BACnet *Errata*
ANSI/ASHRAE STANDARD 135.1-2019
A Data Communication Protocol for Building Automation and Control Networks

June 18, 2021

This document lists *errata additions* to ANSI/ASHRAE Standard 135.1-2019 as of the above date. Each entry is cited first by clause, then page number, except where an erratum covers more than one clause. The back page marking identifying the electronic publication of Standard 135.1-2019 is "Product code: D-86437 9/19".

Changes are indicated by using ~~strikeout~~ for text to be removed and *italics* for text to be added, unless noted otherwise. Grey highlighting is used for marking small changes.

1) **Clause 12.1.3.9 –** Frame Type explicitly stated

12.1.3.9 Verify $T_{usage\_timeout}$ w/ Serial Analyzer
…
Test Steps:

1. MAKE (Power on both devices.)
2. WAIT (several seconds)
3. VERIFY (Has Token passing been established between the devices?)
4. MAKE (Power off the other master device, but not the IUT.)
5. WAIT (10 seconds)
6. MAKE (Stop the data capture.)
7. CHECK (Did the IUT send a ~~type 0~~ *Token* frame to the other master, and, when the other master did not use the Token (because it was powered off), did the IUT follow the ~~type 0~~ *Token* frame with one ~~type 0~~ *Token* frame (Token retry) followed by a series of ~~type 1~~ *Poll For Master* frames?)
8. CHECK (Is the time difference between the last octet of the ~~type 0~~ *Token* frame sent by the IUT and the first octet of the immediately following ~~type 1~~ *Poll For Master* frame transmitted by the IUT greater than 20 millseconds - T_{neg_err} and less than 100 millseconds + T_{pos_err}?)
9. CHECK (Is the time gap (last character to first character) between any two ~~type 1~~ *Poll For Master* frames ~~(Poll For Master)~~ sent by the IUT greater than 20 millseconds - T_{neg_err}, but less than 100 millseconds + T_{pos_err}?)

2) **Clause 12.1.3.3** – Should not require a Data Frame

12.1.3.3 Verify T_{frame_gap}

Purpose: Verify that the maximum idle time between ~~data~~ octets when transmitting a frame is 20 bit times or less.

Test Steps:

1. Elicit the transmission of any ~~data~~ frame from the IUT.
2. Measure the longest EIA-485 idle time that appears between octets within the ~~data~~ frame transmitted by the IUT. If there is no idle time between octets, pass the IUT.
3. Fail the IUT if the time measured in step 2 is greater than the time intervals shown below for each baud rate.

9600 baud:	fail if interval is greater than 2,083 microseconds
19200 baud:	fail if interval is greater than 1,042 microseconds
38400 baud:	fail if interval is greater than 521 microseconds
57600 baud:	fail if interval is greater than 347 microseconds
76800 baud:	fail if interval is greater than 261 microseconds
115200 baud:	fail if interval is greater than 173 microseconds
x baud:	fail if interval is greater than (20/x) seconds

3) **Document to PDF Conversion failed for Some '≥' and '≤' Characters**

7.3.1.11

```
…
3.  IF (Protocol_Revision is present AND Protocol_Revision ≥~~e~~ 13) THEN
…
10. IF (Protocol_Revision is present AND Protocol_Revision ≥~~e~~ 13) THEN
…
17. IF (Protocol_Revision is present AND Protocol_Revision ≥~~e~~ 13) THEN
…
```

7.3.1.16

```
…
3.  WRITE (the array property being tested) = (array of non-zero size $N_2$, where $N_2$ ≤~~d~~ $N_1$)
4.  VERIFY (array is as written in step 3)
5.  WRITE (the array property being tested) = (array of non-zero size $N_3$, where $N_3$ ≥~~e~~ $N_1$)
6.  VERIFY (array is as written in step 5)
7.  WRITE (the array property being tested) = (a non-zero unsigned value $N_4$, where $N_4$ ≤~~d~~ $N_1$), ARRAY INDEX = 0
8.  VERIFY (array contains first $N_4$ elements of the array written in step 5)
9.  WRITE (the array property being tested) = ($N_5$, where $N_5$ ≥~~e~~ $N_4$), ARRAY INDEX = 0
10. VERIFY (array contains first $N_4$ elements of the array written in step 5, plus $N_5 – N_4$ additional elements, initialized to
            particular values if specified for the array property being tested)
11. TRANSMIT WriteProperty-Request,
        'Object Identifier' =           (the object being tested),
        'Property Identifier' =         (the array property being tested),
        'Property Array Index' =        ($N_6$, where $N_6$ ≥~~e~~ $N_5$),
        'Property Value' =              (one array element)
…
```

7.3.2.9.7

```
…
3.  IF (Protocol_Revision is present and Protocol_Revision ≥~~e~~ 10) THEN
…
```

8.4.1

```
…
6.  IF (Protocol_Revision is present AND Protocol_Revision ≥~~e~~ 13) THEN
…
13. IF (Protocol_Revision is present AND Protocol_Revision ≥~~e~~ 13) THEN
…
```

8.4.2

```
…
2.  IF ((Protocol_Revision is present AND Protocol_Revision ≥~~e~~ 13)
            OR ((Protocol_Revision is present AND Protocol_Revision ~~d 12~~< *13*)
            AND (pAlarmValues contains at least one value))) THEN {
        IF (pMonitoredValue is writable) THEN
            WRITE pMonitoredValue = (a value from pAlarmValues)
        ELSE
        MAKE (pMonitoredValue have a value pAlarmValues)
    WAIT (pTimeDelay)
    BEFORE **Notification Fail Time**
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                object being tested),
            'Time Stamp' =                      (T1, any valid time stamp),
            'Notification Class' =              (the configured notification class),
            'Priority' =                        (the value configured to correspond to a TO_OFFNORMAL
                                                transition),
            'Event Type' =                      CHANGE_OF_STATE,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      NORMAL,
            'To State' =                        OFFNORMAL,
            'Event Values' =                    pMonitoredValue, pStatusFlags
    TRANSMIT BACnet-SimpleACK-PDU
    IF (Protocol_Revision is present AND Protocol_Revision ≥~~e~~ 13) THEN
        VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
    VERIFY pCurrentState = OFFNORMAL
    IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
        VERIFY Event_Time_Stamps = (T1, Ta, Tb)
    IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = (a value that corresponds to a NORMAL state)
    ELSE
        MAKE (pMonitoredValue have a value that corresponds to a NORMAL state)
    WAIT (pTimeDelay)
    BEFORE **Notification Fail Time**
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                object being tested),
            'Time Stamp' =                      (T2, any valid time stamp),
            'Notification Class' =              (the configured notification class),
            'Priority' =                        (the value configured to correspond to a TO_NORMAL transition),
            'Event Type' =                      CHANGE_OF_STATE,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      OFFNORMAL,
            'To State' =                        NORMAL,
            'Event Values' =                    pMonitoredValue, pStatusFlags
    TRANSMIT BACnet-SimpleACK-PDU
    IF (Protocol_Revision is present AND Protocol_Revision ≥~~e~~ 13) THEN
        VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
    VERIFY pCurrentState = NORMAL
    IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
        VERIFY Event_Time_Stamps = (T1, Ta, T2)
  }
3.  IF ((Protocol_Revision is present AND Protocol_Revision ~~d 12~~< *13*)
…
```

8.4.3.1

```
…
9.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
```

8.4.3.2

```
…
9.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
```

8.4.4

```
…
2.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
7.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
14. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
```

8.4.5

```
…
10. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
21. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
32. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
43. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
```

8.4.6

```
…
10. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
21. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
32. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
43. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
```

8.4.8.1

```
…
7.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
        VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
8.  VERIFY pCurrentState = OFFNORMAL
9.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
…
```

8.4.8.2

…

```
4.  IF (latching is supported) THEN {
        CHECK (pCurrentState = OFFNORMAL)
        MAKE (the object reset)
        BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                object being tested),
            'Time Stamp' =                      (T1: any valid time stamp),
            'Notification Class' =              (the configured notification class),
             'Priority' =                       (the value configured to correspond to a TO-OFFNORMAL transition),
            'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
            'Message Text' =                    (S1: optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      OFFNORMAL,
            'To State' =                        NORMAL,
            'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU,
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (FALSE, FALSE, ?, ?),
        VERIFY pCurrentState = NORMAL,
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (*, *, T1)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (*, *, S1)
    }
    ELSE {
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                    object being tested),
                'Time Stamp' =                      (T2: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                   'Priority' =                     (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                    (S2: optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      OFFNORMAL,
                'To State' =                        NORMAL,
                'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
        VERIFY pCurrentState = NORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (*, *, T2)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (*, *, S2)
    }
```

8.4.8.3

…

```
7.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
        VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
8.  VERIFY pCurrentState = LIFE_SAFETY_ALARM
9.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
```

…

8.4.8.4

…

```
4.  IF (latching is supported) THEN {
        CHECK (pCurrentState = LIFE_SAFETY_ALARM)
        MAKE (the object reset)
        BEFORE **Notification Fail Time**
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment object
                                                being tested),
            'Time Stamp' =                      (T1: any valid time stamp),
            'Notification Class' =              (the configured notification class),
            'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),
            'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
            'Message Text' =                    (S1: optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      LIFE_SAFETY_ALARM,
            'To State' =                        NORMAL,
            'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
            VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
        VERIFY pCurrentState = NORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
            VERIFY Event_Time_Stamps = (*, *, T1)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (*, *, S1)
    }
    ELSE {
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment object
                                                    being tested),
                'Time Stamp' =                      (T2: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                    (S2: optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
            'From State' =                  LIFE_SAFETY_ALARM,
            'To State' =                    NORMAL,
            'Event Values' =                pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
        VERIFY pCurrentState = NORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (*, *T2)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (*, *, S2)
    }
```

8.4.8.5

…

```
4.  IF (latching is supported) THEN {
        CHECK (pCurrentState = LIFE_SAFETY_ALARM)
        MAKE (the object reset)
        BEFORE **Notification Fail Time**
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                object being tested),
            'Time Stamp' =                      (T1: any valid time stamp),
            'Notification Class' =              (the configured notification class),
            'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),
            'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
            'Message Text' =                    (S1: optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      LIFE_SAFETY_ALARM,
            'To State' =                        OFFNORMAL,
            'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = OFFNORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (T1, *, *)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (S1, *, *)
    }
    ELSE {
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                     object being tested),
                'Time Stamp' =                      (T2: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                    (S2: optional, any valid message text),
                'Notify Type' =              EVENT | ALARM,
                'AckRequired' =              TRUE | FALSE,
                'From State' =               LIFE_SAFETY_ALARM,
                'To State' =                 OFFNORMAL,
                'Event Values' =             pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = OFFNORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
            VERIFY Event_Time_Stamps = (T2, *, *)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (S2, *, *)
    }
```

8.4.8.6

…

```
4.  IF (latching is supported) THEN {
        CHECK (pCurrentState = OFFNORMAL)
        MAKE (the object reset)
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment object
                                                    being tested),
                'Time Stamp' =                      (T1: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),

                'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                    (S1: optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      OFFNORMAL,
                'To State' =                        LIFE_SAFETY_ALARM,
                'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = LIFE_SAFETY_ALARM
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
            VERIFY Event_Time_Stamps = (T1, *, *)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (S1, *, *)
    }
    ELSE {
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment object
                                                    being tested),
                'Time Stamp' =                      (T2: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =              CHANGE_OF_LIFE_SAFETY,
                'Message Text' =            (S2: optional, any valid message text),
                'Notify Type' =             EVENT | ALARM,
                'AckRequired' =             TRUE | FALSE,
                'From State' =              OFFNORMAL,
                'To State' =                LIFE_SAFETY_ALARM,
                'Event Values' =            pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = LIFE_SAFETY_ALARM
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (T2, *, *)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (S2, *, *)
    }
```

8.4.8.7

```
…
8.  IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
        VERIFY pStatusFlags =               (FALSE, FALSE, ?, ?)
9.  VERIFY pCurrentState =                  NORMAL
10. IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
…
17. IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
        VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
18. VERIFY pCurrentState = OFFNORMAL
19. IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
…
26. IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
        VERIFY pStatusFlags =                   (TRUE, FALSE, ?, ?)
27. VERIFY pCurrentState =                      OFFNORMAL
28. IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
…
```

8.4.8.8

```
…
6.  IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
        VERIFY pStatusFlags =             (TRUE, FALSE, ?, ?)
7.  VERIFY pCurrentState =                OFFNORMAL
8.  IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
…
```

8.4.8.9

```
…
3.  IF (latching is supported) THEN {
        CHECK (pCurrentState = OFFNORMAL)
        MAKE (the object reset)
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =             (any valid process ID),
                'Initiating Device Identifier' =   IUT,
                'Event Object Identifier' =        (the intrinsic reporting object being tested or the Event Enrollment
                                        object being tested),
            'Time Stamp' =              (T1: any valid time stamp),
            'Notification Class' =      (the configured notification class),
            'Priority' =                (the value configured to correspond to a TO-OFFNORMAL transition),
            'Event Type' =              CHANGE_OF_LIFE_SAFETY,
            'Message Text' =            (S1: optional, any valid message text),
            'Notify Type' =             EVENT | ALARM,
            'AckRequired' =             TRUE | FALSE,
            'From State' =              OFFNORMAL,
            'To State' =                NORMAL,
            'Event Values' =            pMonitoredValue, pMode, pStatusFlags, pOperationExpected
    TRANSMIT BACnet-SimpleACK-PDU
    IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
        VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
    VERIFY pCurrentState = NORMAL
    IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
        VERIFY Event_Time_Stamps = (*, *, T1)
    IF (Event_Message_Texts property exists) THEN
        VERIFY Event_Message_Texts = (*, *, S1)
}
ELSE {
    BEFORE **Notification Fail Time**
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =          (any valid process ID),
            'Initiating Device Identifier' = IUT,
            'Event Object Identifier' =     (the intrinsic reporting object being tested or the Event Enrollment
                                             object being tested),
            'Time Stamp' =                  (T2: any valid time stamp),
            'Notification Class' =          (the configured notification class),
            'Priority' =                    (the value configured to correspond to a TO-OFFNORMAL transition),
            'Event Type' =                  CHANGE_OF_LIFE_SAFETY,
            'Message Text' =                (S2: optional, any valid message text),
            'Notify Type' =                 EVENT | ALARM,
            'AckRequired' =                 TRUE | FALSE,
            'From State' =                  OFFNORMAL,
            'To State' =                    NORMAL,
            'Event Values' =                pMonitoredValue, pMode, pStatusFlags, pOperationExpected
    TRANSMIT BACnet-SimpleACK-PDU
    IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
        VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
    VERIFY pCurrentState = NORMAL
    IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
        VERIFY Event_Time_Stamps = (*, *,T2)
    IF (Event_Message_Texts property exists) THEN
        VERIFY Event_Message_Texts = (*, *, S2)
}
```

8.4.8.10

```
…
6.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
        VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
7.  VERIFY pCurrentState = LIFE_SAFETY_ALARM
8.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
…
```

8.4.8.11

```
…
3.  IF (latching is supported) THEN {
        CHECK (pCurrentState = LIFE_SAFETY_ALARM)
        MAKE (the object reset)
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                    object being tested),
                'Time Stamp' =                      (T1: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),

                'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                    (S1: optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      LIFE_SAFETY_ALARM,
                'To State' =                        NORMAL,
                'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
        VERIFY pCurrentState = NORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (*, *, T1)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (*, *, S1)
    }
    ELSE {
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                    object being tested),
                'Time Stamp' =                      (T2: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                    (S2: optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      LIFE_SAFETY_ALARM,
                'To State' =                        NORMAL,
                'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
        VERIFY pCurrentState = NORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (*, *, T2)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (*, *, S2)
    }
```

8.4.8.12

…

```
2.  MAKE (pMode a different value that forces the state to OFFNORMAL)
    IF (latching is supported) THEN {
        CHECK (pCurrentState = OFFNORMAL)
        MAKE (the object reset)
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =            (any valid process ID),
                'Initiating Device Identifier' =  IUT,
                'Event Object Identifier' =       (the intrinsic reporting object being tested or the Event Enrollment
                                                  object being tested),
                'Time Stamp' =                    (T1: any valid time stamp),
                'Notification Class' =            (the configured notification class),
                'Priority' =                      (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =                    CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                  (S1: optional, any valid message text),
                'Notify Type' =                   EVENT | ALARM,
                'AckRequired' =                   TRUE | FALSE,
                'From State' =                    LIFE_SAFETY_ALARM,
                'To State' =                      OFFNORMAL,
                'Event Values' =                  pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = OFFNORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (T1, *, *)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (S1, *, *)
    }
    ELSE {
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =            (any valid process ID),
                'Initiating Device Identifier' =  IUT,
                'Event Object Identifier' =       (the intrinsic reporting object being tested or the Event Enrollment
                                                  object being tested),
                'Time Stamp' =                    (T2: any valid time stamp),
                'Notification Class' =            (the configured notification class),
                'Priority' =                      (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =                    CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                  (S2: optional, any valid message text),
                'Notify Type' =                   EVENT | ALARM,
                'AckRequired' =                   TRUE | FALSE,
                'From State' =                    LIFE_SAFETY_ALARM,
                'To State' =                      OFFNORMAL,
                'Event Values' =                  pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = OFFNORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
```

```
            VERIFY Event_Time_Stamps = (T2, *, *)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (S2, *, *)
    }
```

8.4.8.13

…

```
3.  IF (latching is supported) THEN
        CHECK (pCurrentState = OFFNORMAL)
        MAKE (the object reset)
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                    object being tested),
                'Time Stamp' =                      (T1: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                    (S1: optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      LIFE_SAFETY_ALARM,
                'To State' =                        OFFNORMAL,
                'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = OFFNORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
            VERIFY Event_Time_Stamps = (T1, *, *)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (S1, *, *)
    ELSE {
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested or the Event Enrollment
                                                    object being tested),
                'Time Stamp' =                      (T2: any valid time stamp),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured to correspond to a TO-OFFNORMAL transition),
                'Event Type' =                      CHANGE_OF_LIFE_SAFETY,
                'Message Text' =                    (S2: optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      LIFE_SAFETY_ALARM,
                'To State' =                        OFFNORMAL,
                'Event Values' =                    pMonitoredValue, pMode, pStatusFlags, pOperationExpected
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = OFFNORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (T2, *, *)
        IF (Event_Message_Texts property exists) THEN
            VERIFY Event_Message_Texts = (S2, *, *)
    }
```

8.4.9

…

```
1.  IF (the object generates TO-OFFNORMAL transitions) THEN {
        READ CS1 = pCurrentState
        MAKE (an OFFNORMAL condition exist)
        WAIT D1
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the event generating object),
                'Time Stamp' =                      (TS1: the current local time),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured for TO_OFFNORMAL),
                'Event Type' =                      EXTENDED,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      CS1,
                'To State' =                        (CS2: any offnormal valid event state),
                'Event Values' =                    (pVendorId: any valid vendor id),
                                                    (pEventType: any valid event-type),
                                                    (a list of 0 or more valid parameters as defined by the Vendor)
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 13) THEN
            VERIFY pStatusFlags = (TRUE, FALSE,?,?)
        VERIFY pCurrentState = CS2
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            VERIFY Event_Time_Stamps = (TS1, *, *)
    }
    IF (the object generates TO_NORMAL transitions) THEN {
        READ CS2 = pCurrentState
        MAKE (a NORMAL condition exist)
        WAIT D2
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the intrinsic reporting object being tested),
                'Time Stamp' =                      (TS2: the current local time),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured for TO-NORMAL),
                'Event Type' =                      EXTENDED,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      CS2,
                'To State' =                        NORMAL,
                'Event Values' =                    (pVendorId: any valid vendor id),
                                        (pEventType: any valid event-type),
                                        (a list of 0 or more valid parameters as defined by the Vendor)
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
            VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
        VERIFY pCurrentState = NORMAL
        IF (Protocol_Revision is present AND Protocol_Revision ≥e 1) THEN
            VERIFY Event_Time_Stamps = (*, *, TS2)
    }
```

8.4.13

```
…
2.  IF ((Protocol_Revision is present AND Protocol_Revision ≥ 13)
            OR ((Protocol_Revision is present AND Protocol_Revision ~~d 12~~< 13)
            AND (pAlarmValues contains at least one characterstring))) THEN {
…
7.      IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
15.         IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
22.     IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
```

8.4.15

```
…
6.  IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
12. IF (Protocol_Revision is present AND Protocol_Revision ≥e 13) THEN
…
```

8.23.2

```
…
1.  RECEIVE WritePropertyMultiple-Request,
        'Object Identifier' =    (any valid object identifier),
        'Property Identifier' =  (any valid non-array property of the specified object),
        'Property Value' =       (any value appropriate to the specified property),
        'Property Identifier' =  (any valid property of the specified object that was not previously used),
        'Property Value' =       (any value appropriate to the specified property)
--      …   (Any number of properties ≥e 2 is acceptable.)
…
```

8.23.3

```
…
1.  RECEIVE WritePropertyMultiple-Request,
        'Object Identifier' =    (any valid object identifier),
        'Property Identifier' =  (any valid non-array property of the specified object),
        'Property Value' =       (any value appropriate to the specified property),
        'Object Identifier' =    (any valid object identifier not previously used),
        'Property Identifier' =  (any valid property of the specified object),
        'Property Value' =       (any value appropriate to the specified property)
--      …   (Any number of (object, property, value) tuples ≥e 2 is acceptable.)
…
```

9.1.1.10

…
Test Concept: An event is triggered and the IUT notifies the TD and one other device. The TD acknowledges the event and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all recipients that the event has been acknowledged. The TD then acknowledges the event again, and the IUT again notifies all recipients. This behavior was not defined before Protocol_Revision 7 and so this test shall only be performed if Protocol_Revision is present (i.e., Protocol_Revision ≥ 7).
…

9.1.1.11

…
Test Concept: An event is triggered and the IUT notifies the TD and one other device. The TD acknowledges the event and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all recipients that the event has been acknowledged. The TD then acknowledges the event again, and the IUT again notifies all recipients. This behavior was not defined before Protocol_Revision 7 and so this test shall only be performed if Protocol_Revision is present (i.e., Protocol_Revision ≥ 7).
…

9.21.1.4

…
```
1.  TRANSMIT ReadRange-Request,
        'Object Identifier' =      (the log object configured for this test),
        'Property Identifier' =    Log_Buffer,
        'Reference Time' =         (any value x: x is older (of earlier time) than the time of an entry in
                                   the buffer which has a sequence number of S, and x is newer
                                   than or equal to the time of any preceding entry),
        'Count' =                  (any value y: 0 < y ≤ Total_Record_Count – S + 1)
```
…

9.21.1.4.1

…
```
1.  TRANSMIT ReadRange-Request,
        'Object Identifier' =      (the log object configured for this test),
        'Property Identifier' =    Log_Buffer,
        'Reference Time' =         (x, selected as described above),
        'Count' =                  (any value y: 0 < |y| ≤ number of records in the buffer)
```
…

13.2.2

13.2.2 TimeSynchronization Recipients Test, Protocol_Revision ≥ 7
…

13.2.6

Dependencies: 13.2.2, "TimeSynchronization Recipients Test, Protocol_Revision ≥ 7".
…

13.8.2.1

…

Backup Characteristics:

…

8. The TD is configured with a Protocol_Revision ≥e 10. This is only used if the IUT claims Protocol_Revision ≥e 10.

Restore Characteristics:

…

8. The TD is configured with a Protocol_Revision ≥e 10. This is only used if the IUT claims Protocol_Revision ≥e 10.

…