ADDENDA

ANSI/ASHRAE Addendum j to
ANSI/ASHRAE Standard 135.1-2011

Method of Test for Conformance to BACnet®

Approved by the ASHRAE Standards Committee on June 23, 2012; by the ASHRAE Board of Directors on June 27, 2012; and by the American National Standards Institute on June 28, 2012.

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

SPECIAL NOTE

This American National Standard (ANS) is a national voluntary consensus standard developed under the auspices of 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.

DISCLAIMER

ASHRAE uses its best efforts to promulgate Standards and Guidelines for the benefit of the public in light of available information and accepted industry practices. However, ASHRAE does not guarantee, certify, or assure the safety or performance of any products, components, or systems tested, installed, or operated in accordance with ASHRAE's Standards or Guidelines or that any tests conducted under its Standards or Guidelines will be nonhazardous or free from risk.

ASHRAE INDUSTRIAL ADVERTISING POLICY ON STANDARDS

ASHRAE Standards and Guidelines are established to assist industry and the public by offering a uniform method of testing for rating purposes, by suggesting safe practices in designing and installing equipment, by providing proper definitions of this equipment, and by providing other information that may serve to guide the industry. The creation of ASHRAE Standards and Guidelines is determined by the need for them, and conformance to them is completely voluntary.

In referring to this Standard or Guideline and in marking of equipment and in advertising, no claim shall be made, either stated or implied, that the product has been approved by ASHRAE.

[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

The purpose of this addendum is to present a proposed change for public review. 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 proposed changes are summarized below.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135.1-2011 and Addenda is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout. Only this new and deleted text is open to comment at this time. All other material in this addendum is provided for context only and is not open for public review comment except as it relates to the proposed changes.

135.1-2011j-1. Improve the Read All Properties Test.

Rationale

The existing 7.1 test does not consider the IUT's behavior in cases where a property either cannot be read by ReadProperty or whose value may be too large to return in the given APDU and segment limitations of the IUT.

[Change **Clause 7.1.1**, p. 32]

7.1.1 Read Support Test Procedure

Dependencies: ReadProperty Service Execution Tests, 9.18.

Purpose: ~~To verify that all properties of all objects can be~~ *Verifies that a correct response is returned when each property of each object is* read using BACnet ReadProperty and ReadPropertyMultiple services. The test is performed once using ReadProperty and once using ReadPropertyMultiple. When verifying array properties, ~~the~~ whole array*s* shall be read without using an array index, where possible.

Test Steps:

1. REPEAT X = (all objects in the IUT's database) DO {
   REPEAT Y = (all properties in object X) DO {
   IF (Y is defined by the standard as not accessible via the ReadProperty service) THEN
   TRANSMIT ReadProperty-Request,
   'Object Identifier' = X,
   'Property Identifier' = Y
   IF (Protocol_Revision >= 7) THEN
   RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = READ_ACCESS_DENIED
   ELSE
   RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = READ_ACCESS_DENIED | OTHER
   | (BACnet-Error-PDU,
   Error Class = OBJECT,
   Error Code = OTHER)
   | (BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = SERVICE_REQUEST_DENIED | OTHER)
   ELSE IF (Y is an array and its value is too large to return given the IUT's APDU and segmentation limitations) THEN
   TRANSMIT ReadProperty-Request,
   'Object Identifier' = X,
   'Property Identifier' = Y
   RECEIVE BACnet-Abort-PDU,
   'Abort Reason' = SEGMENTATION_NOT_SUPPORTED | BUFFER_OVERFLOW
   TRANSMIT ReadProperty-Request,
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Property Array Index' = 0
   RECEIVE ReadProperty-ACK,
   'Object Identifier' = X,
   'PropertyIdentifier' = Y,
   'Array Index' = 0,
   'Property Value' = (N: the number of array elements in Y as indicated in the EPICS)
   REPEAT Z = (1 .. N) DO {
   VERIFY (X), Y = (the value for element Z as indicated in the EPICS), ARRAY INDEX = Z

```
            }
        ELSE
            VERIFY (X), Y = (the value for this property specified in the EPICS)
    }
}
```

Notes to Tester: For cases where the EPICS indicates that the value of a property is unspecified using the "?" symbol, any value that is of the correct datatype shall be considered to be a match.

135.1-2011j-2. Improve the Write Support Test.

Rationale

The existing 7.2.2 write support test does not consider restrictions placed on property value ranges by the object or property definition.

[Change **Clause 7.2.1.1**, p. 33]

7.2.1.1 Enumerated and Boolean Values

For enumerated and Boolean values ~~each~~ *a set of the* defined enumeration*s, including, at a minimum, the lowest, highest, and a mid-range value,* shall be ~~explicitly~~ tested after taking into consideration any permitted restrictions as defined in *Clause* 4.4.2.

[Change **Clause 7.2.1.3**, p. 33]

7.2.1.3 Octetstrings and Characterstrings

Properties with an octetstring or characterstring datatype shall be tested with a string of ~~length zero~~*the minimum supported length*, a string with the maximum supported length, and a string with some length between the two. The vendor shall provide the ~~actual~~ value*s* of the *minimum and* maximum *string* length*s* ~~string~~ in the EPICS. *For string properties that do not have a fixed maximum length, the vendor shall provide a maximum length that is acceptable under normal operating conditions for use in these tests. In such cases, no statement is made as to whether or not a longer string value would or would not be accepted by the property.* See *Clause* 4.4.2.

[Change **Clause 7.2.2**, p. 33]

7.2.2 Write Support Test Procedure

Purpose: To verify that all writable properties of all objects can be written to using BACnet WriteProperty and WritePropertyMultiple services. The test is performed once using WriteProperty and once using WritePropertyMultiple. When writing to array properties, the whole array shall be written without using an array index, where possible.

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

Test Steps:

1. REPEAT X = (all objects in the IUT's database) DO {
   REPEAT Y = (all writable properties in object X) DO {
   REPEAT Z = (~~all~~ *a set of* values meeting the ~~functional~~ range requirements of 7.2.1 *and any additional restrictions placed on the allowable property values as specified by the vendor*) DO {
   WRITE (X), Y = Z,
   VERIFY (X), Y = Z
   }
   }
   }

135.1-2011j-3. Improve the Command Prioritization Test.

Rationale

The existing command prioritization test does not allow for non-binary objects that can only take on 2 values. The specification as to which object types the test applies to has been made generic.

[Change **Clause 7.3.1.3**, p. 37]

7.3.1.3 Command Prioritization Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clause: 19.2.

Purpose: To verify that the command prioritization algorithm is properly implemented. ~~This test applies to Analog Output, Analog Value, Binary Output, Binary Value, Multi-state Output, and Multi-state Value objects that are commandable.~~ *This test applies to all commandable objects.*

Test Concept: The TD selects three different values V_{low}, V_{med}, and V_{high} chosen from the valid values specified in 4.4.2. For ~~binary datatypes~~ *objects that are limited to two values,* V_{low} and V_{high} shall be the same, and V_{med} shall be different. The TD also selects three priorities P_{low}, P_{med}, and P_{high}, all between 1 and 5, such that numerically $P_{low} > P_{med} > P_{high}$. The selected values are written one at a time to Present_Value at the corresponding priority. The Present_Value and Priority_Array are checked to verify correct operation. Priorities numerically smaller than 6 (higher priority) are used to eliminate minimum on/off time considerations.

Configuration Requirements: The object to be tested shall be configured such that all slots in the Priority_Array with a priority higher than 6 have a value of NULL.

Test Steps:

1. WRITE Present_Value = V_{low}, PRIORITY = P_{low}
2. VERIFY Present_Value = V_{low}
3. VERIFY Priority_Array = V_{low}, ARRAY INDEX = P_{low}
4. REPEAT Z = (each index 1 through 5 not equal to P_{low}) DO {
   VERIFY Priority_Array = NULL, ARRAY INDEX = Z
   }
5. WRITE Present_Value = V_{high}, PRIORITY = P_{high}
6. VERIFY Present_Value = V_{high}
7. VERIFY Priority_Array = V_{high}, ARRAY INDEX = P_{high}
8. REPEAT Z = (each index 1 through 5 not equal to P_{low} or P_{high}) DO {
   VERIFY Priority_Array = NULL, ARRAY INDEX = Z
   }
9. WRITE Present_Value = V_{med}, PRIORITY = P_{med}
10. VERIFY Present_Value = V_{high}
11. VERIFY Priority_Array = V_{med}, ARRAY INDEX = P_{med}
12. REPEAT Z = (each index 1 through 5 not equal to P_{low}, P_{med} or P_{high}) DO {
    VERIFY Priority_Array = NULL, ARRAY INDEX = Z
    }
13. WRITE Present_Value = NULL, PRIORITY = P_{high}
14. VERIFY Present_Value = V_{med}
15. REPEAT Z = (each index 1 through 5 not equal to P_{low} or P_{med}) DO {
    VERIFY Priority_Array = NULL, ARRAY INDEX = Z
    }
16. WRITE Present_Value = NULL, PRIORITY = P_{med}
17. VERIFY Present_Value = V_{low}
18. REPEAT Z = (each index 1 through 5 not equal to P_{low}) DO {
    VERIFY Priority_Array = NULL, ARRAY INDEX = Z

}

19. WRITE Present_Value = NULL, PRIORITY = P_{low}
20. REPEAT Z = (each index 1 through 5) DO {
    VERIFY Priority_Array = NULL, ARRAY INDEX = Z
    }

135.1-2011j-4. Clarify the Application of the Event_Enable Test.

Rationale

The existing test does not indicate what to do if the Event_Enable property is read-only and the IUT cannot be configured as specified in the Configuration Requirements. The existing test does not apply to objects that only generate TO_NORMAL transitions so a new test is added.

[Change **Clause 7.3.1.10**, p. 43]
[Note that the test is renumbered to "Clause 7.3.1.10.1" to allow for a second Event_Enable test, added below]

7.3.1.10 Event_Enable Tests

7.3.1.10.1 Event_Enable Test for TO_OFFNORMAL and TO_NORMAL

Dependencies: ConfirmedEventNotification Service Initiation Tests, 8.4; UnconfirmedEventNotification Service Initiation Tests, 8.5; ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

~~BACnet Reference Clauses: 12.1.27, 12.2.23, 12.3.24, 12.4.20, 12.6.22, 12.7.26, 12.8.24, 12.12.10, 12.15.19, 12.16.19, 12.17.34, 12.18.17, 12.19.18, 12.20.18, 12.23.26, and 12.25.22.~~

Purpose: To verify that notification messages are transmitted only if the bit in Event_Enable corresponding to the event transition has a value of TRUE. This test applies to ~~Event Enrollment objects and Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Life Safety Point, Life Safety Zone, Loop, Multi-state Input, Multi-state Output, and Multi-state Value objects that support intrinsic reporting~~ *all event generating object types that are capable of generating TO-OFFNORMAL and TO-NORMAL event transitions*.

Test Concept: The IUT is configured such that the Event_Enable property indicates that some event transitions are to trigger an event notification and some are not. Each event transition is triggered and the IUT is monitored to verify that notification messages are transmitted only for those transitions for which the Event_Enable property has a value of TRUE.

Configuration Requirements: The Event_Enable property shall be configured with a value of TRUE for either the TO-OFFNORMAL transition or the TO-NORMAL transition and the other event transition shall have a value of FALSE. *If the Event_Enable property is not configurable, then follow the test steps as written and verify correct behavior for the value of the Event_Enable property.* For analog objects the Limit_Enable property shall be configured with the value (TRUE, TRUE). The referenced event-triggering property shall be set to a value that results in a NORMAL condition. The value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE).
. . .

[Add **Clause 7.3.1.10.2** , p. 44]

7.3.1.10.2 Event_Enable Tests for TO_NORMAL only Algorithms

Dependencies: ConfirmedEventNotification Service Initiation Tests, 8.4; UnconfirmedEventNotification Service Initiation Tests, 8.5; ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

Purpose: To verify that notification messages are transmitted only if the bit in Event_Enable corresponding to the event transition has a value of TRUE. This test applies to objects that only support generation of TO_NORMAL transitions.

Test Concept: The IUT is configured such that the Event_Enable property indicates that some event transitions are to trigger an event notification and some are not. Each event transition is triggered, and the IUT is monitored to verify that notification messages are transmitted only for those transitions for which the Event_Enable property has a value of TRUE.

Configuration Requirements: In the Notification Class object providing recipient information, the value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE).

1. VERIFY Event_State = NORMAL
2. MAKE (the TO-NORMAL bit of the Event_Enable property equal to TRUE)
3. MAKE (a condition exist that would cause the object to generate a TO_NORMAL transition)
4. 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' = (the current local time),
   '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,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = NORMAL,
   'Event Values' = (values appropriate to the event type)
5. TRANSMIT SimpleAck-PDU
6. VERIFY Event_State = NORMAL
7. IF (Event_Enable can be changed such that the TO-NORMAL transition is FALSE)
   MAKE (the TO-NORMAL bit of the Event_Enable property equal to FALSE)
   MAKE (a condition exist that would cause the object to generate a TO_NORMAL transition)
   CHECK (verify that the IUT did not transmit an event notification message)
8. IF (the event-generating object can be placed into a fault condition) THEN
   IF (Event_Enable can be modified) THEN
   MAKE (Event_Enable TO-FAULT transition equal TRUE)
   IF (Event_Enable TO-FAULT transition = TRUE) THEN
   MAKE (the event-triggering object change to a fault condition)
   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' = (the current local time),
   '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,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = FAULT,
   'Event Values' = (values appropriate to the event type)
   TRANSMIT SimpleAck-PDU
   VERIFY Event_State = FAULT
   MAKE (the event-triggering object change to a normal condition)
   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' = (the current local time),
   '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,
   'AckRequired' = TRUE | FALSE,
   'From State' = FAULT,
   'To State' = NORMAL,
   'Event Values' = (values appropriate to the event type)

TRANSMIT SimpleAck-PDU

9. IF (Event_Enable can be modified) THEN
   MAKE (Event_Enable TO-FAULT transition equal FALSE)
10. IF (Event_Enable TO-FAULT transition = FALSE) THEN
   MAKE (the event-triggering object change to a fault condition)
   VERIFY Event_State = FAULT
   CHECK (verify that the IUT did not transmit an event notification message)
   MAKE (the event-triggering object change to a normal condition)

Notes to Tester: For objects that do not have a Time_Delay property, the Time_Delay value used in the test shall be 0. The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the TD shall skip all of the steps in which a BACnet-SimpleACK-PDU is sent. The 'Message Text' parameter is omitted in the test description because it is optional. The IUT may include this parameter in the notification messages.

135.1-2011j-5. Improve the Limit_Enable Test.

Rationale

The existing test does not properly account for devices in which Limit_Enable is not writable or configurable. The existing test is also incorrect in how it waits for transitions to occur due to Time_Delay.

[Change **Clause 7.3.1.13**, p. 49]

7.3.1.13 Limit_Enable Test

Dependencies: ConfirmedEventNotification Service Initiation Tests, 8.4; UnconfirmedEventNotification Service Initiation Tests, 8.5; ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

~~BACnet Reference Clauses: 12.1.26, 12.2.22, 12.3.23,12.4.19 and 12.23.25.~~

Purpose: To verify that the Limit_Enable property correctly enables or disables reporting of out of range events. This test applies to *objects with a Limit_Enable property.* ~~Accumulator, Analog Input, Analog Output, Analog Value and Pulse Converter objects that support intrinsic reporting~~. ~~If the Limit_Enable property is not writable and cannot be reconfigured this test shall be omitted.~~

Test Concept: The event-triggering property is manipulated to cause both the high limit and the low limit to be exceeded for each possible combination of values for Limit_Enable. The resulting event notification messages are monitored to verify that they are transmitted only for circumstances where the associated event limit is enabled.

Configuration Requirements: ~~The Limit_Enable property shall be configured with the value (TRUE, TRUE). If Limit_Enable is not writable there shall be additional event-generating objects of the same type that have Limit_Enable configured with the values (FALSE, TRUE), (TRUE, FALSE) and (FALSE, FALSE).~~ *Configure the object with High_Limit, Low_Limit, and Deadband values such that High_Limit - Deadband > Low_Limit and both the Low_Limit and High_Limit values are within the valid range of values for Present_Value. If the device cannot be configured with limit values that meet these conditions, then this test shall be skipped. The Event_Enable property shall be set to (TRUE, ?, TRUE) for this test. If the Event_Enable cannot be configured such that the TO-NORMAL and the TO-OFFNORMAL transitions are TRUE, then this test shall be skipped.*

In the test description below, "X" is used to designate the event-triggering property.

Test Steps:

[The test steps have been renumbered without change marking to improve clarity]

~~1. VERIFY Limit_Enable = (TRUE, TRUE)~~
~~2. VERIFY Event_State = NORMAL~~
1. IF Limit_Enable can be made to be equal (TRUE, TRUE) THEN
2. IF Limit_Enable is writable THEN
WRITE Limit_Enable = (TRUE, TRUE)
ELSE
MAKE (Limit_Enable = (TRUE, TRUE))
3. WAIT (Time_Delay + **Notification Fail Time**)
4. VERIFY Event_State = NORMAL
5. IF (X is writable) THEN
WRITE X = (a value that exceeds High_Limit)
ELSE
MAKE (X a value that exceeds High_Limit)
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 object configured for this test),

'Time Stamp' = (the current local time),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the value configured to correspond to a TO-OFFNORMAL transition),
'Event Type' = OUT_OF_RANGE,
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = HIGH_LIMIT,
'Event Values' = (values appropriate to the event type)

8. *TRANSMIT SimpleAck-PDU*
9. IF (X is writable) THEN
   WRITE X = (a value that is lower than Low_Limit)
   ELSE
   MAKE (X a value that is lower than Low_Limit)
10. WAIT (Time_Delay)
11. *BEFORE **Notification Fail Time***
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object configured for this test),
   'Time Stamp' = (the current local time),
   'Notification Class' = (the class corresponding to the object being tested),
   'Priority' = (the value configured to correspond to a TO-NORMAL transition),
   'Event Type' = OUT_OF_RANGE,
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = HIGH_LIMIT,
   'To State' = NORMAL,
   'Event Values' = (values appropriate to the event type)
12. *TRANSMIT SimpleAck-PDU*
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 object configured for this test),
   'Time Stamp' = (the current local time),
   'Notification Class' = (the class corresponding to the object being tested),
   'Priority' = (the value configured to correspond to a TO-OFFNORMAL transition),
   'Event Type' = OUT_OF_RANGE,
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = ~~HIGH_LIMIT~~*NORMAL*,
   'To State' = LOW_LIMIT,
   'Event Values' = (values appropriate to the event type)
15. *TRANSMIT SimpleAck-PDU*
16. *IF (X is writable) THEN*
   WRITE X = (a value that is between Low_Limit + deadband and High_Limit)
   ELSE
   MAKE (X a value that is between than Low_Limit + deadband and High_Limit)
17. *WAIT (Time_Delay)*
18. *BEFORE Notification Fail Time*
   RECEIVE ConfirmdEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object configured for this test),
   'Time Stamp' = (the current local time),
   'Notification Class' = (the class corresponding to the object being tested),

'Priority' = (the value configured to correspond to a TO-NORMAL transition),
'Event Type' = OUT_OF_RANGE,
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = LOW_LIMIT,
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)

19. TRANSMIT SimpleAck-PDU
20. IF Limit_Enable can be made to equal (FALSE, TRUE) THEN
21. IF Limit_Enable is writable THEN
WRITE Limit_Enable = (FALSE, TRUE)
~~11. VERIFY Limit_Enable = (FALSE, TRUE)~~
ELSE
MAKE (Limit_Enable = (FALSE,TRUE))
22. IF (X is writable) THEN
WRITE X = (a value that exceeds High_Limit)
ELSE
MAKE (X a value that exceeds High_Limit)
23. WAIT (Time_Delay)
24. BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object configured for this test),
'Time Stamp' = (the current local time),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the value configured to correspond to a TO-OFFNORMAL transition),
'Event Type' = OUT_OF_RANGE,
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = ~~LOW_LIMIT~~*NORMAL*,
'To State' = HIGH_LIMIT,
'Event Values' = (values appropriate to the event type)
25. TRANSMIT SimpleAck-PDU
26. IF (X is writable) THEN
WRITE X = (a value that is *between Low_Limit and High_Limit-Deadband*~~lower than Low_Limit~~)
ELSE
MAKE (X a value that *between Low_Limit and High_Limit-Deadband*~~is lower than Low_Limit~~)
~~16. WAIT (Time_Delay + **Notification Fail Time**)~~
~~17. CHECK (verify that no notification message was transmitted)~~
~~18. WRITE Limit_Enable = (TRUE, FALSE)~~
~~19. VERIFY Limit_Enable = (TRUE, FALSE)~~
~~20. IF (X is writable) THEN~~
~~WRITE X = (a value that exceeds High_Limit)~~
~~ELSE~~
~~MAKE (X a value that exceeds High_Limit)~~
~~21. WAIT (Time_Delay + **Notification Fail Time**)~~
~~22. CHECK (verify that no notification message was transmitted)~~
~~23. IF (X is writable) THEN~~
~~WRITE X = (a value that is lower than Low_Limit)~~
~~ELSE~~
~~MAKE (X a value that is lower than Low_Limit)~~
27. WAIT (Time_Delay)
28. BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object configured for this test),

'Time Stamp' = (the current local time),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the value configured to correspond to a TO-NORMAL transition),
'Event Type' = OUT_OF_RANGE,
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = HIGH_LIMIT,
'To State' = ~~LOW_LIMIT~~*NORMAL*,
'Event Values' = (values appropriate to the event type)

29. TRANSMIT SimpleAck-PDU

~~26. WRITE Limit_Enable = (FALSE, FALSE)~~
~~27. VERIFY Limit_Enable = (FALSE, FALSE)~~
~~28. IF (X is writable) THEN~~
~~WRITE X = (a value that exceeds High_Limit)~~
~~ELSE~~
~~MAKE (X a value that exceeds High_Limit)~~
~~29. WAIT (Time_Delay + **Notification Fail Time**)~~
~~30. CHECK (verify that no notification message was transmitted)~~

30. IF (X is writable) THEN
WRITE X = (a value that is lower than Low_Limit)
ELSE
MAKE (X a value that is lower than Low_Limit)
31. WAIT (Time_Delay + **Notification Fail Time**)
32. CHECK (verify that no notification message was transmitted)
33. IF (X is writable) THEN
WRITE X = (a value that is between than Low_Limit+Deadband and High_Limit)
ELSE
MAKE (X a value that is lower than Low_Limit+Deadband and High_Limit)
*34. WAIT (Time_Delay + **Notification Fail Time**)*
33. IF Limit_Enable can be made to equal (TRUE, FALSE) THEN
34. IF Limit_Enable is writable THEN
WRITE Limit_Enable = (TRUE, FALSE)
ELSE
MAKE (Limit_Enable = (TRUE, FALSE))
35. IF (X is writable) THEN
WRITE X = (a value that exceeds High_Limit)
ELSE
MAKE (X a value that exceeds High_Limit)
36. WAIT (Time_Delay + Notification Fail Time)
37. CHECK (Verify that no notification message was transmitted)
38. IF (X is writable) THEN
WRITE X = (a value that is lower than Low_Limit)
ELSE
MAKE (X a value that is lower than Low_Limit)
39. WAIT (Time_Delay)
*40. BEFORE **Notification Fail Time***
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object configured for this test),
'Time Stamp' = (the current local time),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the value configured to correspond to a TO-OFFNORMAL transition),
'Event Type' = OUT_OF_RANGE,
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,

'To State' = LOW_LIMIT,
'Event Values' = (values appropriate to the event type)

41. *TRANSMIT SimpleAck-PDU*
42. *IF (X is writable) THEN*
    WRITE X = (a value that is between Low_Limit + Deadband and High_Limit)
    ELSE
    MAKE (X a value that is between Low_Limit + Deadband and High_Limit)
43. *WAIT (Time_Delay)*
44. *BEFORE **Notification Fail Time***
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the object configured for this test),
    'Time Stamp' = (the current local time),
    'Notification Class' = (the class corresponding to the object being tested),
    'Priority' = (the value configured to correspond to a TO-NORMAL transition),
    'Event Type' = OUT_OF_RANGE,
    'Notify Type' = ALARM | EVENT,
    'AckRequired' = TRUE | FALSE,
    'From State' = LOW_LIMIT,
    'To State' = NORMAL,
    'Event Values' = (values appropriate to the event type)
45. *IF Limit_Enable can be made to equal (FALSE, FALSE) THEN*
46. *IF Limit_Enable is writable THEN*
    WRITE Limit_Enable = (FALSE, FALSE)
    ELSE
    MAKE (Limit_Enable = (FALSE, FALSE))
47. *IF (X is writable) THEN*
    WRITE X = (a value that exceeds High_Limit)
    ELSE
    MAKE (X a value that exceeds High_Limit)
48. *WAIT (Time_Delay + Notification Fail Time)*
49. *CHECK (verify that no notification message was transmitted)*
50. *IF (X is writable) THEN*
    WRITE X = (a value that is lower than Low_Limit)
    ELSE
    MAKE (X a value that is lower than Low_Limit)
51. *WAIT (Time_Delay + Notification Fail Time)*
52. *CHECK (verify that no notification message was transmitted)*
53. *IF (X is writable) THEN*
    WRITE X = (a value that is between Low_Limit and High_Limit)
    ELSE
    MAKE (X a value that is between Low_Limit and High_Limit)
54. *WAIT (Time_Delay + Notification Fail Time)*
55. *CHECK (verify that no notification message was transmitted)*

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service *in which case the TD shall skip all of the steps in which a BACnet-SimpleACK-PDU is sent.* The 'Message Text' parameter is omitted in the test description because it is optional. The IUT may include this parameter in the notification messages. ~~If the Limit_Enable property is not writable the equivalent tests may be applied to separate or reconfigured objects that have the appropriate value for Limit_Enable.~~

135.1-2011j-6.Update the Calendar Test.

Rationale

The existing calendar tests incorrectly take UTC_Offset into account when using the TimeSynchronization service and do not allow for use of the UTCTimeSynchronization service.

The WeekNDay test is also updated to allow the tester to select the dates and times for testing and is improved by resetting the device back to the initial time before each test section.

[Change **Clause 7.3.2.8.1**, p. 60]

7.3.2.8.1 Single Date Rollover Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30*; UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.9.

Purpose: To verify the ability to represent the Calendar status when the Date_List is in the form of an individual date. Either execution of the TimeSynchronization *or the UTCTimeSynchronization* service must be supported or another means must be supplied to reset the IUT's clock during the test.

Test Concept: The Calendar object is configured with a Date_List containing a single date. The IUT's clock is set to the date that immediately precedes the one specified in Date_List and a time near the end of the day. The test verifies that the Present_Value of the Calendar object is initially FALSE and that as the time rolls over to the next day the Present_Value changes to TRUE.

Configuration Requirements: The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a Date.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (the day preceding the one specified in Date_List,
   24:00:00 ~~+ UTC_Offset~~ – **Schedule Evaluation Fail Time** – 1 minute)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (the day preceding the one specified in Date_List,
   24:00:00 – ***Schedule Evaluation Fail Time*** *– 1 minute converted to UTC)) |*
   MAKE (the local time = 24:00:00 – **Schedule Evaluation Fail Time** – 1 minute)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = FALSE
4. WAIT (**Schedule Evaluation Fail Time** + 2 minutes)
5. VERIFY Present_Value = TRUE

[Change **Clause 7.3.2.8.2**, p. 61]

7.3.2.8.2 Date Range Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30*; UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.9.

Purpose: To verify the ability to represent the Calendar status when the Date_List is in the form of a BACnetDateRange. Either execution of the TimeSynchronization *or the UTCTimeSynchronization* service must be supported or another means must be supplied to reset the IUT's clock during the test.

Test Concept: The Calendar object is configured with a Date_List containing a single BACnetDateRange. The IUT's clock is set to a time and date that is outside of the date range. The Present_Value is read and verified to be FALSE. The clock is reset to a value within the date range and the Present_Value is read again to verify that it has the value TRUE. If the IUT can be configured with wildcard fields in the date range then it shall be tested with and without wildcards.

Configuration Requirements: The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a BACnetDateRange.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any day and time outside of the specified date range selected by the tester)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (any day and time outside of the specified date range selected by the tester)) |
   MAKE (the local time = any day and time outside of the specified date range selected by the tester)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = FALSE
4. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any day and time inside the specified date range selected by the tester)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (any day and time inside the specified date range selected by the tester)) |
   MAKE (the local time = any day and time inside the specified date range selected by the tester)
5. WAIT **Schedule Evaluation Fail Time**
6. VERIFY Present_Value = TRUE

[Change **Clause 7.3.2.8.3** , p. 61]

7.3.2.8.3 WeekNDay Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30*; UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.9.

Purpose: To verify the ability to represent the Calendar status when the Date_List is in the form of a BACnetWeekNDay. Either execution of the TimeSynchronization *or the UTCTimeSynchronization* service must be supported or another means must be supplied to reset the IUT's clock during the test.

Test Concept: The Calendar object is configured with a Date_List containing a single BACnetWeekNDay. The IUT's clock is set to a time and date*,* T_1*,* that matches the BACnetWeekNDay mask. The Present_Value is read and verified to be TRUE. The clock is reset to a value*,* T_2*,* that matches the BACnetWeekNDay mask except for the month. The Present_Value is read and verified to be FALSE. The clock is reset again to a value*,* T_3*,* that matches the BACnetWeekNDay mask except for the week of the month. The Present_Value is read and verified to be FALSE. The clock is reset again to a value*,* T_4*,* that matches the BACnetWeekNDay mask except for the day of the week. The Present_Value is read and verified to be FALSE. *In between each change, the clock is reset to* T_1 *to force the Present_Value back to TRUE.*

Configuration Requirements: The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a BACnetWeekNDay. The BACnetWeekNDay shall be the 11th month, last seven days, and Saturday.

Test Steps:

[Note: test steps are renumbered without change marking for clarity.]

1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (T_1~~24-November-2001, 13:00:00 + UTC_Offset~~)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (T_1)) |
   MAKE (the local time = T_1~~24-November-2001 at 13:00:00~~)

2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = TRUE
4. (TRANSMIT TimeSynchronization-Request,
   'Time' = (T_2~~27-October-2001, 13:00:00 + UTC_Offset~~) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (T_2)) |
   MAKE (the local time = T_2~~27-October-2001 at 13:00:00~~)
5. WAIT **Schedule Evaluation Fail Time**
6. VERIFY Present_Value = FALSE
7. *(TRANSMIT TimeSynchronization-Request,*
   'Time' = (T_1)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (T_1)) |
   MAKE (the local time = T_1)
8. *WAIT **Schedule Evaluation Fail Time***
9. *VERIFY Present_Value = TRUE*
10. (TRANSMIT TimeSynchronization-Request,
    'Time' = (T_3~~17-November-2001, 13:00:00 + UTC_Offset~~) |
    (TRANSMIT UTCTimeSynchronization-Request,
    'Time' = (T_3)) |
    MAKE (the local time = T_3~~17-November-2001 at 13:00:00~~)
11. WAIT **Schedule Evaluation Fail Time**
12. VERIFY Present_Value = FALSE
13. *(TRANSMIT TimeSynchronization-Request,*
    'Time' = (T_1)) |
    (TRANSMIT UTCTimeSynchronization-Request,
    'Time' = (T_1)) |
    MAKE (the local time = T_1)
14. *WAIT **Schedule Evaluation Fail Time***
15. *VERIFY Present_Value = TRUE*
16. (TRANSMIT TimeSynchronization-Request,
    'Time' = (T_4~~25-November-2001, 13:00:00 + UTC_Offset~~) |
    (TRANSMIT UTCTimeSynchronization-Request,
    'Time' = (T_4)) |
    MAKE (the local time = T_4~~25-November-2001 at 13:00:00~~)
17. WAIT **Schedule Evaluation Fail Time**
18. VERIFY Present_Value = FALSE

135.1-2011j-7. Update Notification Class Tests to use UTCTimeSynchronization.

Rationale

The existing ValidDays Test and FromTime and ToTime tests do not allow for the use of the UTCTimeSynchronization service.

[Change **Clause 7.3.2.21.3.1** , p. 85]

7.3.2.21.3.1 ValidDays Test

Dependencies: ConfirmedEventNotification Service Initiation Tests, 8.4; UnconfirmedEventNotification Service Initiation Tests, 8.5; ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22; TimeSynchronization Service Execution Tests, 9.30*; UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.21.8.

Purpose: To verify the operation of the Valid Days parameter of a BACnetDestination as used in the Recipient_List property of the Notification Class object.

Test Concept: The TD will select one instance of the Notification Class object and one instance of an event-generating object that is linked to it. The Recipient_List of the Notification Class object shall contain a single recipient with the Valid Days parameter configured so that at least one day is TRUE and at least one day is FALSE. The properties of the event-generating object will be manipulated to cause the Event_State to change from NORMAL to OFFNORMAL. The tester verifies that if the local date is one of the valid days a notification message is transmitted and if local date is not a valid day then no notification message is transmitted.

Configuration Requirements: The IUT shall be configured with one or more instance of the Notification Class object and at least one event-generating object that is linked to the Notification Class object. The event-generating object may be any object that supports intrinsic reporting or it may be an Event Enrollment object. The event-generating object shall have the Event_Enable property configured to transmit notification messages for all event transitions. The event-generating object shall be configured to be in a NORMAL event state at the start of the test. The Notification Class object shall be configured with a single recipient in the Recipient_List. The Valid Days parameter shall be configured so that at least one day of the week has a value of TRUE and at least one day of the week has a value of FALSE. The Transitions parameter shall be configured for the recipient to receive notifications for all event transitions.

In the test description below, “X” is used to designate the event-triggering property.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any time within the window defined by From Time and To Time in the BACnet Destination that corresponds to one of the valid days)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days)) |
   MAKE (the local date and time = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days))
2. WAIT (Time_Delay + **Notification Fail Time**)
3. VERIFY Event_State = NORMAL
4. IF (X is writable) THEN
   WRITE X = (a value that is OFFNORMAL)
   ELSE
   MAKE (X have a value that is OFFNORMAL)
5. WAIT (Time_Delay)
6. 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' = (the current local time),
'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,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = OFFNORMAL,
'Event Values' = (values appropriate to the event type)

7. *TRANSMIT BACnet-SimpleACK-PDU*
8. VERIFY Event_State = OFFNORMAL
9. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any time within the window defined by From Time and To time in the BACnet Destination that corresponds to one of the invalid days)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   Time' = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the invalid days)) |
   MAKE (the local date and time = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the invalid days))
10. IF (X is writable) THEN
    WRITE X = (a value that is NORMAL)
    ELSE
    MAKE (X have a value that is NORMAL)
11. WAIT (Time_Delay + **Notification Fail Time**)
12. CHECK (verify that no notification message was transmitted)

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service*, in which case the TD shall skip all of the steps in which a BACnet-SImpleACK-PDU is sent*. The 'Message Text' parameter is omitted in the test description because it is optional. The IUT may include this parameter in the notification messages.

[Change **Clause 7.3.2.21.3.2**, p. 86]

7.3.2.21.3.2 FromTime and ToTime Test

Dependencies: ValidDays Test, 7.3.2.21.3.1; ConfirmedEventNotification Service Initiation Tests, 8.4; UnconfirmedEventNotification Service Initiation Tests, 8.5; ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30*; UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.21.8.

Purpose: To verify the operation of the From Time and To Time parameters of a BACnetDestination as used in the Recipient_List property of the Notification Class object.

Test Concept: The case where the local date and time fall within the window defined by the From Time and To Time parameters is covered by the ValidDays test in 7.3.2.21.3.1. This test uses the same IUT configuration and sets the local time to a value that is one of the ValidDays but outside of the window defined by the From Time and To Time parameters. The objective is to verify that an event notification message is not transmitted when the event is triggered.

Configuration Requirements: The configuration requirements are identical to the requirements in 7.3.2.21.3.1.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any time outside the window defined by From Time and To Time in the BACnet Destination that corresponds to one of the valid days)) |
   (TRANSMIT UTCTimeSynchronization-Request,

'Time' = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days) |

MAKE (the local date and time = (any time outside the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days))

2. WAIT (Time_Delay + **Notification Fail Time**)
3. VERIFY Event_State = NORMAL
4. IF (X is writable) THEN
   WRITE X = (a value that is OFFNORMAL)
   ELSE
   MAKE (X have a value that is OFFNORMAL)
5. WAIT (Time_Delay + **Notification Fail Time**)
6. CHECK (verify that no notification message was transmitted)

135.1-2011j-8. Update Schedule Tests to use UTCTimeSynchronization.

Rationale

A number of existing Schedule and Calendar tests do not allow for the use of the UTCTimeSynchronization service.

Also, the final step of Clause 7.3.2.23.1, the Effective_Period Test, is corrected to check for the correct value.

[Change **Clause 7.3.2.23.1**, p. 91]

7.3.2.23.1 Effective_Period Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31.*

BACnet Reference Clause: 12.24.6.

Purpose: To verify that Effective_Period controls the range of dates during which the Schedule object is active.

Test Concept: Two Date values are chosen by the TD based on the criteria in Table 7-1 such that one is outside of the Effective_Period and the other corresponds to a known scheduled state inside the Effective_Period. The IUT's local date and time are changed between these dates and the Present_Value property is monitored to verify that write operations occur only within the Effective_Period.

Configuration Requirements: The IUT shall be configured with a schedule object such that the time periods defined in Table 7-1 have uniquely scheduled values. The local date and time shall be set such that the Present_Value property has a value other than V_1.

Table 7-1. Criteria for Effective_Period Test Dates and Values

Date:	Criteria:	Value:
D_1	1. Date occurs during Effective_Period, and 2. Date is active either in Weekly_Schedule or Exception_Schedule.	V_1
D_2	1. Date does not occur during Effective_Period, and 2. Date is active either in Weekly_Schedule or Exception_Schedule.	V_2 different from V_1.

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1)
   | (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1)
   | MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2)
   | (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2)
   | MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = ~~V_2~~ *V_1*

[Change **Clause 7.3.2.23.2**, p. 92]

7.3.2.23.2 Weekly_Schedule Property Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31.*

BACnet Reference Clause: 12.24.7.

Purpose: To verify that Weekly_Schedule contains distinguishable schedules for each day of the week, and that a day's entire schedule can be executed.

Test Concept: The IUT's local date and time are changed sequentially to represent each day of the week as shown in Table 7-2. The Present_Value property is monitored to verify that write operations occur for each separately scheduled day.

Configuration Requirements: The IUT shall be configured with a schedule object containing a weekly schedule with seven distinguishable daily schedules meeting the requirements of Table 7-2. The local date and time shall be set such that the Present_Value property has a value other than V_1. If no schedule exists that meets these requirements and none can be configured, *then* this test shall be omitted.

Table 7-2. Criteria for Weekly_Schedule Test Dates and Values

Date:	**Criteria:**	**Value:**
D_1	1. Date occurs during Effective_Period, 2. Date occurs on a Monday, and 3. Date is not active in Exception_Schedule.	V_1
D_2	1. Date occurs during Effective_Period, 2. Date occurs on a Tuesday, and 3. Date is not active in Exception_Schedule.	V_2 is different from V_1.
D_3	1. Date occurs during Effective_Period, 2. Date occurs on a Wednesday, and 3. Date is not active in Exception_Schedule.	V_3 is different from V_2.
D_4	1. Date occurs during Effective_Period, 2. Date occurs on a Thursday, and 3. Date is not active in Exception_Schedule.	V_4 is different from V_3.
D_5	1. Date occurs during Effective_Period, 2. Date occurs on a Friday, and 3. Date is not active in Exception_Schedule.	V_5 is different from V_4.
D_6	1. Date occurs during Effective_Period, 2. Date occurs on a Saturday, and 3. Date is not active in Exception_Schedule.	V_6 is different from V_5.
D_7	1. Date occurs during Effective_Period, 2. Date occurs on a Sunday, and 3. Date is not active in Exception_Schedule.	V_7 is different from V_6.

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = V_2
8. (TRANSMIT TimeSynchronization-Request, 'Time' = D_3) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_3) |
   MAKE (the local date and time = D_3)
9. WAIT **Schedule Evaluation Fail Time**
10. VERIFY Present_Value = V_3
11. (TRANSMIT TimeSynchronization-Request, 'Time' = D_4) |
    (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_4) |
    MAKE (the local date and time = D_4)
12. WAIT **Schedule Evaluation Fail Time**
13. VERIFY Present_Value = V_4

14. (TRANSMIT TimeSynchronization-Request, 'Time' = D_5) |
    (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_5) |
    MAKE (the local date and time = D_5)
15. WAIT **Schedule Evaluation Fail Time**
16. VERIFY Present_Value = V_5
17. (TRANSMIT TimeSynchronization-Request, 'Time' = D_6) |
    (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_6) |
    MAKE (the local date and time = D_6)
18. WAIT **Schedule Evaluation Fail Time**
19. VERIFY Present_Value = V_6
20. (TRANSMIT TimeSynchronization-Request, 'Time' = D_7) |
    (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_7) |
    MAKE (the local date and time = D_7)
21. WAIT **Schedule Evaluation Fail Time**
22. VERIFY Present_Value = V_7
23. REPEAT X = (the time portion of the BACnetTimeValue entries for one of the daily schedules in Table 7-2) DO {
    (TRANSMIT TimeSynchronization-Request, 'Time' = X) |
    (TRANSMIT UTCTimeSynchronization-Request, 'Time' = X) |
    MAKE (the local date and time = X)
    WAIT Schedule Evaluation Fail Time
    VERIFY Present_Value = (the scheduled value corresponding to time X)
    }

[Change **Clause 7.3.2.23.3.2**, p. 94]

7.3.2.23.3.2 Calendar Entry Date Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31.*

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a specified date appearing in an Exception_Schedule enables the referencing Schedule object.

Test Concept: The IUT's local date and time are changed to values that are selected by the TD based on the criteria in Table 7-3. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a specific date. The criteria for the dates used in the test are given in Table 7-3. The local date and time shall be set such that the Present_Value property has a value other than V_1. This test shall not be performed if the Protocol_Revision property is present in the Device object and has a value of 4 or greater.

Table 7-3. Criteria for Calendar Entry Date Test Dates and Values

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: Date, 2B. Date matches calendarEntry: Date, and 2C. Higher eventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, and 2. Date does not appear in any BACnetSpecialEvents, and 3. Date occurs on the same date as, but with time following, an entry in a BACnetDailtySchedule in the referencing Schedule object.	Other than V_1

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |

(TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) /
MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
(TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) /
MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = (any value other than V_1)

[Change **Clause 7.3.2.23.3.3**, p. 94]

7.3.2.23.3.3 Calendar Entry DateRange Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a date appearing in an Exception_Schedule's date range enables the referencing Schedule object.

Test Concept: The IUT's local date and time are changed to values that are selected by the TD based on the criteria in Table 7-4. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a date range. The criteria for the dates used in the test are given in Table 7-4. The local date and time shall be set such that the Present_Value property has a value other than V_1. This test shall not be performed if the Protocol_Revision property is present in the Device object and has a value of 4 or greater.

Table 7-4. Criteria for Calendar Entry DateRange Test Dates and Values

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: DateRange, 2B. Date matches BACnetCalendarEntry: DateRange, and 2C. Higher eventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, and 2. Date does not appear in any BACnetSpecialEvents, and 3. Date occurs on the same date as, but with time following, an entry in a BACnetDailtySchedule in the referencing Schedule object.	Other than V_1

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
(TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) /
MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
(TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) /
MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = (any value other than V_1)

[Change **Clause 7.3.2.23.3.4**, p. 95]

7.3.2.23.3.4 Calendar Entry WeekNDay Month Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a date matching a WeekNDay's Month field in an Exception_Schedule enables the referencing Schedule object.

Test Concept: The IUT's local date and time are changed to values that are selected by the TD based on the criteria in Table 7-5. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a WeekNDay entry specifying a month. The criteria for the dates used in the test are given in Table 7-5. The local date and time shall be set such that the Present_Value property has a value other than V_1. This test shall not be performed if the Protocol_Revision property is present in the Device object and has a value of 4 or greater.

Table 7-5. Criteria for Calendar Entry WeekNDay Month Test Dates and Values

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay: specifies Month, 2C. Date matches calendarEntry: WeekNDay: Month, and 2.D. Higher eventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, and 2. Date does not appear in any BACnetSpecialEvents, and 3. Date occurs on the same date as, but with time following, an entry in a BACnetDailtySchedule in the referencing Schedule object.	Other than V_1

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = (any value other than V_1)

[Change **Clause 7.3.2.23.3.5**, p. 96]

7.3.2.23.3.5 Calendar Entry WeekNDay Week Of Month Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a date matching a WeekNDay's WeekOfMonth field in an Exception_Schedule enables the referencing Schedule object.

Test Concept: The IUT's local date and time are changed to values that are selected by the TD based on the criteria in Table 7-6. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a WeekNDay entry specifying a week of the month. The criteria for the dates used in the test are given in Table 7-6. The local date and time shall be set such that the Present_Value property has a

value other than V_1. This test shall not be performed if the Protocol_Revision property is present in the Device object and has a value of 4 or greater.

Table 7-6. Criteria for Calendar Entry WeekNDay Week Of Month Test Dates and Values

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies WeekOfMonth, 2C. calendarEntry: WeekNDay: WeekOfMonth is in the range 1..5, 2D. Date matches calendarEntry: WeekNDay: WeekOfMonth, and 2E Higher eventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies WeekOfMonth, 2C. calendarEntry: WeekNDay: WeekOfMonth is in the range 1..5, and 2D. Date does not match calendarEntry: WeekNDay: WeekOfMonth.	Other than V_1

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) /
   MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) /
   MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = (any value other than V_1)

[Change **Clause 7.3.2.23.3.6**, p. 96]

7.3.2.23.3.6 Calendar Entry WeekNDay Last Week Of Month Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a date matching a WeekNDay's WeekOfMonth field in an Exception_Schedule enables the referencing Schedule object.

Test Concept: The IUT's local date and time are changed to values that are selected by the TD based on the criteria in Table 7-7. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a WeekNDay entry specifying the last week of the month. The criteria for the dates used in the test are given in Table 7-7. The local date and time shall be set such that the Present_Value property has a value other than V_1. This test shall not be performed if the Protocol_Revision property is present in the Device object and has a value of 4 or greater.

Table 7-7. Criteria for Calendar Entry WeekNDay Last Week Of Month Test Dates and Values

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies WeekOfMonth, 2C. calendarEntry: WeekNDay: WeekOfMonth has the value 6, 2D. Date is in the last week of the month, and 2E. Higher eventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies WeekOfMonth, 2C. calendarEntry: WeekNDay: WeekOfMonth has the value 6, and 2D. Date is not in the last week of the month.	Other than V_1

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = (any value other than V_1)

[Change **Clause 7.3.2.23.3.7**, p. 97]

7.3.2.23.3.7 Calendar Entry WeekNDay Day Of Week Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a date matching a WeekNDay's DayOfWeek field in an Exception_Schedule enables the referencing Schedule object.

Test Concept: The IUT's local date and time are changed to values that are selected by the TD based on the criteria in Table 7-8. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a WeekNDay entry specifying the day of the week. The criteria for the dates used in the test are given in Table 7-8. The local date and time shall be set such that the Present_Value property has a value other than V_1. This test shall not be performed if the Protocol_Revision property is present in the Device object and has a value of 4 or greater.

Table 7-8. Criteria for Calendar Entry WeekNDay Day of Week Test Dates and Values

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies only DayOfWeek, 2C. Date falls on the specified day of the week, and 2D. Higher eventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies only DayOfWeek, and 2C. Date does not fall on the specified day of the week.	

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) /
   MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) /
   MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = (any value other than V_1)

[Change **Clause 7.3.2.23.3.8**, p. 98]

7.3.2.23.3.8 Event Priority Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a BACnetSpecialEvent of a higher priority takes precedence over one of lower priority when both specify the same date.

Configuration Requirements: The IUT shall be configured with a Schedule object containing two or more BACnetSpecialEvents, all active on the same date, with different eventPriority values, with distinguishable BACnetTimeValue entries. If possible all BACnetSpecialEvents shall have a BACnetTimeValue entry with identical time but different values. In the test description D_1 represents a date and time where all of the special events are active.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) /
   MAKE (the local date and time = D_1)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = (the value corresponding to the special event with the highest eventPriority)

[Change **Clause 7.3.2.23.3.9**, p. 98]

7.3.2.23.3.9 List of BACnetTimeValue Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a Special_Event's entire schedule can be executed.

Test Concept: A special event is scheduled that contains multiple BACnetTimeValue entries. The local date and time are changed to values that match each of the BACnetTimeValue entries and the Present_Value property is read to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured with a Schedule object containing a BACnetSpecialEvent with two or more BACnetTimeValue entries and no BACnetSpecialEvents with a higher priority. Each BACnetTimeValue entry shall have a distinguishable value.

Test Steps:

1. REPEAT D_i = (the times used in the BACnetTimeValue pairs of the special event) DO {
   (TRANSMIT TimeSynchronization-Request, 'Time' = D_i) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_i) |
   MAKE (the local date and time = D_i)
   WAIT **Schedule Evaluation Fail Time**
   VERIFY Present_Value = (the value corresponding to the special event with the highest eventPriority)
   }

[Change **Clause 7.3.2.23.4**, p. 99]

7.3.2.23.4 Weekly_Schedule and Exception_Schedule Interaction Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clauses: 12.24.7, 12.24.8.

Purpose: To verify that an Exception_Schedule takes precedent over a coincident BACnetDailySchedule.

Test Concept: The IUT is configured with a Weekly_Schedule and an Exception_Schedule that apply to the same time. The local date and time are changed to the time when the Exception_Schedule is supposed to take control and the Present_Value is read to verify that the scheduled write operation occurs. The local date and time are changed again to a value that would cause another change if the Weekly_Schedule were in control. The Present_Value is read to verify the Exception_Schedule is still controlling.

Configuration Requirements: The IUT shall be configured with a Schedule object containing a Weekly_Schedule and an Exception_Schedule that apply to the same dates. The BACnetSpecialEvents in the Exception_Schedule shall have a higher EventPriority than any other coincident BACnetSpecialEvent. The BACnetTimeValue pairs shall be assigned values such that the values written by the Weekly_Schedule are distinguishable from the values written by the Exception_Schedule. Let D_1 represent the date and time when the Exception_Schedule is configured to take control and write value V_1. There shall be at least one BACnetTimeValue pair in the Weekly_Schedule that specifies a time, D_2, that is after D_1 but before the Exception_Schedule expires. The Weekly_Schedule is configured to write value V_2 at time D_2.

~~For BACnet implementations with a Protocol_Revision of 4 or higher, the date D_2 shall be chosen to occur between D_1 and any entry in the Exception schedule that schedules a NULL value.~~

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = V_1
4. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
5. WAIT **Schedule Evaluation Fail Time**
6. VERIFY Present_Value = V_1

[Change **Clause 7.3.2.23.5**, p. 99]

7.3.2.23.5 Exception_Schedule Restoration Test

Dependencies: ReadProperty Service Execution Tests, 9.18; ReinitializeDevice Service Execution Tests, 9.27; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clauses: 12.24.4, 12.24.7, 12.24.8, 12.24.9.

Purpose: To verify the restoration behavior in an Exception_Schedule.

Test Concept: The IUT is configured with a Schedule object containing an Exception_Schedule with BACnetTimeValue entries that do not include the time 00:00. The local date and time are changed to a value between 00:00 and the first entry in the Exception_Schedule. Present_Value is read to verify that it contains the Schedule_Default value, or V_{last} for implementations with a Protocol_Revision less than 4. The IUT is reset and the Present_Value is checked again to verify that it contains the Schedule_Default value, or V_{last} for implementations with a Protocol_Revision less than 4.

Configuration Requirements: The IUT shall be configured with a Schedule object that contains an Exception_Schedule that has more than one scheduled write operation for a particular day and the first scheduled write is scheduled to occur before the first entry in the corresponding Weekly_Schedule entry. None of the write operations shall be scheduled for time 00:00 and there shall be no higher priority coincident BACnetSpecialEvents. In the test description D_1 represents a time between 00:00 on the day the Exception_Schedule is active and the time of the first schedule write operation in the BACnetSpecialEvent. V_{last} represents the value that is scheduled to be written in the last BACnetTimeValue pair for the day. ~~This test shall not be performed if the Protocol_Revision property is present in the Device object and has a value of 4 or greater.~~

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
2. WAIT **Schedule Evaluation Fail Time**
3. IF (Protocol_Revision is present and Protocol_Revision • 4) THEN
   VERIFY Present_Value = Schedule_Default
   ELSE
   VERIFY Present_Value = V_{last}
4. IF (ReinitializeDevice execution is supported) THEN
   TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = COLDSTART,
   'Password' = (any valid password)
   RECEIVE BACnet-SimpleACK-PDU
   ELSE
   MAKE (the IUT reinitialize)
5. CHECK (Did the IUT perform a COLDSTART reboot?)
6. WAIT **Schedule Evaluation Fail Time**
7. IF (Protocol_Revision is present and Protocol_Revision • 4) THEN
   VERIFY Present_Value = Schedule_Default
   ELSE
   VERIFY Present_Value = V_{last}

[Change **Clause 7.3.2.23.6**, p. 100]

7.3.2.23.6 Weekly_Schedule Restoration Test

Dependencies: ReadProperty Service Execution Tests, 9.18; ReinitializeDevice Service Execution Tests, 9.27; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.4, 12.24.7, 12.24.9.

Purpose: To verify the restoration behavior in a Weekly_Schedule.

Test Concept: The IUT is configured with a Schedule object containing a Weekly_Schedule with a BACnetDailySchedule that has multiple BACnetTimeValue entries that do not include the time 00:00. There shall be no Exception_Schedule that overrides this Weekly_Schedule during the date and time used for this test. The local date and time are changed to a value between 00:00 and the first entry in the BACnetDailySchedule. Present_Value is read to verify that it contains the Schedule_Default value, or V_{last} for implementations with a Protocol_Revision less than 4. The IUT is reset and the Present_Value is checked again to verify that it contains the Schedule_Default value, or V_{last} for implementations with a Protocol_Revision less than 4.

Configuration Requirements: The IUT shall be configured with a Schedule object that contains a Weekly_Schedule that has more than one scheduled write operation for a particular day. None of the write operations shall be scheduled for time 00:00 and there shall be no higher priority coincident BACnetSpecialEvents. In the test description D_1 represents a time between 00:00 and the time of the first scheduled write operation in the BACnetDailySchedule. V_{last} represents the value that is scheduled to be written in the last BACnetTimeValue pair for the day. This test shall not be performed if the Protocol_Revision property is present in the Device object and has a value of 4 or greater.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
2. WAIT **Schedule Evaluation Fail Time**
3. IF (Protocol_Revision is present and Protocol_Revision • 4) THEN
   VERIFY Present_Value = Schedule_Default
   ELSE
   VERIFY Present_Value = V_{last}
4. IF (ReinitializeDevice execution is supported) THEN
   TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = COLDSTART,
   'Password' = (any valid password)
   RECEIVE BACnet-SimpleACK-PDU
   ELSE
   MAKE (the IUT reinitialize)
5. CHECK (Did the IUT perform a COLDSTART reboot?)
6. WAIT **Schedule Evaluation Fail Time**
7. IF (Protocol_Revision is present and Protocol_Revision • 4) THEN
   VERIFY Present_Value = Schedule_Default
   ELSE
   VERIFY Present_Value = V_{last}

[Change **Clause 7.3.2.23.7**, p. 101]

7.3.2.23.7 List_Of_Object_Property_Reference Internal Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31.*

BACnet Reference Clause: 12.24.10.

Purpose: To verify that the Schedule object writes to objects and properties contained within the IUT.

Test Concept: The Schedule object is configured to write to a property of another object within the same device. The IUT's clock is then set to a time between a pair of scheduled write operations, and verification of the first write operation's data value is performed. The time is advanced to the second time, the Schedule object's Present_Value is checked, and verifications of the write operations are performed. If the IUT does not support writing to object properties within the IUT, then this test shall not be performed.

Configuration Requirements: The IUT is configured with a Schedule object containing a List_Of_Object_Property_References property that references, if possible, at least one property in another object within the IUT. The Schedule

object is configured with either a Weekly_Schedule or an active Exception_Schedule, during a period where Effective_Period is active, with at least two consecutive entries with distinguishable values in the List of BACnetTimeValues, and with no Exception_Schedules at a higher priority. D_1 represents the date and time of the first of these two BACnetTimeValues, with corresponding value V_1, while D_2 and V_2 (a value distinguishable from V_1) represent the second BACnetTimeValue. A time D_t is defined to occur between D_1 and D_2.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = D_t) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_t) |
   MAKE (the local date and time = D_t)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = V_1
4. VERIFY (value of referenced property in IUT) = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = V_2
8. VERIFY (value of referenced property in IUT) = V_2

[Change **Clause 7.3.2.23.8**, p. 101]

7.3.2.23.8 List_Of_Object_Property_Reference External Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; *UTCTimeSynchronization Service Execution Tests, 9.31*.

BACnet Reference Clause: 12.24.10.

Purpose: To verify that the Schedule object writes to object properties contained in a device other than the IUT.

Test Concept: The Schedule object is configured to write to a property of another object in the same device and a property of an object in the TD. The IUT's clock is then set to a time between a pair of scheduled write operations, and verification of the first write operation's data value is performed. The time is advanced to the second time, the Schedule object's Present_Value is checked, and verifications of the write operation are performed. If the IUT does not support writes to object properties contained in a device other than the IUT, then this test shall not be performed.

Configuration Requirements: The TD is configured to indicate that it supports the WriteProperty-Request service but not WritePropertyMultiple-Request. The IUT is configured with a Schedule object containing a List_Of_Object_Property_References property that references a property of an object contained in the TD. The Schedule object is configured with either a Weekly_Schedule or an active Exception_Schedule, during a period where Effective_Period is active, with at least two consecutive entries with distinguishable values in the List of BACnetTimeValues, and with no Exception_Schedules at a higher priority. D_1 represents the date and time of the first of these two BACnetTimeValues, with corresponding value V_1, while D_2 and V_2 (a value distinguishable from V_1) represent the second BACnetTimeValue. A time D_t is defined to occur between D_1 and D_2.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = D_t) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_t) |
   MAKE (the local date and time = D_t)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = V_1
4. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
5. BEFORE **Schedule Evaluation Fail Time**
   RECEIVE WriteProperty-Request,
   'Object Identifier' = (the referenced object in the TD),

'Property Identifier' = (the referenced property in the TD),
'Property Value' = V_2

6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = V_2

135.1-2011j-9. Add Protocol Revision 4 Schedule Object Tests.

Rationale

A number of tests were added for Protocol Revision schedules in Addendum 135.1-2009g. This section adds more.

[Add new **Clauses 7.3.2.23.10.3.7** through **7.3.2.23.10.3.13**, p. 110]

7.3.2.23.10.3.7 Revision 4 Calendar Entry WeekNDay Day Of Week Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a date matching a WeekNDay's DayOfWeek field in an Exception_Schedule enables the referencing Schedule object.

Test Concept: The IUT's local date and time are changed to values that are selected by the TD based on the criteria in Table 7-16.1. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a WeekNDay entry specifying the day of the week. The criteria for the dates used in the test are given in Table 7-16.1. The local date and time shall be set such that the Present_Value property has a value other than V_1.

Table 7-16.1. Criteria for Calendar Entry WeekNDay Day of Week Test Dates and Values

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies only DayOfWeek, 2C. Date falls on the specified day of the week, and 2D. Higher EventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies only DayOfWeek, and 2C. Date does not fall on the specified day of the week.	V_2

Test Steps:

1. VERIFY Present_Value = (any value other than V_1)
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2)
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2)
   MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = (any value other than V_2)

7.3.2.23.10.3.8 Revision 4 Event Priority Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a BACnetSpecialEvent of a higher priority takes precedence over one of lower priority when both are active at the same time, and that it relinquishes to the lower priority.

Configuration Requirements: The IUT shall be configured with a Schedule object containing two or more BACnetSpecialEvents, all active on the same date, with different EventPriority values (if possible, all 16 priority levels should be represented), and with overlapping BACnetTimeValue entries distributed thus: the entry with the lowest priority shall have the earliest time-value pair (D_1) with a non-NULL value, and the last time-value pair (D_N) with a NULL value; the next higher priority shall have a time-value pair D_2 occurring after D_1 with a different non-NULL value, and a time-value pair D_{N-1} with a NULL value and occurring before D_N; and so on. The result is that the time-value pairs shall be ordered chronologically thus: D_1, D_2, D_3, ..., D_{N-1}, D_N. An example of such a configuration testing five priority levels is shown in Table 7-16.2.

Table 7-16.2. Example of event and value prioritization

Event	Time:								
Priority:	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
1	-	-	-	-	V_5	NULL	-	-	-
2	-	-	-	V_4	-	-	NULL	-	-
3	-	-	V_3	-	-	-	-	NULL	-
4	-	V_2	-	-	-	-	-	-	NULL
5	V_1	-	-	-	-	-	-	-	-
Present_Value:	V_1	V_2	V_3	V_4	V_5	V_4	V_3	V_2	V_1

Note: Each event priority in the table above represents one BACnetSpecialEvent. The BACnetSpecialEvent shall contain the time value pairs listed in the table (D_x, V_x). There shall be only one BACnetSpecialEvent per priority for this test.

Test Steps:

1. REPEAT D = (the times in the configured time-value pairs with non-NULL values) DO {
   (TRANSMIT TimeSynchronization-Request, 'Time' = D) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D) |
   MAKE (the local date and time = D)
   WAIT **Schedule Evaluation Fail Time**
   VERIFY Present_Value = (the value corresponding to the time D)
   }
2. REPEAT D = (the times in the configured time-value pairs with NULL values,
   except the final D_N) DO
   (TRANSMIT TimeSynchronization-Request, 'Time' = D) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D) |
   MAKE (the local date and time = D)
   WAIT **Schedule Evaluation Fail Time**
   VERIFY Present_Value = (the non-NULL value corresponding to the priority lower than that
   associated with D)
   }

7.3.2.23.10.3.9 Revision 4 List of BACnetTimeValue Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a BACnetSpecialEvent's entire schedule can be executed.

Test Concept: A special event is scheduled that contains multiple BACnetTimeValue entries with distinguishable non-NULL values. The local date and time are changed to values that match each of the BACnetTimeValue entries, and the Present_Value property is read to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured with a Schedule object containing a BACnetSpecialEvent with two or more BACnetTimeValue entries. Each BACnetTimeValue entry shall be non-NULL and have a distinguishable value. The BACnetSpecialEvent selected for this test shall be the highest priority event active for the day selected for testing.

Test Steps:

1. REPEAT D_i = (the times used in the BACnetTimeValue pairs of the special event) DO {
   (TRANSMIT TimeSynchronization-Request, 'Time' = D_i)
   | (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_i)
   | MAKE (the local date and time = D_i)
   WAIT **Schedule Evaluation Fail Time**
   VERIFY Present_Value = (the value corresponding to the special event)
   }

7.3.2.23.10.3.10 Revision 4 Calendar Entry WeekNDay Odd-Numbered Month Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that a date matching a WeekNDay's Month field, specifying odd-numbered months (BACnetWeekNDay month enumeration value 13), in an Exception_Schedule enables the referencing Schedule object.

Test Concept: The IUT's local date and time are changed such that all months of the year are tested. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a WeekNDay entry specifying odd-numbered months in a specific year and a second, lower priority, BACnetCalendarEntry with a WeekNDay entry specifying all days of the year specified in the first. The criteria for the dates used in the test are given in Table 7-16.3.

Table 7-16.3. Criteria for Calendar Entry WeekNDay Month Test Dates and Values

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay: specifies odd-numbered months with a specific year (Y), and 2C. Higher eventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay:Month specifies all days of the year used in D_1, and 2C. Lower eventPriority than that used for D_1.	V_2 (different from V_1)

Test Steps:

1. REPEAT X = (All months, 1-12) {
   (TRANSMIT TimeSynchronization-Request, 'Time' = D, where month = X and year is Y)
   | (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D, where month = X and year is Y)
   | MAKE (the local date and time = D, where month = X and year is Y)
   WAIT **Schedule Evaluation Fail Time**
   IF (X is odd) THEN
   VERIFY Present_Value = V_1
   ELSE
   VERIFY Present_Value = V_2
   }

7.3.2.23.10.3.11 Revision 4 Calendar Entry WeekNDay Even-Numbered Month Test

This test is identical to **7.3.2.23.X2.3.10**, except that even-numbered months (BACnetWeekNDay month enumeration value 14) are used instead of odd-numbered months.

7.3.2.23.10.3.12 Revision 4 Lower Event Priority Change Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.24.8.

Purpose: To verify that when a BACnetSpecialEvent of a higher priority takes precedence over one of lower priority, that a change in the lower priority level is not observed in Present_Value until control is relinquished to it.

Configuration Requirements: A Schedule object is configured with two BACnetSpecial Events, thus: the first event is at lower priority than the second and contains two time-value pairs: the first, D_1, has a non-NULL value V_1 and the second, D_4, has a non-NULL value V_4 which is different from V_1 and different from V_3. The second event contains three time-value pairs: the first, D_2, occurs after D_1 and before D_3 and has a non-NULL value V_2 different from the value V_1; the second, D_3, occurs after D_2 and has a non-NULL value V_3 different from the value V_2; the third, D_5, occurs after D_4 and has a NULL value. (This arrangement of events facilitates testing Schedule objects that schedule only BOOLEAN or two-state enumerations.) Table 7-16.4 illustrates the time and value pairs in this test.

Table 7-16.4. Event and value prioritization test times and value

	Time:				
Event Priority:	D_1	D_2	D_3	D_4	D_5
Higher	-	V_2	V_3	-	NULL
Lower	V_1	-	-	V_4	-
Present_Value:	V_1	V_2	V_3	V_3	V_4

Note: Each event priority in the table above represents one BACnetSpecialEvent. The BACnetSpecialEvent shall contain the time value pairs listed in the table (D_x, V_x). There shall be only one BACnetSpecialEvent per priority for this test.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
2. VERIFY Present_Value = V_1
3. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
4. VERIFY Present_Value = V_2
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_3) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_3) |
   MAKE (the local date and time = D_3)
6. VERIFY Present_Value = V_3
7. (TRANSMIT TimeSynchronization-Request, 'Time' = D_4) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_4) |
   MAKE (the local date and time = D_4)
8. VERIFY Present_Value = V_3
9. (TRANSMIT TimeSynchronization-Request, 'Time' = D_5) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_5) |
   MAKE (the local date and time = D_5)
10. VERIFY Present_Value = V_4

7.3.2.23.10.3.13 Revision 4 Schedule_Default Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.24.9.

Purpose: To verify that the value in Schedule_Default is applied when no weekly or exception schedule is in effect.

Configuration Requirements: The IUT shall be configured with a Schedule object with a Schedule_Default value $V_{default}$ and containing at least one of, and if possible, both (non-overlapping):

- a Weekly_Schedule containing a time-value pair at time D_1 with a non-NULL value V_1 different from $V_{default}$ and a subsequent time-value pair with a NULL value at time D_2.
- an Exception_Schedule with no overlap with the time frame D_1 to D_2, a time-value pair at time D_3 with a non-NULL value V_3 different from $V_{default}$, and a subsequent time-value pair with a NULL value at time D_4.

Test Steps:

1. IF (the Schedule object is configured with a Weekly_Schedule) THEN
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_1) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_1) |
   MAKE (the local date and time = D_1)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = $V_{default}$
8. IF (the Schedule object is configured with an Exception_Schedule) THEN
9. (TRANSMIT TimeSynchronization-Request, 'Time' = D_3) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_3) |
   MAKE (the local date and time = D_3)
10. WAIT **Schedule Evaluation Fail Time**
11. VERIFY Present_Value = V_3
12. (TRANSMIT TimeSynchronization-Request, 'Time' = D_4) |
    (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_4) |
    MAKE (the local date and time = D_4)
13. WAIT **Schedule Evaluation Fail Time**
14. VERIFY Present_Value = $V_{default}$

[Add new **Clauses 7.3.2.23.10.4** through **7.2.23.10.8, p. 110**]

7.3.2.23.10.4 Revision 4 Weekly_Schedule and Exception_Schedule Interaction Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30.

BACnet Reference Clauses: 12.24.7 and 12.24.8.

Purpose: To verify that an Exception_Schedule takes precedence over a coincident Weekly_Schedule; to verify that Weekly_Schedule automatically takes control once it becomes active and after the Exception_Schedule is expired; and to verify that the value in Schedule_Default is applied when no weekly or exception schedule is in effect.

Test Concept: The IUT is configured with a Weekly_Schedule and an Exception_Schedule that apply to the same date. The local time is changed to the time when the Exception_Schedule takes control, and the Present_Value is read to verify that the scheduled write operation occurs. The local time is changed again to a value that would cause another change if the Weekly_Schedule were in control. The Present_Value is read to verify the Exception_Schedule is still in control. The local time is changed again to a value where Exception_Schedule is not in effect, and the Present_Value is read to verify that Weekly_Schedule is in control. The local time is changed again to a value where both Exception_Schedule and Weekly_Schedule are not in effect, and the Present_Value is read to verify that Schedule_Default is written into it.

Configuration Requirements: The IUT shall be configured with a Schedule object containing a Weekly_Schedule and an Exception_Schedule that both apply to a particular day, D_1, within the Effective_Period. There shall be no other BACnetSpecial Events in the Exception_Schedule. The Weekly_Schedule for the day of week entry related to D_1 is configured with the time value pairs (T_2, V_2) and (T_4, NULL), and the Exception_Schedule is configured with the time value pairs (T_1, V_1) and (T_3, NULL) where $T_1 < T_2 < T_3 < T_4$. The values V_1 and V_2 shall be different, and if possible, different from $V_{default}$, the Schedule_Default. If possible, a non-NULL Schedule_Default is used.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = T_1 on day D_1)
   | (TRANSMIT UTCTimeSynchronization-Request, 'Time' = T_1 on day D_1)
   | MAKE (the local time = T_1 on day D_1)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = V_1
4. (TRANSMIT TimeSynchronization-Request, 'Time' = T_2 on day D_1)
   | (TRANSMIT UTCTimeSynchronization-Request, 'Time' = T_2 on day D_1)
   | MAKE (the local time = T_2 on day D_1)
5. WAIT **Schedule Evaluation Fail Time**
6. VERIFY Present_Value = V_1
7. (TRANSMIT TimeSynchronization-Request, 'Time' = T_3 on day D_1)
   | (TRANSMIT UTCTimeSynchronization-Request, 'Time' = T_3 on day D_1)
   | MAKE (the local time = T_3 on day D_1)
8. WAIT Schedule Evaluation Fail Time
9. VERIFY Present_Value = V_2
10. (TRANSMIT TimeSynchronization-Request, 'Time' = T_4 on day D_1)
    | (TRANSMIT UTCTimeSynchronization-Request, 'Time' = T_4 on day D_1)
    | MAKE (the local time = T_4 on day D_1)
11. WAIT **Schedule Evaluation Fail Time**
12. VERIFY Present_Value = $V_{default}$

7.3.2.23.10.5 Revision 4 Exception_Schedule Restoration Test

Purpose: To verify the restoration behavior in an Exception_Schedule.

Test Steps: The Revision 4 version of this test is identical to the test in Clause 7.3.2.23.5 Exception_Schedule Restoration Test.

7.3.2.23.10.6 Revision 4 Weekly_Schedule Restoration Test

Purpose: To verify the restoration behavior in an Exception_Schedule.

Test Steps: The Revision 4 version of this test is identical to the test in Clause 7.3.2.23.6.

7.3.2.23.10.7 Revision 4 List_Of_Object_Property_Reference Internal Test

Purpose: To verify that the Schedule object writes to objects and properties contained within the IUT.

Test Steps: The Revision 4 version of this test is identical to the test in Clause 7.3.2.23.7.

7.3.2.23.10.8 Revision 4 List_Of_Object_Property_Reference External Test

Purpose: To verify that the Schedule object writes to object properties contained in a device other than the IUT.

Test Steps: The Revision 4 version of this test is identical to the test in Clause 7.3.2.23.8.

[Add new **Clause 7.3.2.23.12**, p. 115]

7.3.2.23.12 Revision 4 Midnight Evaluation Test

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.24.4.

Purpose: To verify that the Schedule object evaluates its schedule as it passes through midnight (00:00).

Configuration Requirements: The IUT shall be configured with a Schedule object with a Schedule_Default value $V_{default}$, and containing at least one of, and if possible, both (non-overlapping):

- a Weekly_Schedule containing a time-value pair at time D_1 (not 00:00) with a non-NULL value V_1 different from $V_{default}$, and no scheduled write operations during the day after D_1, and none on the day following.

- an Exception_Schedule with an event occurring on a day different from D_1, containing a time-value pair at time D_3 with a non-NULL value V_3 different from $V_{default}$, and no scheduled write operations during the day after D_3, and none on the day following.

Two additional times, used in the execution of the test, are defined as follows:

- D_2 occurring on the same day as D_1, after D_1, and before midnight.
- D_4 occurring on the same day as D_3, after D_3, and before midnight.

It is recommended that to minimize testing time, D_1 through D_4 be chosen to be close to midnight. However, all times used in this test shall be separated by at least **Schedule Evaluation Fail Time**.

An illustration of the test times and values configured and observed is shown in **Table 7-21**.

Table 7-21. Test Times and Values

Time:	D_1	D_2	00:00	D_3	D_4	00:00
Exception_Schedule:	-	-	-	V_3	-	-
Weekly_Schedule:	V_1	-	-	-	-	-
Present_Value:		V_1	$V_{default}$		V_3	$V_{default}$

Test Steps:

1. IF (the Schedule object is configured with a Weekly_Schedule) THEN
2. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
3. WAIT **Schedule Evaluation Fail Time**
4. VERIFY Present_Value = V_1
5. WAIT (until 00:00)
6. WAIT **Schedule Evaluation Fail Time**
7. VERIFY Present_Value = $V_{default}$
8. IF (the Schedule object is configured with an Exception_Schedule) THEN
9. (TRANSMIT TimeSynchronization-Request, 'Time' = D_4) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_4) |
   MAKE (the local date and time = D_4)
10. WAIT **Schedule Evaluation Fail Time**
11. VERIFY Present_Value = V_3
12. WAIT (until 00:00)
13. WAIT **Schedule Evaluation Fail Time**
14. VERIFY Present_Value = $V_{default}$

135.1-2011j-10. Revise Stop_When_Full Test.

Rationale

Simplify the tests by monitoring Record_Count instead of Total_Record_Count and generalize for all logging object types.

[Change **Clause 7.3.2.24.6.1**, p. 119]

7.3.2.24.6.1 Stop_When_Full TRUE Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

~~BACnet Reference Clause: 12.25.12.~~

Purpose: To verify that Stop_When_Full set to TRUE properly indicates that the ~~Trend Log~~ *logging object* ceases collecting data when its Log_Buffer acquires Buffer_Size data items.

Test Concept: The ~~Trend Log~~ *logging object* is configured to acquire data by whatever means. Data is collected until more than Buffer_Size records have been collected and ~~Log_~~Enable is verified to be FALSE.

Configuration Requirements: Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present, shall be configured with the latest possible date and time, in order that it occur after the end of the test. Stop_When_Full, if configurable, shall be set to ~~FALSE~~*TRUE*. ~~Log_~~Enable shall be set to FALSE.

Test Steps:

[Note: test steps are renumbered without change marking for clarity.]

1. WRITE Record_Count = 0
~~2. WAIT **Internal Processing Fail Time**~~
~~3. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Total_Record_Count,~~
~~4. RECEIVE ReadProperty-ACK,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Total_Record_Count,~~
~~'Property Value' = (any valid value, X)~~
2. WRITE ~~Log_~~Enable = TRUE
3. WHILE (~~(Total_~~Record_Count ~~– (value X returned in step 4)) modulo 2^{32}~~ < Buffer_Size) DO { }
4. WAIT **Internal Processing Fail Time**
5. VERIFY ~~Log_~~Enable = FALSE

[Change **Clause 7.3.2.24.6.2**, p. 120]

7.3.2.24.6.2 Stop_When_Full FALSE Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

~~BACnet Reference Clause: 12.25.12.~~

Purpose: To verify that Stop_When_Full set to FALSE properly indicates that the ~~Trend Log~~ *logging object* continues collecting data after its Log_Buffer acquires Buffer_Size data items.

Test Concept: The ~~Trend Log~~ *logging object* is configured to acquire data by whatever means. Data is collected until more than Buffer_Size records have been collected and ~~Log_~~Enable is verified to be TRUE.

Configuration Requirements: Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured with the latest possible date and time, in order that it occur after the end of the test. Stop_When_Full, if configurable, shall be set to FALSE. ~~Log_~~Enable shall be set to FALSE.

Test Steps:

[Note: test steps are renumbered without change marking for clarity.]

1. WRITE Record_Count = 0
~~2. WAIT **Internal Processing Fail Time**~~
~~3. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Total_Record_Count~~
~~4. RECEIVE ReadProperty-ACK,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Total_Record_Count,~~
~~'Property Value' = (any valid value, X)~~
2. WRITE ~~Log_~~Enable = TRUE
3. WHILE (~~(Total_~~Record_Count ~~– (value X returned in step 4)) modulo 2~~^{32} < ~~(~~Buffer_Size~~+1)~~) DO { }
4. WAIT **Internal Processing Fail Time**
5. VERIFY ~~Log_~~Enable = TRUE

135.1-2011j-11. Make the Start_Time Test Generic.

Rationale

Generalize the Start_Time test for use on all logging object types.

[Change **Clause 7.3.2.24.2**, p. 116]

7.3.2.24.2 Start_Time Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

~~BACnet Reference Clause: 12.23.6.~~

Purpose: To verify that logging is enabled at the time specified by Start_Time.

Test Concept: The *logging object*~~Trend Log~~ is configured to acquire data by each means ~~(polling and COV subscription)~~ available to the implementation. The test is begun at some time prior to the time specified in Start_Time and non-collection of records is confirmed. Collection of records after the time specified by Start_Time is then confirmed.

Configuration Requirements: Start_Time shall be configured with a date and time such that steps 1 through 6 will be concluded before that time. Stop_Time, *if present, shall* ~~if present shall~~ be configured with the latest possible date and time, in order that it occurs after the end of the test. Stop_When_Full, if configurable, shall be set to FALSE; ~~Log~~ Enable shall be set to TRUE.

Test Steps:

1. WRITE Record_Count = 0
2. WAIT **Internal Processing Fail Time**
3. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (the object being tested),
   'Property Identifier' = Total_Record_Count
4. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (the object being tested),
   'Property Identifier' = Total_Record_Count
   'Property Value' = (any valid value, X)

~~5. IF (COV subscription in use) THEN~~
~~MAKE (monitored value change more than Client_COV_Increment)~~
~~ELSE~~
~~WAIT (Log_Interval)~~

5. MAKE (IUT collect another record)

6. WAIT (**Notification Fail Time** + **Internal Processing Fail Time**)
7. VERIFY Total_Record_Count = (value X returned in step 4)
8. WHILE (IUT clock is earlier than Start_Time) DO
   VERIFY Total_Record_Count = (value X returned in step 4)
9. WAIT (**Notification Fail Time** + **Internal Processing Fail Time**)

~~10. IF (COV subscription in use) THEN~~
~~MAKE (monitored value change more than Client_COV_Increment)~~
~~ELSE~~
~~WAIT (Log_Interval)~~

10. MAKE (IUT collect another record)

11. WAIT (**Notification Fail Time** + **Internal Processing Fail Time**)
12. VERIFY Total_Record_Count > (value X returned in step 4)

Note to Tester: For each MAKE (IUT collect another record), perform the following actions:

IF (the logging object is an Event Log Object) THEN
MAKE (Event Log Object collect another record)
ELSE

IF (COV subscription in use) THEN
MAKE (monitored value change sufficient to generate another record)
ELSE IF (interval or period logging is in use) THEN
WAIT (Log_Interval)
ELSE
MAKE (Trend Log or Trend Log Multiple Object collect another record)

135.1-2011j-12. Make the Log_Interval Test Generic.

Rationale

Generalize the Log_Interval test for use on all logging object types.

Note that some of the Trend Log tests were generalized in 135.1-2009i. Others were modified in 135.1-2009g and will be generalized in an addendum to that addendum.

[Change **Clause 7.3.2.24**, p. 115]

7.3.2.24 ~~Trend Log~~ *Logging* Object Tests

~~The Trend Log object has~~ *Logging objects have* only a few properties required to be writable or otherwise configurable. ~~The Trend Log object~~ *Logging objects* shall be configured to accommodate as many of the following tests as is possible for the implementation. If it is impossible to configure the IUT in the manner required for a particular test that test shall be omitted.

Tests of ~~the Trend Log object~~ *logging objects* center upon the collection of ~~(time, value)~~ records in ~~its~~ *the* Log_Buffer and ~~its~~ *the* issuance of notifications when a predetermined number of records have been collected since startup or the last preceding notification.

[Change **Clause 7.3.2.24.4**, p. 117]

7.3.2.24.4 Log_Interval Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

~~BACnet Reference Clause: 12.23.9.~~

Purpose: To verify that the logging period is controlled by Log_Interval.

Test Concept: The ~~Trend Log~~*log object* is configured to acquire data by polling. Polling is done at two different intervals, defined by Log_Interval, with about 10 records acquired at each rate. The timestamps of the records are inspected to verify the polling rate.

Configuration Requirements: Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured with the latest possible date and time, in order that it occur after the end of the test. Stop_When_Full, if configurable, shall be set to FALSE. ~~Log_~~Enable shall be set to TRUE. Non-zero values shall be chosen for Log_Interval in accordance with the range and resolution specified by the manufacturer for this property.

Test Steps:

1. WRITE Log_Interval = (some non-zero value)
2. WRITE Record_Count = 0
3. WAIT (**Internal Processing Fail Time** + 10* Log_Interval hundredths-seconds)
4. VERIFY (Log_Buffer record timestamp intervals, on average, are as written in step 1)
5. WRITE Log_Interval = (a non-zero value different from the one written in step 1)
6. WRITE Record_Count = 0
7. WAIT (**Internal Processing Fail Time** + 10* Log_Interval hundredths-seconds)
8. VERIFY (Log_Buffer record timestamp intervals, on average, are as written in step 5)

135.1-2011j-13. Make the Buffer_Size Test Generic.

Rationale

Generalize the Buffer_Size test for use on all logging object types.

[Change **Clause 7.3.2.24.7**, p. 121]

7.3.2.24.7 Buffer_Size Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clause: 12.23.13.

Purpose: To verify that Buffer_Size properly indicates the number of records that can be stored in the Log_Buffer.

Test Concept: The ~~Trend Log~~*log object* is configured to acquire data by whatever means. Data is collected until at least Buffer_Size records have been collected, then the Log_Buffer is read and the presence of Buffer_Size discrete records is verified.

Configuration Requirements: Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured with the latest possible date and time, in order that it occur after the end of the test. ~~Log_~~Enable shall be set to TRUE.

Test Steps:

1. WHILE (Record_Count < Buffer_Size) DO { }
2. WRITE ~~Log_~~Enable = FALSE
3. WAIT **Internal Processing Fail Time**
4. CHECK (that Log_Buffer has Buffer_Size discrete records)

135.1-2011j-14. Correct the Record_Count Test.

Rationale

The test incorrectly assumed that Log_Buffer would be empty after being purged when it should contain a buffer-purged-event record. The test is also made generic so it can be applied to all logging object types.

[Change **Clause 7.3.2.24.8**, p. 121]

7.3.2.24.8 Record_Count Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

~~BACnet Reference Clause: 12.25.15.~~

Purpose: To verify that the Record_Count property indicates the number of records that are stored in the Log_Buffer.

Test Concept: The ~~Trend Log~~ *logging object* is configured to acquire data by whatever means. Record_Count is set to zero and Log_Buffer is read to verify ~~no records are present~~*that only one record is present and that it is the buffer-purged event*. Collection of data proceeds until Record_Count is about Buffer_Size/2, collection is halted and Log_Buffer is read to verify the Record_Count value. Collection then resumes until Buffer_Size records are read; collection is then halted and Log_Buffer read to verify *the* Record_Count again.

Configuration Requirements: Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured with the latest possible date and time, in order that it occur after the end of the test. ~~Log_~~Enable shall be set to FALSE.

Test Steps:

1. WRITE Record_Count = 0
2. WAIT **Internal Processing Fail Time**
~~3. CHECK (that Log_Buffer has no records)~~
3. CHECK (Log_Buffer contains one entry, and it is the buffer-purged event)
4. WRITE ~~Log_~~Enable = TRUE
5. WHILE (Record_Count < Buffer_Size/2) DO { }
6. WRITE ~~Log_~~Enable = FALSE
7. WAIT **Internal Processing Fail Time**
8. ~~VERIFY~~ *CHECK* (that Log_Buffer has the number of records indicated by Record_Count)
9. WRITE ~~Log_~~Enable = TRUE
10. WHILE (Record_Count < Buffer_Size) DO { }
11. WRITE ~~Log_~~Enable = FALSE
12. WAIT **Internal Processing Fail Time**
13. ~~VERIFY~~ *CHECK* (Log_Buffer has the number of records indicated by Record_Count)

135.1-2011j-15. Correct the Notification_Threshold Test.

Rationale

The calculation for the Total_Record_Count and other record count properties are incorrect and generalized for all logging object types.

The Event_Values description is also incorrect.

[Change **Clause 7.3.2.24.11**, p. 123]

7.3.2.24.11 Notification_Threshold Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

~~BACnet Reference Clause: 12.25.17.~~

Purpose: To verify that the Notification_Threshold property reflects the number of records collected since a previous notification, or since logging started, that causes a Buffer_Ready notification to be sent.

Test Concept: The ~~Trend Log~~ *logging object* is configured to acquire data by whatever means. Record_Count is set to zero. Collection of data proceeds until a notification is seen, ~~collection is halted~~ and the value of Record_Count is checked. Collection ~~resumes~~ *continues* until the second notification, when ~~collection is again halted and~~ Record_Count is verified *again*. ~~If, for whatever reason, the IUT cannot be configured such that the TD is able to halt collection before another record is collected after issuing the notification this test shall not be performed.~~

Configuration Requirements: Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured with the latest possible date and time, in order that it occur after the end of the test. ~~Log_~~Enable shall be set to FALSE. *Notification_Threshold shall be set to a non-zero value.*

Test Steps:

[Note that the test steps are renumbered with change marks for clarity.]

1. WRITE Record_Count = 0
2. WAIT **Internal Processing Fail Time**
~~3. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Total_Record_Count~~
~~4. RECEIVE ReadProperty-ACK,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Total_Record_Count~~
~~'Property Value' = (any valid value, X)~~
3. *READ X = Total_Record_Count*
4. WRITE ~~Log_~~Enable = TRUE
5. MAKE (~~Trend Log~~ *the logging* object collect number of records specified by Notification_Threshold)
6. RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process ID),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the ~~Trend Log~~ object being tested),
'Time Stamp' =	(any appropriate BACnetTimeStamp value),
'Notification Class' =	(the configured notification class),
'Priority' =	(the value configured to correspond to a TO-NORMAL transition),
'Event Type' =	BUFFER_READY,
'Notify Type' =	EVENT | ALARM,
'AckRequired' =	TRUE | FALSE,
'From State' =	NORMAL,
'To State' =	NORMAL,

' Event Values' =	~~(BACnetObjectIdentifier of the IUT's Device object),~~ ('*Buffer Property*': ~~BACnetObjectIdentifier of~~ the ~~Trend Log~~ *logging* object), (*'Previous Notification':* ~~any BACnetDateTime~~*any valid value < X*), (*'Current Notification':* ~~current local BACnetDateTime~~*any value Y_1 where Y_1 >= X and Y_1 <= X + Notification_Threshold*)

7. TRANSMIT BACnet-SimpleACK-PDU
8. *VERIFY Total_Record_Count >= Y_1*
~~9. WRITE Log_Enable = FALSE~~
~~10. IF (Total_Record_Count – (value read in step 4) != Notification_Threshold) THEN~~
~~ERROR "Notification_Threshold value is incorrect."~~
~~11. WRITE Log_Enable = TRUE~~
9. MAKE (~~Trend Log~~ *the logging* object collect number of records specified by Notification_Threshold)
10. RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process ID),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the ~~Trend Log~~ object being tested),
'Time Stamp' =	(any appropriate BACnetTimeStamp value),
'Notification Class' =	(the configured notification class),
'Priority' =	(the value configured to correspond to a TO-NORMAL transition),
'Event Type' =	BUFFER_READY,
'Notify Type' =	EVENT | ALARM,
'AckRequired' =	TRUE | FALSE,
'From State' =	NORMAL,
'To State' =	NORMAL,
'Event Values' =	~~(BACnetObjectIdentifier of the IUT's Device object),~~ (*'Buffer Property':* ~~BACnetObjectIdentifier of~~ the ~~Trend Log~~ *logging* object), (*'Previous Notification':* ~~BACnetDateTime sent in step 7~~ *Y_1*), (*'Current Notification':* ~~current local BACnetDateTime~~ *Y_1 + Notif*ication_Threshold)

11. TRANSMIT BACnet-SimpleACK-PDU
12. *VERIFY Total_Record_Count >= Y_1 + Notification_Threshold*
13. WRITE Log_Enable = FALSE
~~16. IF (Total_Record_Count – (value X returned in step 4) != 2 * Notification_Threshold) THEN~~

135.1-2011j-16. Add Trigger Verification Tests.

Rationale

Add a test for Trend Log and Trend Log Multiple trigger functionality.

[Add new **Clause 7.3.2.24.19**, p 131]

7.3.2.24.19 Trigger Verification Test

Dependencies: ReadRange Service Execution, 9.21;

BACnet Reference Clause: 12.25.27 and 12.30.12.

Purpose: To verify logged samples are based on the triggered Logging_Type.

Test Concept: The log is configured to log based on TRIGGERED. Logging is enabled. After a period of time, the buffer is checked to verify the data in the buffer is based on triggered values.

Configuration Requirements: The IUT shall be configured such that the monitored object's Logging_Type is set to TRIGGERED.

Test Steps:

1. WRITE Enable = FALSE
2. WRITE Record_Count = 0 -- results in a buffer purged record
3. WRITE Enable = TRUE -- results in a logging enable record
4. WAIT (10 seconds)
5. WRITE Trigger = TRUE
6. WAIT (20 seconds)
7. WRITE Trigger = TRUE
8. WAIT (40 seconds)
9. WRITE Trigger = TRUE
10. WAIT (30 seconds)
11. WRITE Enable = FALSE -- results in a logging disabled record
12. VERIFY RecordCount = 6
13. REPEAT X = (1 through 6)
    TRANSMIT ReadRange
    'Object Identifier' = O1,
    'Property Identifier' = Log_Buffer,
    'Reference Index' = X,
    'Count' = 1
    RECEIVE ReadRangeAck
    'Object Identifier' = O1,
    'Property Identifier' = Log_Buffer,
    'Result Flags' = (?, ?, False),
    'Item Count' = 1
    'Item Data' = ((one data record storing the timestamp in TS[X]))
16. CHECK (TS[3] - TS[2] ~= 10 seconds)
17. CHECK (TS[4] - TS[3] ~= 20 seconds)
18. CHECK (TS[5] - TS[4] ~= 40 seconds)
19. CHECK (TS[6] - TS[5] ~= 30 seconds)

135.1-2011j-17. Update BUFFER_READY Tests.

Rationale

Update the BUFFER_READY test to conform with 135-2001b and generalize the test for all logging object types.

[Change **Clause 8.4.7**, p. 159]

8.4.7 BUFFER_READY Tests

Dependencies: ReadProperty Service Execution Tests, 9.18.

BACnet Reference Clauses: 12.12, 12.25, 13.2, 13.3.7, and 13.8.

Purpose: To verify the correct operation of the BUFFER_READY event algorithm. This test applies to ~~Trend Log~~ *logging* objects that support intrinsic *reporting* ~~notification~~ and to Event Enrollment objects with an Event_Type of BUFFER_READY.

Test Concept: The object that performs the notification ~~("the notifying object")~~ *("the event initiating object")* begins the test in a NORMAL state~~, with no records stored in the object containing the buffer ("the buffer object")~~. *The object containing the buffer ("the buffer object")* ~~buffer object~~ acquires *enough records to generate a notification* ~~the number of records specified by Records_Since_Notification~~, at which time the notifying object performs a TO-NORMAL transition and sends *a* BUFFER_READY notification~~s~~.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-NORMAL transition. The ~~notifying~~ *event initiating* object shall be in a NORMAL state at the start of the test. The 'Issue Confirmed Notifications' parameter of the element of the Notification Class' Recipient_List property referring to the ~~IUT~~ *TD* shall be set to TRUE.

Test Steps:

1. VERIFY Event_State = NORMAL
2. MAKE (buffer object collect *enough records to generate a notification.* ~~number of records specified by Notification_Threshold~~)
3. RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process ID),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the intrinsic reporting object being tested or the object referenced by the Event Enrollment object being tested),
'Time Stamp' =	(any appropriate BACnetTimeStamp value),
'Notification Class' =	(the configured notification class),
'Priority' =	(the value configured to correspond to a TO-NORMAL transition),
'Event Type' =	BUFFER_READY,
'Notify Type' =	EVENT | ALARM,
'AckRequired' =	TRUE | FALSE,
'From State' =	NORMAL,
'To State' =	NORMAL,
' Event Values' =	~~(BACnetObjectIdentifier of the IUT's Device object),~~
	~~(BACnetObjectIdentifier of the buffer object),~~
	~~(any BACnetDateTime),~~
	~~(current local BACnetDateTime)~~
	('Buffer Property': reference to the Log_Buffer property),
	('Previous Notification': any valid value),
	('Current Notification': any valid value CN1)

4. TRANSMIT BACnet-SimpleACK-PDU
5. MAKE (~~buffer~~ *logging* object collect *the* number of records specified by Notification_Threshold)
6. RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process ID),

'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the intrinsic reporting object being tested or the object referenced by the Event Enrollment object being tested),
'Time Stamp' =	(any appropriate BACnetTimeStamp value),
'Notification Class' =	(the configured notification class),
'Priority' =	(the value configured to correspond to a TO-NORMAL transition),
'Event Type' =	BUFFER_READY,
'Notify Type' =	EVENT | ALARM,
'AckRequired' =	TRUE | FALSE,
'From State' =	NORMAL,
'To State' =	NORMAL,
' Event Values' =	~~(BACnetObjectIdentifier of the IUT's Device object),~~ ~~(BACnetObjectIdentifier of the buffer object),~~ ~~(current local BACnetDateTime sent in step 3),~~ ~~(current local BACnetDateTime)~~ *('Buffer Property': reference to the Log_Buffer property),* *('Previous Notification': CN1),* *('Current Notification': CN1 + Notification_Threshold)*

7. TRANSMIT BACnet-SimpleACK-PDU

[Change **Clause 8.5.7**, p. 181]

8.5.7 BUFFER_READY Tests

Dependencies: ReadProperty Service Execution Tests, 9.18.

BACnet Reference Clauses: 12.12, 12.25, 13.2, 13.3.7, and 13.9.

Purpose: To verify the correct operation of the BUFFER_READY event algorithm. This test applies to ~~Trend Log~~ *logging* objects that support intrinsic notification and to Event Enrollment objects with an Event_Type of BUFFER_READY.

The test concept, configuration requirements and test steps are identical to those in 8.4.7, except that the 'Issue Confirmed Notifications' parameter shall have a value of FALSE, UnconfirmedEventNotification requests are issued instead of ConfirmedEventNotification requests and the TD does not acknowledge receiving the notifications.

135.1-2011j-18. Add COV Subscription Lifetime Value Range Tests.

Rationale

In addendum 135-2008s, minimum requirements for COV lifetime were added. These new tests ensure that devices implement these requirements.

[Add new **Clause 9.10.1.10**, p. 266]

9.10.1.10 Accepts 8 Hour Lifetimes

Purpose: To verify that the IUT correctly accepts lifetimes of at least 8 hours. Either confirmed or unconfirmed notifications may be used, but at least one of these options shall be supported by the IUT.

1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any object supporting COV notifications),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = 28800
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
   IF (the subscription was for confirmed notifications) THEN
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same identifier used in the subscription),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = (the same object used in the subscription),
   'Time Remaining' = (the requested subscription lifetime),
   'List of Values' = (values appropriate to the object type of the monitored object)
   ELSE
   RECEIVE UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same identifier used in the subscription),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = (the same object used in the subscription),
   'Time Remaining' = (the requested subscription lifetime),
   'List of Values' = (values appropriate to the object type of the monitored object)
4. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (the same identifier used in the subscription),
   'Monitored Object Identifier' = (the same object used in the subscription)

[Add new **Clause 8.10.4**, p. 189]

8.10.4 Requests 8 Hour Lifetimes

Purpose: To verify that the IUT correctly generates subscription requests with lifetimes less than or equal to 8 hours. Either confirmed or unconfirmed notifications may be used, but at least one of these options shall be supported by the IUT.

1. RECEIVE SubscribeCOV-Request,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any object supporting COV notifications),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid lifetime between 1 and 28800)
2. TRANSMIT BACnet-SimpleACK-PDU

135.1-2011j-19. Modify List Management Test.

Rationale

The existing tests require that devices initiate AddListElement and RemoveListElement services with two or more entries in order to pass the tests. This is reduced to 1 entry.

[Change **Clause 8.14**, p. 191]

8.14.1 Non-Array Properties

Purpose: To verify that the IUT can initiate an AddListElement service request that does not contain the 'Property Array Index' parameter.

Test Steps:

1. RECEIVE AddListElement-Request,
   'Object Identifier' = (any object that contains a property having a list datatype),
   'Property Identifier' = (any property having a list datatype),
   'List of Elements' = (~~two~~ *one* or more elements with the correct datatype to add to the list)
2. TRANSMIT BACnet-SimpleACK-PDU

[Change **Clause 8.14.2**, p. 191]

8.14.2 Array Properties

Purpose: To verify that the IUT can initiate an AddListElement service request that contains the 'Property Array Index' parameter.

Test Steps:

1. RECEIVE AddListElement-Request,
   'Object Identifier' = (any object that contains a property having a datatype that is an array of lists),
   'Property Identifier' = (any property having a datatype that is an array of lists),
   'Property Array Index' = (any value > 0),
   'List of Elements' = (~~two~~ *one* or more elements with the correct datatype to add to the list)
2. TRANSMIT BACnet-SimpleACK-PDU

[Change **Clause 8.15.1**, p. 192]

8.15.1 Non-Array Properties

Purpose: To verify that the IUT can initiate a RemoveListElement service request that does not contain the 'Property Array Index' parameter.

Test Steps:

1. RECEIVE RemoveListElement-Request,
   'Object Identifier' = (any object that contains a property having a list datatype),
   'Property Identifier' = (any property having a list datatype),
   'List of Elements' = (~~two~~ *one* or more elements with the correct datatype to remove from the list)
2. TRANSMIT BACnet-SimpleACK-PDU

[Change **Clause 8.15.2**, p. 192]

8.15.2 Array Properties

Purpose: To verify that the IUT can initiate a RemoveListElement service request that contains the 'Property Array Index' parameter.

Test Steps:

1. RECEIVE RemoveListElement-Request,
   'Object Identifier' = (any object that contains a property having a datatype that is an array of lists),
   'Property Identifier' = (any property having a datatype that is an array of lists),
   'Property Array Index' = (any value > 0),
   'List of Elements' = (~~two~~ *one* or more elements with the correct datatype to remove from the list)
2. TRANSMIT BACnet-SimpleACK-PDU

135.1-2011j-20. Implement COV Testing By Datatype.

Rationale

The existing test indicates that the test is to be applied to all of the appropriate object types. This direction is removed as is it to be provided by the test plan (as developed by the BTL).

[Change **Clause 9.2.1.1**, p. 245]

9.2.1.1 Change of Value Notification*s* ~~**from Analog, Binary, Multi-state, and Life Safety Objects**~~

~~COV notifications convey the value of the Present_Value and Status_Flags properties when initiated by Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, and Life Safety Zone objects. Since the ability to subscribe to COV notifications is general and can be applied to any of these object types, the IUT shall demonstrate that it correctly responds to COV notifications from objects representing each of these object types. The test procedure defined in this clause shall be applied once for each object type.~~

Purpose: To verify that the IUT can execute ConfirmedCOVNotification requests from ~~analog, binary, and multi-state objects~~ *object types that provide the Present_Value and Status_Flags properties in COV notifications*.

Test Steps:

~~REPEAT X = (one object of each type in the set {Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, Life Safety Zone}) DO {~~

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications ' = TRUE,
   'Lifetime' = (a value greater than one minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 2),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining in the subscription),
   'List of Values' = (Present_Value and Status_Flags appropriate to object *type* X)
4. RECEIVE BACnet-SimpleACK-PDU
5. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

~~}~~

[Insert **Clauses 9.2.1.3-4**, p. 246]

9.2.1,3 Change of Value Notification from Pulse Converter Object

Purpose: To verify that the IUT can execute ConfirmedCOVNotification requests from Pulse Converter objects.

Test Steps:

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any Pulse Converter object, X),
   'Issue Confirmed Notifications ' = FALSE,
   'Lifetime' = (a value greater than 1 minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 2)

'Initiating Device Identifier' = TD,
'Monitored Object Identifier' = X,
'Time Remaining' = (the time remaining for the subscription),
'List of Values' = (Present_Value, Status_Flags, and Update

4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

9.2.1.4 Change of Value Notification from Load Control Object

Purpose: To verify that the IUT can execute ConfirmedCOVNotification requests from Load Control objects.

Test Steps:

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any Load Control object, X),
   'Issue Confirmed Notifications ' = TRUE,
   'Lifetime' = (a value greater than 1 minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 2),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining for the subscription),
   'List of Values' = (Present_Value, Status_Flags, Requested_Shed_Level, Start_Time, Shed_Duration, and Duty_Window appropriate to object X)
4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

[Change **Clause 9.3**, p. 248]

9.3 UnconfirmedCOVNotification Service Execution Tests

~~BACnet does not define a service procedure for executing the UnconfirmedCOVNotification service and thus no tests are needed.~~ *The purpose of this test group is to verify correct execution of the UnconfirmedCOVNotification service requests under circumstances where the service is expected to be successfully completed.*

[Insert new **Clauses 9.3.2-5**, p. 248]

9.3.2 Change of Value Notifications

Purpose: To verify that the IUT can execute UnconfirmedCOVNotification requests from objects that provide the Present_Value and Status_Flags properties in COV notifications.

Test Steps:

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications ' = FALSE,
   'Lifetime' = (a value greater than 1 minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 2),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining for the subscription),
   'List of Values' = (Present_Value and Status_Flags appropriate to object type X)

4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

9.3.3 Change of Value Notification from Loop Objects

Purpose: To verify that the IUT can execute UnconfirmedCOVNotification requests from Loop objects.

Test Steps:

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any Loop object, X),
   'Issue Confirmed Notifications ' = FALSE,
   'Lifetime' = (a value greater than 1 minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 2),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining for the subscription),
   'List of Values' = (Present_Value, Status_Flags, Setpoint, and Controlled_Variable_Value appropriate to object X)
4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

9.3.4 Change of Value Notification from Pulse Converter Object

Purpose: To verify that the IUT can execute UnconfirmedCOVNotification requests from Pulse Converter objects.

Test Steps:

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any Pulse Converter object, X),
   'Issue Confirmed Notifications ' = FALSE,
   'Lifetime' = (a value greater than 1 minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 2),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining for the subscription),
   'List of Values' = (Present_Value, Status_Flags, and Update_Time appropriate to object X)
4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

9.3.5 Change of Value Notification from Load Control Object

Purpose: To verify that the IUT can execute UnconfirmedCOVNotification requests from Load Control objects.

Test Steps:

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any Load Control object, X),
   'Issue Confirmed Notifications ' = FALSE,
   'Lifetime' = (a value greater than 1 minute)

2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT UnconfirmedCOVNotification-Request,

'Subscriber Process Identifier' =	(the process identifier used in step 2),
'Initiating Device Identifier' =	TD,
'Monitored Object Identifier' =	X,
'Time Remaining' =	(the time remaining for the subscription),
'List of Values' =	(Present_Value, Status_Flags, Requested_Shed_Level, Start_Time, Shed_Duration, and Duty_Window appropriate to object X)

4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

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.