnet-SimpleACK-PDU
41. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
42. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' in step 7),
'Event Object Identifier' = (the 'Event Object Identifier' in step 7),
'Event State Acknowledged' = OFFNORMAL,
'Time Stamp' = (the 'Time Stamp in step 7),
'Time of Acknowledgment' = (~~the current time~~*the TD's current time*)
43. RECEIVE BACnet-SimpleACK-PDU
44. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step 7),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step 7),
'Time Stamp' = *(the current time or sequence number)* ~~(the 'Time Stamp' in step 7)~~,
'Notification Class' = (the 'Notification Class' in step 7),
'Priority' = (the 'Priority' in step 7),
'Event Type' = (the 'Event Type' in step 7),
'Notify Type' = ACK_NOTIFICATION,
'To State' = OFFNORMAL
ELSE
BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step 7),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step 7),
'Time Stamp' = *(the current time or sequence number)* ~~(the 'Time Stamp' in step 7)~~,
'Notification Class' = (the 'Notification Class' in step 7),
'Priority' = (the 'Priority' in step 7),
'Event Type' = (the 'Event Type' in step 7),
'Notify Type' = ACK_NOTIFICATION
45. TRANSMIT BACnet-SimpleACK-PDU
46. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)

[Change **Clause 9.1.1.1,** p. 188**]**

Test Steps:

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. WAIT (Time_Delay)
3. BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (any valid event type),
'Notify Type' = ~~ALARM~~*(the notify type configured for this event)*,
'AckRequired' = TRUE,
'From State' = NORMAL,

'To State' = (any appropriate non-normal event state),
'Event Values' = (the values appropriate to the event type)

4. *TRANSMIT BACnet-SimpleACK-PDU*
5. RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = ~~(the current time or sequence number)~~*(the timestamp or sequence number received in step 3)*,
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (any valid event type),
'Notify Type' = ~~ALARM~~*(the notify type configured for this event)*,
'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = (any appropriate non-normal event state),
'Event Values' = (the values appropriate to the event type)
6. *TRANSMIT BACnet-SimpleACK-PDU*
7. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'011'~~*(FALSE,TRUE,TRUE)*
8. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
'Time Stamp' = (the time stamp conveyed in the notification),
'Time of Acknowledgment' = (the *TD's* current time using a Time format)
9. RECEIVE BACnet-Simple-ACK-PDU
10. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the *current time or sequence number*~~time or sequence number from the notification in step 5~~),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step ~~2~~*3*),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step 3)

ELSE
BEFORE ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the *current time or sequence number*~~time or sequence number from the notification in step 2~~),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),

'Event Type' = (the event type included in step ~~2~~*3*),
'Notify Type' = ACK_NOTIFICATION~~,~~
~~'To State' = (the 'To State' used in step 3)~~

11. TRANSMIT BACnet-SimpleACK-PDU

12. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE ***Notification Fail Time*** RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = ~~(the time or sequence number from the notification in step 2)~~ *(the timestamp or sequence number received in step 10)*,
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step ~~2~~*3*),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step 3)
ELSE
BEFORE ***Notification Fail Time*** RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = ~~(the time or sequence number from the notification in step 2)~~ *(the timestamp or sequence number received in step 10),*
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step ~~2~~*3*),
'Notify Type' = ACK_NOTIFICATION

13. TRANSMIT BACnet-SimpleACK-PDU

14. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'111'~~*(TRUE,TRUE,TRUE)*

Notes to Tester: The destination address used for the acknowledgment notification in step ~~8~~*12* shall be the same address used in step ~~3~~*5*. Inclusion of the 'To State' parameter in acknowledgement notifications was added in protocol version 1, protocol revision 1. Implementations that precede this version will not include this parameter. *When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant.*

[Change **Clause 9.1.1.4,** p. 190**.]**

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. WAIT Time_Delay
3. BEFORE ***Notification Fail Time***
RECEIVE UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ~~ALARM~~*(the notify type configured for this event)*,

'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = (any appropriate non-normal event state),
'Event Values' = (the values appropriate to the event type)

4. IF (the notification in step ~~2~~*3* was not a broadcast) THEN
   RECEIVE
   DESTINATION = (at least one device other than the TD),
   SOURCE = IUT,
   UnconfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object detecting the alarm),
   'Time Stamp' = ~~(the current time or sequence number)~~ *(the timestamp or sequence number received in step 3)*,
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event type),
   'Event Type' = (any valid event type),
   'Notify Type' = ~~ALARM~~*(the notify type configured for this event)*,
   'AckRequired' = TRUE,
   'From State' = NORMAL,
   'To State' = (any appropriate non-normal event state),
   'Event Values' = (the values appropriate to the event type)
5. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'011'~~*(FALSE,TRUE,TRUE)*
6. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (the time stamp conveyed in the notification),
   'Time of Acknowledgment' = (the *TD's* current time using a Time format)
7. RECEIVE BACnet-Simple-ACK-PDU
8. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
   BEFORE ***Notification Fail Time***
   RECEIVE
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   SOURCE = IUT,
   UnconfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object detecting the alarm),
   'Time Stamp' = (the *current time or sequence number*~~time or sequence number from the notification in step 2~~),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event type),
   'Event Type' = (any valid event type),
   'Notify Type' = ACK_NOTIFICATION,
   'To State' = (the 'To State' used in step ~~2~~*3* or ~~3~~*4*)

   ELSE
   BEFORE ***Notification Fail Time***
   RECEIVE
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   SOURCE = IUT,
   UnconfirmedEventNotification-Request,

'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the *current time or sequence number*~~time or sequence number from the notification in step 2~~),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION

9. IF (the notification in step ~~7~~*8* was not broadcast) THEN
IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the time*stamp* or sequence number from the notification in step ~~2~~*8*),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step ~~2~~*3* or ~~3~~*4*)
ELSE
RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the time*stamp* or sequence number from the notification in step ~~2~~*8*),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION~~,~~
'To State' = (the 'To State' used in step 2 or 3)

10. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'111'~~*(TRUE,TRUE,TRUE)*

Notes to Tester: The destination address used for the acknowledgment notification in step ~~8~~*9* shall be the same address used in step ~~3~~*4*. Inclusion of the 'To State' parameter in acknowledgement notifications was added in protocol version 1, protocol revision 1. Implementations that precede this version will not include this parameter. *When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant.*

[Change **Clause 9.1.2.1,** p. 194**]**

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. *WAIT Time_Delay*
3. *BEFORE* ***Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),

'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ~~ALARM~~*(the not*ify *type configured for the event)*,
'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = (any appropriate non-normal event state),
'Event Values' = (the values appropriate to the event type)

4. *TRANSMIT BACnet-SimpleACK-PDU*
5. RECEIVE
   DESTINATION = (at least one device other than the TD),
   SOURCE = IUT,
   ConfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object detecting the alarm),
   'Time Stamp' = ~~(the current time or sequence number)~~*(the timestamp or sequence number received in step 3)*,
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event type),
   'Event Type' = (any valid event type),
   'Notify Type' = ~~ALARM~~*(the not*ify *type configured for the event)*,
   'AckRequired' = TRUE,
   'From State' = NORMAL,
   'To State' = (any appropriate non-normal event state),
   'Event Values' = (the values appropriate to the event type)
6. *TRANSMIT BACnet-SimpleACK-PDU*
7. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'011'~~*(FALSE,TRUE,TRUE)*
8. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (a time stamp older than the one conveyed in the notification),
   'Time of Acknowledgment' = (the current time using a Time format)
9. RECEIVE BACnet-Error-PDU
   Error Class = SERVICES,
   Error Code = INVALID_TIME_STAMP
10. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'011'~~*(FALSE,TRUE,TRUE)*
11. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the process identifier configured for this event),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (the time stamp conveyed in the notification),
   'Time of Acknowledgment' = (the current time using a Time format)
12. RECEIVE BACnet-Simple-ACK-PDU

13. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE ***Notification Fail Time***
RECEIVE
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the *current time or sequence number*~~time or sequence number from the notification in step 2~~),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step ~~2~~*3* or ~~3~~*5*)
ELSE
BEFORE ***Notification Fail Time***
RECEIVE
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the *current time or sequence number*~~time or sequence number from the notification in step 2~~),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION
14. TRANSMIT BACnet-SimpleACK-PDU
15. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
RECEIVE
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the time*stamp* or sequence number from the notification in step ~~2~~*13*),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step ~~2~~*3* or ~~3~~*5*)
ELSE
RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the time*stamp* or sequence number from the notification in step ~~2~~*13*),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = any valid event type),
'Notify Type' = ACK_NOTIFICATION
16. TRANSMIT BACnet-SimpleACK-PDU

17 VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'111'~~*(TRUE,TRUE,TRUE)*

[Change **Clause 9.1.2.5,** p. 198**]**

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. *WAIT Time_Delay*
3. *BEFORE **Noitification Fail Time***
————RECEIVE UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ~~ALARM~~*(the notify type configured for the event)*,
'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = (any appropriate non-normal event state),
'Event Values' = (the values appropriate to the event type)
4. IF (the notification in step ~~2~~*3* was not a broadcast) THEN
RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = ~~(the current time or sequence number)~~ *(the timestamp or sequence number from step 3)*,
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ~~ALARM~~*(the notify type configured for the event)*,
'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = (any appropriate non-normal event state),
'Event Values' = (the values appropriate to the event type)
5. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'011'~~*(FALSE,TRUE,TRUE)*
6. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
'Time Stamp' = (a time stamp older than the one conveyed in the notification),
'Time of Acknowledgment' = (the *TD's* current time using a Time format)
7. RECEIVE BACnet-Error-PDU
Error Class = SERVICES,
Error Code = INVALID_TIME_STAMP
8. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'011'~~*(FALSE,TRUE,TRUE)*
9. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the process identifier configured for this event),
'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),

'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
'Time Stamp' = (the time stamp conveyed in the notification),
'Time of Acknowledgment' = (the *TD's* current time using a Time format)

10. RECEIVE BACnet-Simple-ACK-PDU

11. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN

BEFORE ***Notifcation Fail Time***

RECEIVE

DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the *current time or sequence number*~~time or sequence number from the notification in step 2~~),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step ~~2~~*3* or ~~3~~*4*)

ELSE

BEFORE ***Notifcation Fail Time***

RECEIVE

DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the *current time or sequence number*~~time or sequence number from the notification in step 2~~),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION

12. IF (the notification in step ~~10~~*11* was not broadcast) THEN

IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN

RECEIVE

DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the time*stamp* or sequence number from the notification in step ~~2~~*11*),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step ~~2~~*3* or ~~3~~*4*)

ELSE

RECEIVE

DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,

'Process Identifier' =	(the process identifier configured for this event),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the object detecting the alarm),
'Time Stamp' =	(the time*stamp* or sequence number from the notification in step ~~2~~*11*),
'Notification Class' =	(the notification class configured for this event),
'Priority' =	(the priority configured for this event type),
'Event Type' =	(any valid event type),
'Notify Type' =	ACK_NOTIFICATION

13. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~B'111'~~*(TRUE,TRUE,TRUE)*

Notes to Tester: *The destination address used for the acknowledgment notification in step 11 shall be the same address used in step 3.* The destination address used for the acknowledgment notification in step ~~11~~*12* shall be the same address used in step ~~3~~*4*. *When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant.*

135.1-2009*f*-2. Add new Database_Revision tests.

Rationale
There are currently no tests in Standard 135.1 to test the functionality of the Database_Revision property, but tests for this functionality are needed.

Addendum 135.1-2009*f*-2

[Change **Clause 7.3.2.10,** p. 63**]**

7.3.2.10 Device Object ~~Test~~ *Tests*

These are the tests for the Device object. Other ~~All necessary~~ tests for functionality of the Device object are covered by tests for the application service or special functionality to which they correspond.

[Add new **Clauses 7.3.2.10.X1** through **7.3.2.10.X4,** p. 64.]

7.3.2.10.X1 Successful Increment of the Database_Revision Property after Creating an Object

Dependencies: ReadProperty Service Execution Tests, 9.18

BACnet Reference Clause: 12.11.34

Purpose: To verify that the Database_Revision property of the Device object increments when an object is created. If an object cannot be created, this test shall be omitted.

Test Concept: The Database_Revision property of the Device object is read. An object is created. The Database_Revision property of the Device object is read again to verify that it incremented.

Configuration Requirements: none.

Test Steps:

1. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision
2. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision,
   'Property Value' = (any value = initial value)
3. MAKE (create an object)
4. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision
5. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision,
   'Property Value' = (greater than initial value taking wrapping into account)

7.3.2.10.X2 Successful Increment of the Database_Revision Property after Deleting an Object.

Dependencies: ReadProperty Service Execution Tests, 9.18

BACnet Reference Clause: 12.11.34

Purpose: To verify that the Database_Revision property of the Device object increments when an object is deleted. If an

object cannot be deleted, this test shall be omitted.

Test Concept: The Database_Revision property of the Device object is read. An object is deleted. The Database_Revision property of the Device object is read again to verify that it incremented.

Configuration Requirements: none.

Test Steps:

1. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision
2. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision,
   'Property Value' = (any value = initial value)
3. MAKE (delete an object)
4. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision
5. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision,
   'Property Value' = (greater than initial value taking wrapping into account)

7.3.2.10.X3 Successful Increment of the Database_Revision Property after Changing the Object_Name Property of an Object

Dependencies: ReadProperty Service Execution Tests, 9.18

BACnet Reference Clause: 12.11.34

Purpose: To verify that the Database_Revision property of the Device object increments when the Object_Name property of an object is changed. If the Object_Name property of an object cannot be changed, this test shall be omitted.

Test Concept: The Database_Revision property of the Device object is read. The Object_Name property of an object is changed. The Database_Revision property of the Device object is read again to verify that it incremented.

Configuration Requirements: none.

Test Steps:

1. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision
2. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision,
   'Property Value' = (any value = initial value)
3. MAKE (the Object_Name property of an object change)
4. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision
5. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision,

'Property Value' = (greater than initial value taking wrapping into account)

7.3.2.10.X4 Successful Increment of the Database_Revision Property after Changing the Object_Identifier Property of an Object

Dependencies: ReadProperty Service Execution Tests, 9.18

BACnet Reference Clause: 12.11.34

Purpose: To verify that the Database_Revision property of the Device object increments after changing the Object_Identifier property of an object. If the Object_Identifier property of an object cannot be changed, this test shall be omitted.

Test Concept: The Database_Revision property of the Device object is read. An object's name is changed. The Database_Revision property of the Device object is read again to verify that it incremented.

Configuration Requirements: none.

Test Steps:

1. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision
2. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision,
   'Property Value' = (any value = initial value)
3. MAKE (the Object_Identifier property of an object change)
4. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision
5. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the Device object),
   'Property Identifier' = Database_Revision,
   'Property Value' = (greater than initial value taking wrapping into account)

135.1-2009*f*-3. Update CreateObject Service tests.

Rationale
Change the CreateObject tests to remove inconsistencies in the allowable sets of return codes, update the allowable sets of return codes based on changes in Standard 135, clarify parameter selection language, and make changes that improve automation of the tests.

Addendum 135.1-2009*f*-3

[Change **Clause 8.16.3,** p. 164**]**
[Reason for change: Modify the test to allow it to be executed for objects that have a single modifiable property]

8.16.3 Creating Objects by Specifying the Object Identifier and Providing Initial Values

Purpose: To verify that the IUT can initiate a CreateObject service request that contains an object identifier in the 'Object Specifier' parameter and a list of initial property values for the object to be created.

Test Steps:

1. RECEIVE CreateObject-Request,
   'Object Specifier' = (any BACnetObjectIdentifier)
   'List of Initial Values' = (a list of ~~more than~~ one *or more* ~~BACnetPropertyValues consistent with the specified object type)~~ *properties and their initial values that the IUT will accept)*
2. TRANSMIT CreateObject-ACK,
   'Object Identifier' = (the object identifier specified in step 1)

[Change **Clause 8.16.4,** p. 164**]**
[Reason for change: Modify the test to allow it to be executed for objects that have a single modifiable property]

8.16.4 Creating Objects by Specifying the Object Type and Providing Initial Values

Purpose: To verify that the IUT can initiate a CreateObject service request that contains an object type in the 'Object Specifier' parameter and a list of initial property values for the object to be created.

Test Steps:

1. RECEIVE CreateObject-Request,
   'Object Specifier' = (any BACnetObjectType)
   'List of Initial Values' = (a list of ~~more than~~ one *or more* ~~BACnetPropertyValues consistent with the specified object type)~~ *properties and their initial values that the IUT will accept)*
2. TRANSMIT CreateObject-ACK,
   'Object Identifier' = (an object identifier consistent with the object type specified in step 1)

[Change **Clause 9.16.1.1,** p. 247**]**
[Reason for change: Automate check for the uniqueness of the created object identifier]

9.16.1.1 Creating Objects by Specifying the Object Type with No Initial Values

Purpose: To verify the correct execution of the CreateObject service request when an Object Type is used as the object specifier.

Test Steps:

1. READ X1 = Object_List
2. TRANSMIT CreateObject-Request,
   'Object Specifier' = (any creatable object type)

3. RECEIVE CreateObject-ACK,
   'Object Identifier' = (*O1,* any ~~unique~~ object identifier of the specified type)
4. *CHECK (X1 does not contain O1)*
5. *READ X2 = Object_List*
6. *CHECK (X2 contains O1)*
7. VERIFY (~~the object identifier of the newly created object~~ *O1*),
   (any required property of the specified object) = (any value of the correct datatype for the specified property)

~~4. VERIFY (the IUT's Device object), Object_List = (any object list containing the newly created object)~~

[Change **Clause 9.16.1.3,** p. 248**]**
[Reason for change: Automate check for the uniqueness of the created object identifier]

9.16.1.3 Creating Objects by Specifying the Object Type and Providing Initial Values
Purpose: To verify the correct execution of the CreateObject service request when an Object Type is used as the object specifier and a list of initial property values is provided.

Test Steps:

1. *READ X1 = Object_List*
2. TRANSMIT CreateObject-Request,
   'Object Specifier' = (any creatable object type)
   'List Of Initial Values' = (a list of one or more properties and their initial values *that the IUT will accept*)
3. RECEIVE CreateObject-ACK,
   'Object Identifier' = (O1, any ~~unique~~ object identifier of the specified type)
4. *CHECK (X1 does not contain O1)*
5. *READ X2 = Object_List*
6. *CHECK (X2 contains O1)*
7. REPEAT X = (properties initialized in the CreateObject-Request) DO {
   VERIFY (~~the object identifier for the newly created object~~ O1),
   X = (the value specified in the 'List Of Initial Values' parameter of the CreateObject-Request)
   }

~~4. VERIFY (the IUT's Device object), Object_List = (any object list containing the newly created object)~~

[Change **Clause 9.16.1.4,** p. 248**]**
[Reason for change: Modify the test to allow it to be executed for objects that have a single modifiable property]

9.16.1.4 Creating Objects by Specifying the Object Identifier and Providing Initial Values

Purpose: To verify the correct execution of the CreateObject service request when an Object Identifier is used as the object specifier and a list of initial property values is provided.

Test Steps:

1. TRANSMIT CreateObject-Request,
   'Object Specifier' = (any unique object identifier of a type that is creatable)
   'List Of Initial Values' = (a list of ~~two~~ *one* or more properties and their initial values *that the IUT will accept*)
2. RECEIVE CreateObject-ACK,
   'Object Identifier' = (the object identifier specified in step 1)
3. REPEAT X = (properties initialized in the CreateObject-Request) DO {
   VERIFY (the object identifier for the newly created object),
   X = (the value specified in the 'List Of Initial Values' parameter of the CreateObject-Request)
   }
4. VERIFY (the IUT's Device object), Object_List = (any object list containing the newly created object)

[Change **Clause 9.16.2.2,** p. 249**]**

[Reason for change: clarify that the object type used is one that the device supports, but does not support dynamic creation.]

9.16.2.2 Attempting to Create an Object with an Object Type That is Not Creatable by Specifying the Object Type

Purpose: To verify the correct execution of the CreateObject service request when the 'Object Specifier' parameter conveys an object type that is not dynamically creatable in the IUT.

Test Steps:

1. TRANSMIT CreateObject-Request,
   'Object Specifier' = (any *supported* object type for which dynamic creation is not supported)
2. RECEIVE CreateObject-Error,
   Error Class = OBJECT,
   Error Code = DYNAMIC_CREATION_NOT_SUPPORTED
   'First Failed Element Number' = 0

[Change **Clause 9.16.2.3,** p. 249**]**
[Reason for change: ensure that the tester selects a type that is not creatable with an instance that normally would be.]

9.16.2.3 Attempting to Create an Object with an Object Identifier That is Not Creatable by Specifying the Object Identifier

Purpose: To verify the correct execution of the CreateObject service request when the 'Object Specifier' parameter conveys an object identifier for an object type that is not dynamically creatable in the IUT.

Test Steps:

1. TRANSMIT CreateObject-Request,
   'Object Specifier' = (any object identifier having a supported object type for which dynamic creation is not supported)
2. RECEIVE CreateObject-Error,
   Error Class = OBJECT,
   Error Code = DYNAMIC_CREATION_NOT_SUPPORTED
   'First Failed Element Number' = 0
3. VERIFY (the IUT's Device object),
   Object_List = (any object list that does not contain the object specified in step 1)

Notes to tester: If the IUT limits the instances that can be created, this shall be taken into account when selecting an object identifier in step 1.

[Change **Clause 9.16.2.4,** p. 249**]**
[Reason for change: Automate check for the uniqueness of the created object identifier and allow the test to be executed for objects that have a single modifiable property]
[Note to reviewers: The highlighted vertical bar in step 5 is part of the added text]

9.16.2.4 Attempting to Create an Object with an Object Type Object Specifier and an Error in the Initial Values

Purpose: To verify the correct execution of the CreateObject service request when an object type is used as the object specifier and a list of initial property values containing an invalid value is provided.

Test Steps:

1. READ X1 = Object_List
2. TRANSMIT CreateObject-Request,

'Object Specifier' = (any creatable object type),
'List Of Initial Values' = (a list of ~~two~~ *one* or more properties and their initial*, that the IUT will accept initial values for,* values with one of the values being out of range)

3. IF (Protocol_Revision is present and Protocol_Revision ≥ 4) THEN
RECEIVE CreateObject-Error ~~PDU~~,
Error Class = PROPERTY,
Error Code = VALUE_OUT_OF_RANGE
'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)
ELSE
RECEIVE CreateObject-Error,
Error Class = PROPERTY,
Error Code = VALUE_OUT_OF_RANGE |
~~OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED |~~ OTHER
'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)

~~*4*~~. CHECK(Verify that the new object was not created)

5. TRANSMIT CreateObject-Request,
'Object Specifier' = (~~any creatable~~ object type *from step 2*),
'List Of Initial Values' = (a list of ~~*two*~~ *one* or more properties and their initial values*, that the IUT will accept initial values for,* with one of the values being an inappropriate datatype)

6. IF (Protocol_Revision is present and Protocol_Revision ≥ 4) THEN
RECEIVE CreateObject-Error ~~PDU~~,
Error Class = PROPERTY,
Error Code = INVALID_DATATYPE
'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)
ELSE
RECEIVE CreateObject-Error,
Error Class = PROPERTY,
Error Code = VALUE_OUT_OF_RANGE | INVALID_DATATYPE |
~~OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED |~~OTHER
'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value) |
(BACnet-Reject-PDU
Reject Reason = INVALID_PARAMETER_DATATYPE | INVALID_TAG)

~~6. CHECK(Verify that the new object was not created)~~

7. READ X2 = Object_List

8. CHECK (X1 = X2)

[Change **Clause 9.16.2.5,** p. 250**]**

[Note to reviewers: The highlighted vertical bars in the new step 5 are part of the added text]

9.16.2.5 Attempting to Create an Object with an Object Identifier Object Specifier and an Error in the Initial Values

Purpose: To verify the correct execution of the CreateObject service request when an object identifier is used as the object specifier and a list of initial property values containing an invalid value is provided.

Test Steps:

1. TRANSMIT CreateObject-Request,
'Object Specifier' = (any unique object identifier of a type that is creatable *and an instance number that is creatable*),
'List Of Initial Values' = (a list of ~~two~~ *one* or more properties and their initial values*, that the IUT will accept initial values for,* with one of the values being out of range)
2. IF (Protocol_Revision is present and Protocol_Revision ≥ 4) THEN
RECEIVE CreateObject-Error ~~PDU~~,

Error Class = PROPERTY,
Error Code = VALUE_OUT_OF_RANGE
'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)
ELSE
RECEIVE CreateObject-Error,
Error Class = PROPERTY,
Error Code = VALUE_OUT_OF_RANGE | ~~INVALID_DATATYPE~~ *OTHER*
'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)

3. *CHECK(Verify that the new object was not created)*
4. TRANSMIT CreateObject-Request,
'Object Specifier' = (~~any unique~~ object identifier *from step 1*~~of a type that is creatable~~),
'List Of Initial Values' = (a list of ~~two~~ *one* or more properties and their initial values*, that the IUT will accept initial values for,* with one of the values being an inappropriate datatype)
5. IF (Protocol_Revision is present and Protocol_Revision ≥ 4) THEN
RECEIVE CreateObject-Error ~~PDU~~,
Error Class = PROPERTY,
Error Code = INVALID_DATATYPE
'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)
ELSE
RECEIVE CreateObject-Error,
Error Class = PROPERTY,
Error Code = *VALUE_OUT_OF_RANGE* | INVALID_DATATYPE | *OTHER*
'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value) |
(BACnet-Reject-PDU
Reject Reason = INVALID_PARAMETER_DATATYPE | INVALID_TAG)
6. TRANSMIT ReadProperty-Request,
'Object Identifier' = (the 'Object Identifier' used in step 1),
'Property Identifier' = ~~(any required property of the specified object)~~*Object_Name*
7. IF (Protocol_Revision is present and Protocol_Revision ≥ 4) THEN
RECEIVE BACnet-Error PDU,
Error Class = OBJECT,
Error Code = UNKNOWN_OBJECT
ELSE
RECEIVE BACnet-Error PDU
Error Class = OBJECT,
Error Code = UNKNOWN_OBJECT | NO_OBJECTS_OF_SPECIFIED_TYPE | OTHER

135.1-2009*f*-4. Update DeleteObject Service tests.

Rationale
Change the DeleteObject tests to remove unnecessary test dependencies on EPICS.

Addendum 135.1-2009*f*-4

[Change **Clause 9.17.1.1,** p. 252**]**
[Reason for change: Remove unnecessary reliance on the EPICS]

9.17.1.1 Successful Deletion of an Object

Purpose: To verify the ability to successfully delete an object.

Configuration Requirements: The IUT shall be configured with an object X that can be deleted.

Test Steps:

1. VERIFY (X), Object_Name = (~~the Object_Name specified in the EPICS~~ *any valid value*)
2. TRANSMIT DeleteObject-Request,
   'Object Identifier' = X
3. RECEIVE BACnet-Simple-ACK-PDU
4. TRANSMIT ReadProperty-Request,
   'Object Identifier' = X,
   'Property Identifier' = Object_Name
5. RECEIVE BACnet-Error-PDU,
   Error Class = OBJECT,
   Error Code = UNKNOWN_OBJECT
6. VERIFY (X), Object_List = (any object list that does not contain X)

[Add a new entry to **History of Revisions**, p. 489]

(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

Summary of Changes to the Standard
…
Addendum *f* to ANSI/ASHRAE 135.1-2009 Approved by the ASHRAE Standards Committee January 29, 2011; by the ASHRAE Board of Directors February 2, 2011; and by the American National Standards Institute February 3, 2011. 1. Clarify Tests for Ack Notification Timestamps. 2. Add new Database_Revision tests. 3. Update CreateObject Service tests. 4. Update DeleteObject Service tests.

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.