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

ASHRAE ADDENDA

Method of Test for Conformance to BACnet®

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

This addendum was approved by a Standing Standard Project Committee (SSPC) for which the Standards Committee has established a documented program for regular publication of addenda or revisions, including procedures for timely, documented, consensus action on requests for change to any part of the standard. The change submittal form, instructions, and deadlines may be obtained in electronic form from the ASHRAE Web site (www.ashrae.org) or in paper form from the Manager of Standards.

The latest edition of an ASHRAE Standard may be purchased on the ASHRAE Web site (www.ashrae.org) or from ASHRAE Customer Service, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. E-mail: orders@ashrae.org. Fax: 404-321-5478. Telephone: 404-636-8400 (worldwide), or toll free 1-800-527-4723 (for orders in US and Canada). For reprint permission, go to www.ashrae.org/permissions.

ISSN 1041-2336

Approved American National Standard
ANSI
www.ansi.org

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

SPECIAL NOTE

This American National Standard (ANS) is a national voluntary consensus standard developed under the auspices of the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). *Consensus* is defined by the American National Standards Institute (ANSI), of which ASHRAE is a member and which has approved this standard as an ANS, as "substantial agreement reached by directly and materially affected interest categories. This signifies the concurrence of more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that an effort be made toward their resolution." Compliance with this standard is voluntary until and unless a legal jurisdiction makes compliance mandatory through legislation.

ASHRAE obtains consensus through participation of its national and international members, associated societies, and public review.

ASHRAE Standards are prepared by a Project Committee appointed specifically for the purpose of writing the Standard. The Project Committee Chair and Vice-Chair must be members of ASHRAE; while other committee members may or may not be ASHRAE members, all must be technically qualified in the subject area of the Standard. Every effort is made to balance the concerned interests on all Project Committees.

The Manager of Standards of ASHRAE should be contacted for:

a. interpretation of the contents of this Standard,
b. participation in the next review of the Standard,
c. offering constructive criticism for improving the Standard, or
d. permission to reprint portions of the Standard.

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

FOREWORD

Addendum 135.1*i* to ANSI/ASHRAE Standard 135.1-2009 contains a number of changes to the current standard. These modifications are the result of change proposals made pursuant to the ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135. The changes are summarized below.

In the following document, language added to existing clauses of ANSI/ASHRAE 135.1-2009 and addenda is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are added, plain type is used throughout.

135.1-2009*i*-1. Improve Schedule Object Restoration Tests.

Rationale

ASHRAE 135.1 tests 7.3.2.23.5 and 7.3.2.23.6 examine whether a Schedule object will write the correct value from its Exception_Schedule and Weekly_Schedule properties when the device containing that schedule is reinitialized via the ReinitializeDevice service. These tests specify that the 'COLDSTART' argument shall be used when the device is reinitialized, but the BACnet standard does not define the meaning of 'COLDSTART'. Instead, these tests should specify the 'WARMSTART' argument.

[Change **Clause 7.3.2.23.5** Exception_Schedule Restoration Test, p. 91 and
Clause 7.3.2.23.6 Weekly_Schedule Restoration Test, p. 91]

…
4. IF (ReinitializeDevice execution is supported) THEN
TRANSMIT ReinitializeDevice-Request,
'Reinitialized State of Device' = ~~COLDSTART~~ *WARMSTART*,
'Password' = (any valid password)
RECEIVE BACnet-Simple-ACK-PDU
ELSE
MAKE (the IUT reinitialize)
5. CHECK (Did the IUT perform a ~~COLDSTART~~ *WARMSTART* reboot?)
…

135.1-2009*i*-2. Add Text to Check Range of the Present_Value of Multi-state Objects.

Rationale

This new test verifies that the Present_Value property of a Multi-state Input, Multi-state Output or Multi-state Value object remains within its valid range of values.

[Add new **Clause 7.3.1.X**, p. 49]

7.3.1.X Number_Of_States Range Test

Dependencies: ReadProperty Service Execution Tests, 7.1; WriteProperty Service Execution Tests 7.2.2

BACnet Reference Clauses: 12.18.11, 12.19.4, 12.19.11, 12.20.4, 12.20.10

Purpose: This test case verifies that the Present_Value property of a Multi-state Input, Multi-state Output or Multi-state Value object is limited to the range 1 through the value of its Number_Of_States property.

Test Concept: The Number_Of_States property is first verified to be a non-zero value. An attempt is then made to modify the Present_Value property of the referenced object to contain a value greater than the value of the Number_Of_States property. An attempt is then made to set the Present_Value property to 0. The test shall verify a correct error response for each case.

Configuration Requirements: The IUT shall be configured to contain a Multi-state Input, Multi-state Output, or Multi-state Value object, Object1, which contains a writable Present_Value property. Object1 shall be configured such that the Present_Value property is writable before executing the test. If the Present_Value property cannot be made writable, then this test shall be skipped.

Test Steps:

1. READ COUNT = Number_Of_States
2. VERIFY Number_Of_States = (a non-zero value)
3. TRANSMIT WriteProperty-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = Present_Value,
   'Property Value' = (any value larger than COUNT)
4. RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE
5. VERIFY (Object1), P1 = (a value between 1 and COUNT, inclusive)
6. TRANSMIT WriteProperty-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = Present_Value,
   'Property Value' = 0
7. RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE
8. VERIFY (Object1), P1 = (a value between 1 and COUNT, inclusive)

Note To Tester: When testing an object that is commandable, any priority may be selected.

135.1-2009*i*-3. Add Test for SubscribeCOV Service Execution Without a Lifetime Parameter.

Rationale

Add a test to ensure that the IUT correctly processes SubscribeCOV requests which do not include a Lifetime parameter.

[Add new **Clause 8.2.X1,** p. 116]

8.2.x1 Missing Lifetime Test

Purpose: This test case verifies the special case of the SubscribeCOV where a missing Lifetime parameter shall imply an indefinite Lifetime subscription.

Test Concept: A subscription for COV notification is established with the IUT. The subscribe message shall omit the Lifetime parameter. The COV notification is received from the IUT and the 'Time Remaining' value is verified to be 0.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications' = TRUE
2. RECEIVE BACnet-SimpleACK-PDU,
3. RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same value used in step 1),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = 0,
   'List of Values' = (the initial Present_Value and initial Status_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU

135.1-2009*i*-4. Update Test for Processing of ReadProperty Service Responses.

Rationale

Update the ReadProperty service initiation tests to provide a response to the IUT and verify that the IUT properly processes the responses. Add new ReadProperty service initiation tests to cover cases currently not tested.

[Change **Clause 8.18.1,** p. 165]

8.18.1 Reading Non-Array Properties

Purpose: To verify that the IUT can initiate a ReadProperty service request that does not contain the 'Property Array Index' ~~parameter.~~ *parameter and can correctly process the response.*

Test Steps:

1. RECEIVE ReadProperty-Request,
   'Object Identifier' = (any object),
   'Property Identifier' = (any valid non-array property of the specified object)
2. *TRANSMIT BACnet-ComplexACK-PDU,*
   'Object Identifier' = (object identifier from step 1),
   'Property Identifier' = (property identifier from step 1),
   'Property Value' = (any valid value for the property)
3. *CHECK (that the IUT exhibits the vendor defined results)*

[Change **Clause 8.18.2**, p. 165]

8.18.2 Reading an Array Element

Purpose: To verify that the IUT can initiate a ReadProperty service request that references a specific element of an array ~~property.~~*property and can correctly process the response.*

Test Steps:

1. RECEIVE ReadProperty-Request,
   'Object Identifier' = (any object),
   'Property Identifier' = (any valid array property of the specified object),
   'Array Index' = (any valid array index for the specified property)
2. *TRANSMIT BACnet-ComplexACK-PDU,*
   'Object Identifier' = (object identifier from step 1),
   'Property Identifier' = (property identifier from step 1),
   Array Index' = (array index from step 1),
   'Property Value' = (any valid value for the property)
3. *CHECK (that the IUT exhibits the vendor defined results)*

[Add new **Clause 8.18.X1,** p. 165]

8.18.X1 Reading Whole Array Properties

Purpose: To verify that the IUT can initiate a ReadProperty service request that does not contain the 'Property Array Index' parameter for an array property and can correctly process the response.

Test Steps:

1. RECEIVE ReadProperty-Request,
   'Object Identifier' = (any object),
   'Property Identifier' = (any valid array property of the specified object)
2. TRANSMIT BACnet-ComplexACK-PDU,
   'Object Identifier' = (object identifier from step 1),
   'Property Identifier' = (property identifier from step 1),
   'Property Value' = (any valid array of values for the property)
3. CHECK (that the IUT exhibits the vendor defined results)

[Add new **Clause 8.18.X2,** p. 165]

8.18.X2 Reading an Array Size

Purpose: To verify that the IUT can initiate a ReadProperty service request for the size of an array property and can correctly process the response.

Test Steps:

1. RECEIVE ReadProperty-Request,
   'Object Identifier' = (any object),
   'Property Identifier' = (any valid array property of the specified object),
   'Array Index' = 0
2. TRANSMIT BACnet-ComplexACK-PDU,
   'Object Identifier' = (object identifier from step 1),
   'Property Identifier' = (property identifier from step 1),
   Array Index' = 0,
   'Property Value' = (any valid size for the array)
3. CHECK (that the IUT exhibits the vendor defined results)

135.1-2009*i*-5. Add Tests for Processing of GetEventInformation Service Responses.

Rationale

Update the GetEventInformation service initiation tests to provide a response to the IUT and verify that the IUT properly processes the responses.

[Change **Clause 8.8.1,** p. 158]

8.8.1 Without Chaining

Purpose: To verify that the IUT can initiate a simple GetEventInformation service request.

Test Steps:

1. RECEIVE GetEventInformation-Request,
   'Last Received Object Identifier' = (none)
2. *TRANSMIT BACnet-ComplexACK-PDU,*
   'List of Event Summaries' = (any valid list)
   'More Events' = FALSE
3. *CHECK(that the IUT exhibits the vendor defined results)*

[Change **Clause 8.8.2,** p. 158]

8.8.2 With Chaining

Purpose: To verify that the IUT can respond to the "more events" flag by initiating a GetEventInformation service request with the chaining parameter "last received object identifier" present.

Test Steps:

1. RECEIVE GetEventInformation-Request,
   'Last Received Object Identifier' = (none)
2. TRANSMIT BACnet-ComplexACK-PDU,
   'List of Event Summaries' = (a non-empty list)
   'More Events' = TRUE
3. BEFORE the tester's patience is exceeded
   RECEIVE GetEventInformation-Request,
   'Last Received Object Identifier' = (last object identifier sent in step 2)
4. *TRANSMIT BACnet-ComplexACK-PDU,*
   'List of Event Summaries' = (a non-empty list)
   'More Events' = FALSE
5. *CHECK(that the IUT exhibits the vendor defined results)*

135.1-2009*i*-6. Add Tests for Fallback from ReadPropertyMultiple to ReadProperty.

Rationale

Add tests to verify that IUTs which initiate ReadPropertyMultiple are able to use ReadProperty instead when the other does not support ReadPropertyMultiple.

[Add new **Clause 8.20.Y1,** p. 167]

8.20.Y1 Cases In Which ReadProperty Shall Be Used After ReadPropertyMultiple Fails

The tests defined in this clause are used to verify that an IUT which intiates ReadPropertyMultiple is able to obtain external property values via the ReadProperty service when interoperating with a device that does not support the ReadPropertyMultiple service.

8.20.Y1.1 The IUT Determines the TD does not Support the ReadPropertyMultiple Service

Purpose: Verifies the IUT's ability to automatically change its service choice from ReadPropertyMultiple to ReadProperty when the IUT determines the TD does not support the ReadPropertyMultiple service.

Test Concept: The IUT is configured in a manner that would normally cause it to access one or more properties in the TD via the ReadPropertyMultiple service. Prior to sending a ReadPropertyMultiple request, however, the IUT determines that the TD does not support the ReadPropertyMultiple service. The IUT instead attempts to access the TD's property values via the ReadProperty service (it is assumed that the IUT will make this determination by reading the TD's Protocol_Services_Supported property, but this test specifically does not attempt to verify this behavior).

Configuration Requirements: The TD is configured so that it does not support the ReadPropertyMultiple service. The IUT is configured such that it is capable of accessing one or more properties of a single object in the TD via the ReadProperty and ReadPropertyMultiple services. If the IUT cannot be configured in this way, then this test shall be omitted.

Test Steps:

1. MAKE (a condition in the IUT that would normally cause it to send a ReadPropertyMultiple request to the TD to access one or more properties values of a single object)
2. WAIT (a time interval specified by the vendor as sufficient for the IUT to determine that the TD does not support the ReadPropertyMultiple service)
3. REPEAT X = (the properties that the IUT is to read) DO {
   RECEIVE ReadProperty-Request,
   'Object Identifier' = (object identifier referenced by X),
   'Property Identifier' = (property identifier referenced by X)
   TRANSMIT ReadProperty-Ack,
   'Object Identifier' = (object identifier referenced by X),
   'Property Identifier' = (property identifier referenced by X),
   'Property Value' = (any valid value)
   }

8.20.Y1.2 Fallback to ReadProperty on Reject - UNRECOGNIZED_SERVICE Response

BACnet Reference Clauses: 15.5 and 15.7

Purpose: Verifies the IUT's ability to automatically change its service choice from ReadPropertyMultiple to ReadProperty when the TD returns a Reject-PDU and a Reject Reason of UNRECOGNIZED_SERVICE.

Test Concept: The IUT is configured to send the TD a ReadPropertyMultiple request to access one or more properties of a single object. The TD responds with a Reject-PDU and a Reject Reason of UNRECOGNIZED_SERVICE. With no additional configuration, the IUT sends one or more ReadProperty requests to the TD, where each ReadProperty request specifies an individual property from the original ReadPropertyMultiple request.

Configuration Requirements: The TD is configured so that it does not support the ReadPropertyMultiple service. The IUT is configured such that it attempts to acquire values from the TD using the ReadPropertyMultiple service without first interrogating the TD's Protocol_Services_Supported property. If the IUT cannot be configured in this way then this test shall be omitted.

Test Steps:

1. RECEIVE ReadPropertyMultiple-Request,
   'Object Identifier' = (object identifier of the specified object),
   'List of Property References' = (one or more properties of the specified object)
2. TRANSMIT BACnet-Reject-PDU,
   'Reject Reason' = UNRECOGNIZED_SERVICE
3. REPEAT X = (the properties from Step 1) DO {
   RECEIVE ReadProperty-Request,
   'Object Identifier' = (object identifier referenced by X),
   'Property Identifier' = (property identifier referenced by X)
   TRANSMIT ReadProperty-Ack,
   'Object Identifier' = (object identifier referenced by X),
   'Property Identifier' = (property identifier referenced by X),
   'Property Value' = (any valid value)
   }

135.1-2009*i*-7. Allow Priorities in WriteProperty and WritePropertyMultiple Tests.

Rationale

These changes allow an IUT to include a priority when generating WriteProperty and WritePropertyMultiple service requests.

This note is to be applied to all tests in the standard and all addenda where the IUT is expected to generate a WriteProperty or WritePropertyMultiple request that is not already expecting a Priority parameter to be present.

[Append to **Clause 7.3.2.9.3,** p. 59, **Clause 7.3.2.23.8,** p. 93, **Clause 8.22.1,** p. 168, **Clause 8.22.2,** p. 169 and to any clause in any addendum that expects the IUT to generate an WriteProperty and does not expect a Priority parameter.]

Note to Tester: Any WriteProperty request generated by the IUT may have a Priority parameter. If included, it shall be in the range 1-16, excluding 6.

[Append to **Clause 8.23.1,** p. 169, **Clause 8.23.2,** p. 169, **Clause 8.23.3,** p. 170, **Clause 8.23.4,** p. 170, and **Clause** 8.23.5, p. 170 and to any clause in any addendum that expects the IUT to generate an WritePropertyMultiple and does not expect a Priority parameter.]

Note to Tester: Any WritePropertyMultiple request generated by the IUT may have a Priority parameter for any of the properties written. If included, the Priority parameter shall be in the range 1-16, excluding 6 .

135.1-2009*i*-8. Add Test for Writing Array Size.

Rationale

Add in a test that verifies that the IUT can modify the size of an array.

[Add new **Clause 8.22.X1**, p. 169]

8.22.X1 Writing An Array Size

Purpose: To verify that the IUT can initiate WriteProperty service requests that modify the size of an array.

Test Steps:

1. RECEIVE WriteProperty-Request,
   'Object Identifier' = (any valid object identifier),
   'Property Identifier' = (any valid array property of the specified object),
   'Array Index' = 0
   'Property Value' = (any unsigned value)

Note to Tester: Any WriteProperty request generated by the IUT may have a Priority parameter. If included, it shall be in the range 1-16, excluding 6.

135.1-2009*i*-9. Clarify Test for Writing with a Value that is Out of Range.

Rationale

Clarify the requirements for a value that is out of range and ensure that the tests are executed against writable properties.

[Change **Clause 9.22.2.4**, p. 269]

9.22.2.4 Writing with a Property Value that is Out of Range

Purpose: To verify that the IUT can execute WriteProperty service requests when an attempt is made to write a value that is outside of the supported range.

Test Concept: The TD attempts to write to a property using a value that is outside of the supported range. *If the IUT does not contain any writable properties that have restricted ranges, then this test shall be skipped.*

Test Steps:

1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS),
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (Object1, any object with writable properties),
   'Property Identifier' = (P1, any *writable* property with a restricted range of values),
   'Property Value' = (any value that is outside the supported range)
3. IF (Protocol_Revision is present and Protocol_Revision ≥ 4) THEN
   RECEIVE (BACnet-Error PDU,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE
   ELSE
   RECEIVE (BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE) |
   (BACnet-Reject-PDU,
   Reject Reason = PARAMETER_OUT_OF_RANGE)
4. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)

Notes to Tester: The value used in step 2 shall be of the correct datatype. For bit string types, the bit count shall be correct for Date and Time values, the value shall be within the range defined by the standard for the datatype and for constructed values, the constructed value shall match the structure defined by the ASN.1, and all field values shall be within the ranges defined by the standard for those field values.

[Add new **Clause 9.23.1.X**, p. 274]

9.23.1.X Writing to Properties Based on Data Type

Purpose: This test case verifies that the IUT can execute WritePropertyMultiple service requests to any data type supported by the IUT.

Test Concept: For the specified data type, the TD shall select an object in the IUT that contains a writable property of that data type. This property is designated P1.

Configuration Requirements: Configure the IUT with at least one writable property of the data type that can be used for this test.

Test Steps:

1. READ V = Object1, P1
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any valid value defined for this property subject to the restrictions specified in the EPICS as defined in Clause 4.4.2, except the value, V, read in step 1)
3. RECEIVE Simple-ACK-PDU
4. VERIFY (Object1), P1 = (the value used in step 2)

135.1-2009*i*-10. Update Test for Writing with an Invalid Datatype.

Rationale

Update the invalid datatype tests to take into account the committee's response to IC-2004-28 and changes made in Protocol_Revision 7.

The tests are also updated to not rely on the EPICS value and to use the new READ keyword.

[Change **Clause**, **9.22.2.3**, p. 268]

9.22.2.3 Writing with a Property Value Having the Wrong Datatype

Purpose: To verify that the IUT correctly responds to an attempt to write a property value that has an invalid datatype.

Test Concept: The TD shall select an object in the IUT that contains a writable property designated P1. An attempt will be made to write to this property using ~~an invalid datatype~~ *a datatype that the IUT supports but which is invalid for the property*. If no object supports writable properties, then this test shall be omitted.

Test Steps:

~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~
1. READ V = (Object1), P1
2. TRANSMIT WriteProperty-Request,
'Object Identifier' = Object1,
'Property Identifier' = P1,
'Property Value' = (any value with an invalid datatype)
3. ~~IF (Protocol_Revision is present and Protocol_Revision ≥ 4) THEN~~
~~RECEIVE BACnet-Error PDU,~~
~~Error Class = PROPERTY,~~
~~Error Code = INVALID_DATATYPE~~
~~ELSE~~
RECEIVE (BACnet-Error PDU,
Error Class = PROPERTY,
Error Code = INVALID_DATATYPE) |
(BACnet-Reject-PDU
Reject Reason = INVALID_PARAMETER_DATATYPE) |
(BACnet-Reject-PDU
Reject Reason = INVALID_TAG)
4. VERIFY (Object1), P1 = *V*~~(the value defined for this property in the EPICS)~~

[Change **Clause 9.23.2.6**, p. 277]

9.23.2.6 Writing with a Property Value Having the Wrong Datatype

Purpose: This test case verifies that the IUT correctly responds to an attempt to write a property value that has an invalid datatype. ~~This test shall only be performed if Protocol_Revision is present and has a value greater than or equal to 4.~~

Test Concept: The TD shall select an object, designated Object1, in the IUT that contains a writable property designated P1. An attempt will be made to write to this property using ~~an invalid datatype~~ *a datatype that the IUT supports but which is invalid for the property*. If no object supports writable properties, then this test shall be omitted.

Test Steps:

~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~

1. *READ V = (Object1), P1*
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any value with an invalid datatype)
3. RECEIVE
   WritePropertyMultiple-Error,
   'Error Class' = PROPERTY,
   'Error Code' = INVALID_DATATYPE
   'Object Identifier' = Object1
   'Property Identifier' = P1
   | *(BACnet-Reject-PDU*
   'Reject Reason' = INVALID_PARAMETER_DATATYPE)
   | *(BACnet-Reject-PDU*
   'Reject Reason' = INVALID_TAG)
4. VERIFY (Object1), P1 = *V*~~(the value defined for this property in the EPICS)~~

[Change **Clause 9.14.2.3**, p 245]
[Step number changes are not change marked for clarity.]

9.14.2.3 An AddListElement Failure Part Way Through a List

Purpose: To verify the ability of the IUT to respond to an AddListElement service request to add multiple elements to a list where one of the elements cannot be added. Upon failure, the AddListElement service should leave the list unchanged.

Test Steps:

~~1. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = L,~~
~~'Property Identifier' = ListProp~~
~~2. RECEIVE ReadProperty-ACK,~~
~~'Object Identifier' = L,~~
~~'Property Identifier' = ListProp,~~
~~'Property Value' = (any valid value referred to as "InitialList" below)~~

1. *READ InitialList = (L), ListProp*
2. TRANSMIT AddListElement-Request,
   'Object Identifier' = *(L)*,
   'Property Identifier' = ListProp,
   'List of Elements' = (two or more elements to be added to the list with the second element having the wrong datatype)
3. *IF (Protocol_Revision is present and Protocol_Revision ≥ 7) THEN*
   RECEIVE AddListElement-Error,
   Error Class = PROPERTY,
   Error Code = INVALID_DATATYPE
   'First Failed Element' = 2
   ELSE
   RECEIVE AddListElement-Error,
   Error Class = SERVICES,
   Error Code = INVALID_PARAMETER_DATATYPE
   'First Failed Element' = 2
   | *(AddListElement-Error,*
   Error Class = PROPERTY,
   Error Code = INVALID_DATATYPE)
4. VERIFY (L), ListProp = InitialList

[Change **Clause 9.15.2.2**, p 247]

9.15.2.2 A RemoveListElement Failure Part Way Through a List

Purpose: To verify the ability of the IUT to respond to a RemoveListElement service request to remove multiple elements from a list where one of the elements cannot be removed. Upon failure, the RemoveListElement service should leave the list unchanged.

Test Steps:

~~1. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = L,~~
~~'Property Identifier' = ListProp~~
~~2. RECEIVE ReadProperty-ACK,~~
~~'Object Identifier' = L,~~
~~'Property Identifier' = ListProp,~~
~~'Property Value' = (any valid value referred to as "InitialList" below)~~

1. *READ InitialList = (L), ListProp*
2. TRANSMIT RemoveListElement-Request,
   'Object Identifier' = *(*L*)*,
   'Property Identifier' = *(*ListProp*),*
   'List of Elements' = (one element from InitialList, followed by an element of the correct datatype that is not in InitialList, followed by one or more elements from InitialList)
3. *IF (Protocol_Revision is present and Protocol_Revision ≥ 7) THEN*
   RemoveListElement-Error,
   Error Class = *PROPERTY,*
   Error Code = *INVALID_DATA_TYPE*
   'First Failed Element' = *2*
   ELSE
   RECEIVE RemoveListElement-Error,
   Error Class = SERVICES | PROPERTY,
   Error Code = OTHER
   'First Failed Element' = 2
   | *(RemoveListElement-Error,*
   Error Class = *PROPERTY,*
   Error Code = *INVALID_DATA_TYPE*
   'First Failed Element' = *2)*
4. VERIFY (L), ListProp = InitialList

135.1-2009*i*-11 Relax ReadPropertyMultiple Error Test.

Rationale

Allow the IUT to return a ComplexACK with the error indicated instead of a BACnet-Error-PDU.

[Change **Clause 9.20.1.1**, p 262]

9.20.2.1 Reading a Single, Unsupported Property from a Single Object

Purpose: To verify the ability to correctly execute a ReadPropertyMultiple service request for which the 'List of Read Access Specifications' contains specifications for a single unsupported property.
…

Test Steps:

1. TRANSMIT ReadPropertyMultiple-Request,
   'Object Identifier' = Object1 | Object2,
   'Property Identifier' = (any property, P1, that is not supported in the selected object)
2. RECEIVE BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = UNKNOWN_PROPERTY
   | *(ReadPropertyMultiple-ACK,*
   'Object Identifier' = (the object identifier from step 1),
   'Property Identifier' = P1,
   'Error Class' = PROPERTY,
   'Error Code' = UNKNOWN_PROPERTY)

135.1-2009*i*-12 Add Test for Unicast Who-Is.

Rationale

Add a test that tests unicast Who-Is initiation.

[Add new **Clause 8.34.X1,** p. 178]

8.34.X1 Who-Is Request with no Device Instance Range

Purpose: To verify that the IUT can initiate unicast Who-Is service requests with no device instance range. If the IUT cannot be caused to issue a Who-Is request of this form, then this test shall be omitted.

Test Steps:

1. RECEIVE
   DESTINATION = TD,
   SOURCE = IUT,
   Who-Is-Request

135.1-2009*i*-13 Revise Unknown Network Layer Message Test.

Rationale

Change test 10.2.2.7.2 to match the result of IC-2004-12.

[Change **Clause 10.2.2.7.2**, p. 323]

10.2.2.7.2 Unknown Network Layer Message Type

Purpose: To verify that the IUT will reject a network layer message ~~with an unknown message type~~ *directed to the IUT that contains an unknown message type in the range of message types reserved for use by ASHRAE.*

Test Steps:

1. TRANSMIT PORT A,
   DESTINATION = IUT,
   SOURCE = ~~D1A~~ *TD*,
   Message Type = (any value ~~from X'0A' to X'7F~~ *in the range reserved for use by ASHRAE that is undefined in the protocol revision claimed by the device*)
2. RECEIVE PORT A,
   DESTINATION = ~~D1A~~ *TD*,
   SOURCE = IUT,
   Reject-Message-To-Network,
   Reject Reason = 3 (unknown network layer message type),
   DNET = ~~1~~ *any value*

135.1-2009*i*-14 Add New Trend Log Tests.

Rationale

A collection of new Trend Log tests are added to expand upon the set of testable functionality and existing tests are changed to work on all log object types, where applicable (Trend Log, Trend Log Multiple, and Event Log).

[Add new **Clause 7.3.2.24.X1**, p. 105]

7.3.2.24.X1 Log-Status Test

Dependencies: ReadRange Service Execution Tests, 9.21; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clause: 12.25.14.

Purpose: To verify proper logging of log-disabled and buffer-purged events.

Test Concept: The buffer is cleared. Then the Enable property is changed and it is verified that the Record_Count property is changed and it is verified that the status entry is made correctly in the Log_Buffer. The Record_Count is also set to zero while the Enable property is FALSE and it is verified that the buffer-purged event is recorded into the Log_Buffer.

Test Configuration: The Trend Log, O1, is configured to acquire data by whatever means available. Configure the logging such that the entire test may be run without the trend buffer overflowing.

Test Steps:

1. WRITE Enable = FALSE
2. WRITE Record_Count = 0
3. VERIFY Record_Count = 1
4. TRANSMIT ReadRange
   - 'Object Identifier' = O1,
   - 'Property Identifier' = Log_Buffer,
   - 'Reference Index' = 1,
   - 'Count' = 1
5. RECEIVE ReadRange-Ack
   - 'Object Identifier' = O1,
   - 'Property Identifier' = Log_Buffer,
   - 'Result Flags' = (True, True, False),
   - 'Item Count' = 1
   - 'Item Data' = ((a buffer purged record))
6. WRITE Enable = TRUE
7. WRITE Enable = FALSE
8. TRANSMIT ReadRange
   - 'Object Identifier' = O1,
   - 'Property Identifier' = Log_Buffer,
   - 'Reference Index' = 1,
   - 'Count' = 2
9. RECEIVE ReadRangeAck
   - 'Object Identifier' = O1,
   - 'Property Identifier' = Log_Buffer,
   - 'Result Flags' = (True, False, False),
   - 'Item Count' = 2
   - 'Item Data' = ((a buffer purged record), (a log-enable record))
10. TRANSMIT ReadRange

'Object Identifier' = O1,
'Property Identifier' = Log_Buffer,
'Reference Time' = (2154-12-31, 23:59:59.99),
'Count' = -1

11. RECEIVE ReadRangeAck
'Object Identifier' = O1,
'Property Identifier' = Log_Buffer,
'Result Flags' = (False, True, False),
'Item Count' = 1
'Item Data' = ((a log-disable record))

[Add new **Clause 7.3.2.24.X2**, p. 105]

7.3.2.24.X2 Time_Change Test

Dependencies: ReadRange Service Execution Tests, 9.21; (TimeSynchronization Service Execution Tests, 9.30 or UTCTimeSynchronization Service Execution Tests, 9.31)

BACnet Reference Clause: 12.25.14.

Purpose: To verify proper logging of time-change events in the log buffer

Test Concept: Change the clock in the device and verify that a record is logged indicating the number of seconds that the clock changed by or indicating zero if unknown. This test shall be skipped if the device does not support the Local_Time property in the device object or there is no way to change the time in the device.

Test Configuration: The log is configured to acquire data by whatever means available. The Log_Buffer should be cleared such that the Record_Count is 0. Configure the logging such that the entire test may be run without the trend buffer overflowing.

Test Steps:

1. WRITE Enable = FALSE
2. WRITE Record_Count = 0
3. READ currentTime = (Device Object of device that contains the log object), Local_Time
4. WRITE Enable = TRUE
5. MAKE(the time change on the device by deltaTime where deltaTime >= 1 hour)
6. WRITE Enable = FALSE
7. READ N = Record_Count
8. REPEAT X = (N down through 1) DO {
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, True, False),
'Item Count' = 1,
'Item Data' = ((a record. If the record is a time-change record, save the timestamp into TS and the time-change value into TC))
}
9. CHECK (TC ~= deltaTime)
10. CHECK (TS ~= currentTime + deltaTime)

[Add new **Clause 7.3.2.24.X3**, p. 105]

7.3.2.24.X3 COV-Sampling Verification Test

Dependencies: ReadRange Service Execution, 9.21

BACnet Reference Clause: 12.25.10-11; 13.1

Purpose: To verify logged samples are based on COV rather than by interval.

Test Concept: The trend log is configured to log based on COV increment. Logging is enabled. After a period of time the buffer is checked to verify that the data in the buffer is based on the COV values and not on the set interval.

Configuration Requirements: The IUT shall be configured such that the monitored object has a COV_Increment property that is set to a value other than 0.0, the Client_COV_Increment is set to a value other than 0.0 or NULL, or the monitored property is not of datatype REAL.

Test Steps:

1. WRITE Enable = FALSE
2. WRITE Record_Count = 0
3. WRITE Interval = 0
4. WRITE Enable = TRUE
5. WAIT (10 seconds)
6. MAKE (monitored property change its value)
7. WAIT (60 seconds)
8. MAKE (monitored property change its value)
9. WAIT (90 seconds)
10. MAKE (monitored property change its value)
11. WAIT (40 seconds)
12. MAKE (monitored property change its value)
13. WAIT Notification Fail Time
14. WRITE Enable = FALSE
15. READ N = RecordCount
16. REPEAT X = (1 through 4) {
    TRANSMIT ReadRange
    'Object Identifier' = O1,
    'Property Identifier' = Log_Buffer,
    'Reference Index' = N-5+X,
    'Count' = 1
    RECEIVE ReadRangeAck
    'Object Identifier' = O1,
    'Property Identifier' = Log_Buffer,
    'Result Flags' = (False, False, False),
    'Item Count' = 1
    'Item Data' = ((one data record storing the timestamp in TS[X]))
    }
17. CHECK(TS[2] - TS[1] ~= 60 seconds)
18. CHECK(TS[3] - TS[2] ~= 90 seconds)
19. CHECK(TS[4] - TS[3] ~= 40 seconds)

[Add new **Clause 7.3.2.24.X4**, p. 105]

7.3.2.24.X4 Interval Gathering of External Trends Test

Purpose: To verify the IUT uses ReadProperty to pull external data at the specified intervals.

Test Concept: The log is configured to acquire data from the TD using polling. The TD verifies that the receipt of ReadProperty requests are at the Log_Interval set.

Configuration Requirements: The log shall be configured to be polling for external trends during the entire time of this test. The Stop_When_Full property, if configurable, shall be set to FALSE. Enable shall be set to TRUE. The TD shall be configured so that it does not support execution of ReadPropertyMultiple.

Test Steps:

1. BEFORE (Log_Interval)
   REPEAT X = (for each property logged) DO
   RECEIVE ReadProperty-Request,
   'Object Identifier' = (object that contains the monitored property)
   'Property Identifier' = (external property that is being trended)
   TRANSMIT ReadProperty-Ack
   'Object Identifier' = (object that contains the monitored property)
   'Property Identifier' = (property being monitored)
   'Propert y Value' = (any value)
2. WAIT (Log Interval)
3. REPEAT X = (for each property logged) DO
   RECEIVE ReadProperty-Request,
   'Object Identifier' = (object that contains the monitored property)
   'Property Identifier' = (external property that is being trended)
   TRANSMIT ReadProperty-Ack
   'Object Identifier' = (object that contains the monitored property)
   'Property Identifier' = (property being monitored)
   'Property Value' = (any value)
4. CHECK (to ensure all properties logged are requested)

[Add new **Clause 7.3.2.24.X5**, p. 105]

7.3.2.24.X5 Last_Notify_Record Test

Dependencies: ReadProperty Service Execution Tests, 9.15; WriteProperty Service Execution Tests, 9.18.

BACnet Reference Clauses: 12.25.19 and 12.25.26

Purpose: To verify that the Last_Notify_Record property reflects the values sent in the most recent notification.

Test Concept: The log buffer is cleared. The Log is allowed to collect records until it issues a BUFFER_READY notification. The value of the Last_Notify_Record property is checked.

Test Configuration: The log is configured to send BUFFER_READY notifications to the TD.

Test Steps:

1. WRITE Record_Count = 0
2. READ prev = Last_Notify_Record
3. WRITE Enable = TRUE
4. MAKE (Log object collect number of records specified by Notification_Threshold)

5. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (the configured process identifier)
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (Log object being tested),
   'Time Stamp' = (any appropriate BACnetTimeStamp value),
   'Notification Class' = (configured notification class),
   'Priority' = (value configured to correspond to a TO-NORMAL),
   'Event Type' = BUFFER_READY,
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'FromState' = NORMAL,
   'To State' = NORMAL,
   'Event Values' = ((the log object), Log_Buffer,
   previous-notification = prev,
   current-notification, C1)
6. TRANSMIT BACnet-SimpleACK-PDU
7. WRITE Enable = FALSE
8. VERIFY Last_Notify_Record = C1

[Add new **Clause 7.3.2.24.X6**, p. 105]

7.3.2.24.X6 Records_Since_Notification Test
Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clauses: 12.25.18, 12.25.19, 12.25.22

Purpose: To verify that the Records_Since_Notification property reflects the number of records recorded by the log that have not yet been reported via a BUFFER_READY notification.

Test Concept: The log buffer is cleared. The log is allowed to collect records until it issues a BUFFER_READY notification and is halted before a second notification is generated. The value of the Records_Since_Notification property is checked.

Test Configuration: The Trend Log is configured to send BUFFER_READY notifications to the TD.

Test Steps:

1. WRITE Enable = TRUE
2. WRITE Record_Count = 0
3. MAKE (Log object collect a sufficient number of records in order to trigger a notification)
4. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (the configured process identifier)
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (Log object being tested),
   'Time Stamp' = (Any appropriate BACnetTimeStamp value),
   'Notification Class' = (configured notification class),
   'Priority' = (value configured to correspond to a TO-NORMAL),
   'Event Type' = BUFFER_READY,
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'FromState' = NORMAL,
   'To State' = NORMAL,
   'Event Values' = ((the log object), Log_Buffer),
   previous-notification,
   current-notification, C1)
5. TRANSMIT BACnet-SimpleACK-PDU

6. MAKE (Log object collect N records, such that N < Notification_Threshold-1)
7. WRITE Enable = FALSE
8. READ T1 = Total_Record_Count
9. VERIFY Records_Since_Notification = T1 - C1

[Add new **Clause 7.3.2.24.X7**, p. 105]

7.3.2.24.X7 Trigger Verification Test

Dependencies: ReadRange Service Execution, 9.21;

BACnet Reference Clause: 12.25.27, 12.30.12

Purpose: To verify that 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 that 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
3. WRITE Enable = TRUE
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
12. READ N = RecordCount
13. REPEAT X = (1 through 4)
    TRANSMIT ReadRange
    'Object Identifier' = O1,
    'Property Identifier' = Log_Buffer,
    'Reference Index' = N-4+X,
    'Count' = 1
    RECEIVE ReadRangeAck
    'Object Identifier' = O1,
    'Property Identifier' = Log_Buffer,
    'Result Flags' = (False, False, False),
    'Item Count' = 1
    'Item Data' = ((one data record storing the timestamp in TS[X]))
16. CHECK(TS[2] - TS[1] ~= 20 seconds)
17. CHECK(TS[3] - TS[2] ~= 40 seconds)
18. CHECK(TS[4] - TS[3] ~= 30 seconds)

[Change **Clauses 9.21.1.1** through **9.21.1.4**, p. 264]

9.21.1.1 Reading All Items in the List

Purpose: To verify that the IUT correctly responds to a ReadRange service request to return all of the available data items.

Test Steps:

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (the ~~Trend Log~~*log* object configured for this test),
   'Property Identifier' = Log_Buffer
2. RECEIVE Read-Range-ACK,
   'Object Identifier' = (the ~~Trend Log~~*log* object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {TRUE, TRUE, FALSE},
   'Item Count' = (the number of records in the test object),
   'Item Data' = (all of the records in the test object)

Notes to Tester: The log data may have more items than can be returned in a single message. Under these circumstances 'Result Flags' will have the value {TRUE, FALSE, TRUE} and the 'Item Count' and 'Item Data' parameters will reflect the actual number of items that were able to be returned.

9.21.1.2 Reading Items by Position with Positive Count

Purpose: To verify that the IUT correctly responds to a ReadRange service request to return items specified by indicating a position and the number of items after that position to return.

Test Concept: A ReadRange request is transmitted by the TD requesting a range of items known to be in the Log_Buffer. This range is specified using the 'By Position' option and a positive value for 'Count'. The 'Reference Index' and 'Count' are selected so that the results can be conveyed in a single acknowledgment.

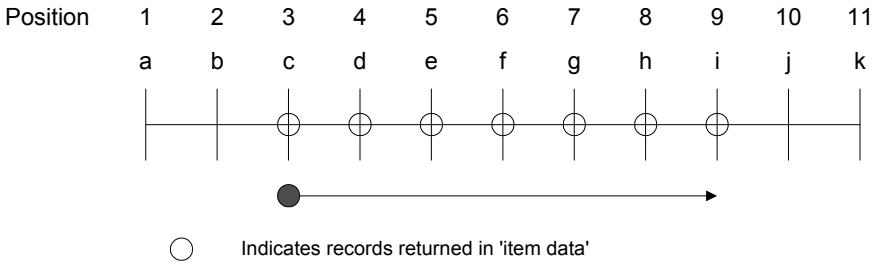
Test Steps:
1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (the ~~Trend Log~~*log* object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Reference Index' = (any value x: $1 \le x \le$ (the number of trend records in the buffer – y + 1)),
   'Count' = (any value y: $0 < y \le$ (the number of trend records in the buffer – x + 1))
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (the ~~Trend Log~~*log* object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {?,?, FALSE},
   'Item Count' = y,
   'Item Data' = (all of the specified records in order of increasing position. The items specified include the item at the index specified by x, plus (y-1) items following.)

Test Example (using the sample buffer at beginning of section) :

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Reference Index' = 3,

'Count' = 7

2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {FALSE, FALSE, FALSE},
   'Item Count' = 7,
   'Item Data' = Records < c, d, e, f, g, h, i > in that order.



9.21.1.3 Reading Items by Position with Negative Count

Purpose: To verify that the IUT correctly responds to a ReadRange service request to return items specified by indicating a position and the number of items before that position to return.

Test Concept: A ReadRange request is transmitted by the TD requesting a range of items known to be in the Log_Buffer. This range is specified using the 'By Position' option and a negative value for 'Count'. The 'Reference Index' and 'Count' are selected so that the results can be conveyed in a single acknowledgement.

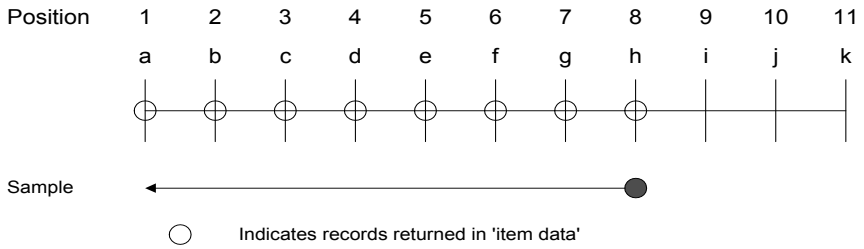
Test Steps:

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (the ~~Trend Log~~*log* object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Reference Index' = (any value x: $2 \le x \le$ the number of trend records in the buffer),
   'Count' = (any value y: $y < 0$ AND $|y| \le x$)
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (the ~~Trend Log~~*log* object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {FALSE, ?, FALSE},
   'Item Count' = |y|,
   'Item Data' = (all of the specified trend records in order of increasing position. The items specified include the item at the index specified by x, plus |y|-1 items preceding.)

Test Example (using the sample buffer at beginning of section):

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Reference Index' = 8,
   'Count' = -8
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {TRUE, FALSE, FALSE},
   'Item Count' = 8

'Item Data' = Records < a, b, c, d, e, f, g, h > in that order.



9.21.1.4 Reading Items by Time

Purpose: To verify that the IUT correctly responds to a ReadRange service request to return items specified by indicating a time and the number of items after that time to return.

Test Concept: A ReadRange request is transmitted by the TD requesting a range of items known to be in the Log_Buffer. This range is specified using the 'By Time' option and a positive value for 'Count'. The 'Reference Index' and 'Count' are selected so that the results can be conveyed in a single acknowledgement.

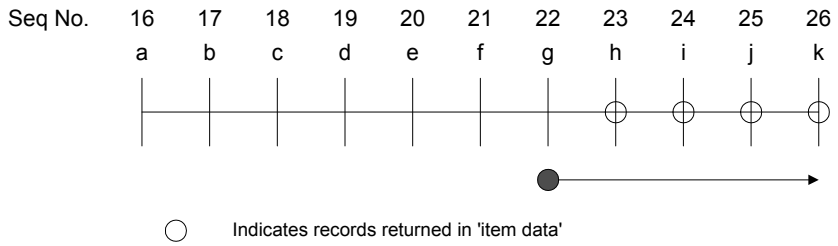
Test Steps:

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (the ~~Trend Log~~*log* object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Reference Time' = (any value x: x is older than (of earlier time) the last time in the buffer),
   'Count' = (any value y: y > 0)
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (the ~~Trend Log~~*log* object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {?, ?, FALSE},
   'Item Count' = min(Total_Record_Count - z + 1, y),
   'Item Data' =('Item Count' trend records in order of increasing sequence number, starting with the record having sequence number z),
   'First Sequence Number' = (Total_Record_Count – y + 1)

Notes to Tester: The first item returned shall be the entry in the Log_Buffer with a timestamp newer (later time) than the time specified by the 'Reference Time' parameter. If there is an entry in the Log_Buffer with a timestamp that exactly matches the 'Reference Time' parameter, that entry should not be included in the 'Item Data'.

Test Example (using sample buffer at beginning of section):

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Reference Time' = 13:21:00.00
   'Count' = 4
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {FALSE, TRUE, FALSE},
   'Item Count' = 4,
   'Item Data' = Records < h, i, j, k > in that order.
   'First Sequence Number' = 23 (26 – 4 + 1)



[Add new **Clause 9.21.1.4.X1**, p. 264]

9.21.1.4.X1 Reading Items by Time with Negative Count

Purpose: To verify that the IUT correctly responds to a ReadRange service request to return items specified by indicating a time and the number of items after that time to return.

Test Concept: A ReadRange request is transmitted by the TD requesting a range of items known to be in the Log_Buffer. This range is specified using the 'By Time' option and a negative value for 'Count'. The 'Reference Time' selected, x, should be newer than the last time in the buffer. The 'Reference Time' and 'Count' are selected so that the results can be conveyed in a single acknowledgement.

Test Configuration: Configure the TD such that no time change requests occur. Configure the TD such that it contains at least 3 items in the Log_Buffer.

Test Steps:

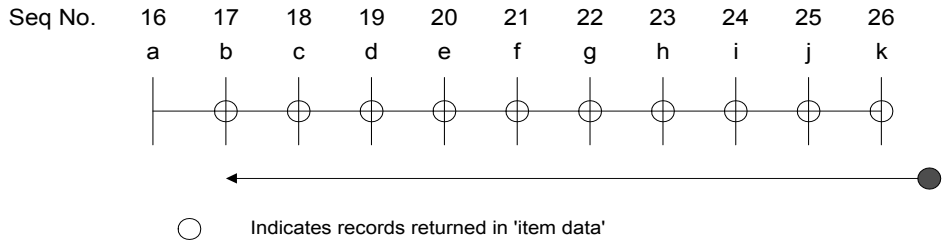
1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (the log object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Reference Time' = (x, selected as described above),
   'Count' = (any value y: $0 < |y| \leq$ number of records in the buffer)
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (the log object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {?, TRUE, FALSE},
   'Item Count' = |y|,
   'Item Data' = (All of the specified trend records in order of increasing sequence number. The items specified include the last item with a timestamp older than x, plus |y|-1 items preceding.)
   'First Sequence Number' = (Total_Record_Count- |y| + 1)

Notes to Tester: All items returned shall contain a timestamp older than the time specified by reference time parameter. The items returned shall be the last 'count' items from the log buffer. If there is an entry in the Log_Buffer with a timestamp that exactly matches the 'Reference Time' parameter, that entry shall not be included in the 'Item Data'.

Test Example (using the sample buffer at beginning of section):

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Reference Time' = 13:40:00.00,
   'Count' = -10
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (Trend Log, Instance 1),

'Property Identifier' = Log_Buffer,
'Result Flags' = {FALSE, TRUE, FALSE},
'Item Count' = 10,
'Item Data' = (records < b, c, d, e, f, g, h, i, j, k > in that order.)
'First Sequence Number' = 17



[Add new **Clause 9.21.1.6.X1**, p. 265]

9.21.1.6.X1 Reading a Range of Items that do not Exist (Using by Sequence)

Purpose: To verify that the IUT correctly responds to a ReadRange service request when there are no items within the specified criteria.

Test Concept: A ReadRange request is transmitted by the TD requesting a specified sequence number and count of items known not to be in the Log_Buffer. The IUT shall respond by returning an empty list.

Test Steps:

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (the log object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Reference Sequence Number' = (any value that will result in no items being present)
   'Count' = (any non-zero number)
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (the log object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Result flags' = {FALSE, FALSE, FALSE},
   'Item Count' = 0,
   'Item Data' = (an empty list)
   'First Sequence Number' = (should be absent)

Test Example (using sample buffer at beginning of section):

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Reference Sequence Number' = 34
   'Count' = 4
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Result flags' = {FALSE, FALSE, FALSE},
   'Item Count' = 0,
   'Item Data' = (an empty list)

[Add new **Clause 9.21.1.6.X2**, p. 265]

9.21.1.6.X2 Reading a Range of Items that do Not Exist (Using by Time)

Purpose: To verify that the IUT correctly responds to a ReadRange service request when there are no items within the specified criteria.

Test Concept: A ReadRange request is transmitted by the TD requesting a specified reference time and count of items known not to be in the Log_Buffer. The IUT shall respond by returning an empty list.

Test Steps:

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (the log object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Reference Time' = (any value that will result in no items being present)
   'Count' = (any non-zero number)
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (the log object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Result flags' = {FALSE, FALSE, FALSE},
   'Item Count' = 0,
   'Item Data' = (an empty list)
   'First Sequence Number' = (should be absent)

Test Example (using sample buffer at beginning of section):

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Reference Time' = 12:00:00.00
   'Count' = -10
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (Trend Log, Instance 1),
   'Property Identifier' = Log_Buffer,
   'Result flags' = {FALSE, FALSE, FALSE},
   'Item Count' = 0,
   'Item Data' = (an empty list)

[Add new **Clause 9.21.1.X1**, p. 266]

9.21.1.X1 Reading Items by Sequence with Positive Count

Purpose: To verify that the IUT correctly responds to a ReadRange service request to return items specified by indicating a sequence number and the number of items after that sequence to return.

Test Concept: A ReadRange request is transmitted by the TD requesting a range of items known to be in the Log_Buffer. This range is specified using the 'By Sequence' option and a positive value for 'Count'. The 'Reference Sequence Number' and 'Count' are selected so that the results can be conveyed in a single acknowledgment.

Test Steps:

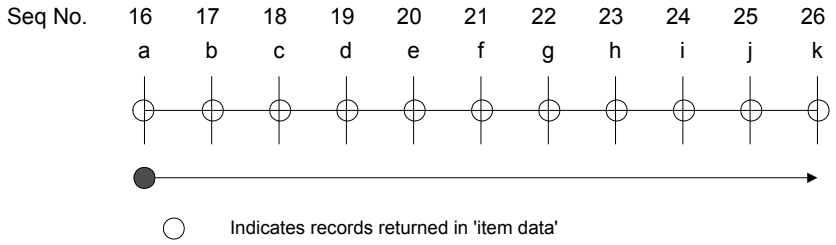
1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (the log object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Reference Sequence Number' = (any value x: (Total_Record_Count – Record_Count

+ 1) ≤ x ≤ (Total_Record_Count – y + 1)),
'Count' = (any value y: 0 < y ≤ Record_Count)

2. RECEIVE ReadRange-ACK,
'Object Identifier' = (the log object configured for this test),
'Property Identifier' = Log_Buffer,
'Result Flags' = {?, ?, FALSE},
'Item Count' = y,
'Item Data' = (All of the specified trend records in the order of increasing sequence number. The items specified are all items with the sequence number in the range of x through (x+y-1) in that order).
'First Sequence Number' = x

Test Example (using sample buffer at beginning of section):

1. TRANSMIT ReadRange-Request,
'Object Identifier' = 20:1,
'Property Identifier' = Log_Buffer,
'Reference Sequence Number' = 16,
'Count' = 11
2. RECEIVE ReadRange-ACK,
'Object Identifier' = 20:1,
'Property Identifier' = Log_Buffer,
'Result Flags' = {TRUE, TRUE, FALSE},
'Item Count' = 11,
'Item Data' = Records < a, b, c, d, e, f, g, h, i, j, k > in that order.
'First Sequence Number' = 16



[Add new **Clause 9.21.1.X2**, p. 266]

9.21.1.X2 Reading Items by Sequence with Negative Count

Purpose: To verify that the IUT correctly responds to a ReadRange service request to return items specified by indicating a sequence number and the number of items after that sequence to return.

Test Concept: A ReadRange request is transmitted by the TD requesting a range of items known to be in the Log_Buffer. This range is specified using the 'By Sequence' option and a negative value for 'Count'. The 'Reference Sequence Number' and 'Count' are selected so that the results can be conveyed in a single acknowledgment.

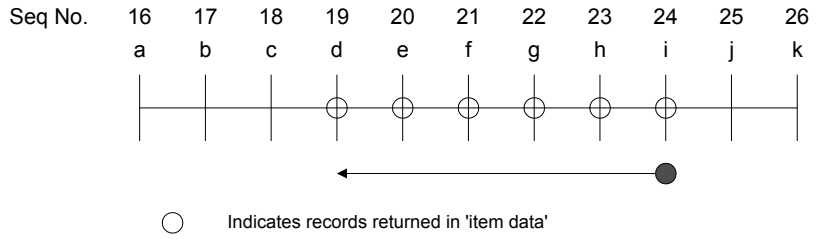
Test Steps:

1. TRANSMIT ReadRange-Request,
'Object Identifier' = (the log object configured for this test),
'Property Identifier' = Log_Buffer,
'Reference Sequence Number' =
(any value x: (Total_Record_Count - Record_Count+2) < x ≤ Total_Record_Count),
'Count' = (any value y: 0 < |y| < (Record_Count - (Total_Record_Count – x) + 1)

2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (the log object configured for this test),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {?, ?, FALSE},
   'Item Count' = y,
   'Item Data' = (All of the specified records in order of increasing sequence number. The items specified are all items in the range of (x-|y|+1) through x in that order.)
   'First Sequence Number' = (x - |y| + 1)

Test Example (using sample buffer at beginning of section):

1. TRANSMIT ReadRange-Request,
   'Object Identifier' = (Trend Log, 1),
   'Property Identifier' = Log_Buffer,
   'Reference Sequence Number' = 24,
   'Count' = -6
2. RECEIVE ReadRange-ACK,
   'Object Identifier' = (Trend Log, 1),
   'Property Identifier' = Log_Buffer,
   'Result Flags' = {FALSE, FALSE, FALSE},
   'Item Count' = 6,
   'Item Data' = Records < d, e, f, g, h, i > in that order.
   'First Sequence Number' = 19



[Add new **Clause 9.21.1.X3**, p. 266]

9.21.1.X3 Data Type Verification Test
Purpose: To verify that the ReadRange service performs properly for each data type supported.

Test Concept: Set Log_DeviceObjectProperty to an external property and verify that ReadRange performs properly after data has been acquired.

Test Configuration: Set Log_DeviceObjectProperty to an external property of type (REAL, UNSIGNED, INTEGER, BOOLEAN, BIT STRING, ENUMERATED). Make the IUT collect samples. Then use one of the qualifying Clause 9.21.1 tests to verify the operation of the ReadRange-Request and ReadRange-Ack. Qualifying tests are: 9.21.1.1, 9.21.1.2, 9.21.1.3, 9.21.1.4, 9.21.1.4.1, 9.21.1.7 and 9.21.1.8.

[Add new **Clause 9.21.1.X4**, p. 266]

9.21.1.X4 Status/Failure logging

Purpose: To verify that a failure is logged when an error is encountered in an attempt to read a data value from the monitored object. If the error is conveyed by an error response from a remote device, verify that the Error Class and Error Code in the response is logged.

Test Concept: Make the monitored object fail and respond with an error by setting the Log_DeviceObjectProperty to an invalid device or object. Wait until the IUT attempts to read a sample for the Log_Buffer. Then check the Log_Buffer to verify that there is a failure entry that consists of the ErrorClass and ErrorCode of the error.

Test Steps:

1. WRITE (Invalid object into the Log_DeviceObjectProperty of the log object)
2. WAIT (until IUT attempts to read a sample for the Log_Buffer)
3. VERIFY (Log_Buffer contains a failure entry of unknown object)

135.1-2009*i*-15 Add Event_Type Test.

Rationale

Add a test to verify that the Event_Type property tracks changes to the Event_Parameter property of the Event Enrollment object.

[Change **Clause 7.3.2.11**, p 64]

7.3.2.11 Event Enrollment Object Test

In addition to tests in this section, additional tests ~~Tests~~ to verify the functionality of the Event Enrollment object are covered by the tests for initiating event notifications in Clauses 8.4 and 8.5. If the IUT supports monitoring objects outside of the IUT one of the tests in Clauses 8.4 and in 8.5 shall be used to demonstrate this capability. This will require providing an additional BACnet device configured appropriately.

[Add new **Clause 7.3.2.11.X**, p 64]

7.3.2.11.X Event_Type Test

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

BACnet Reference Clauses: 12.12.5.

Purpose: This test case verifies that Event_Type property of an Event Enrollment object properly tracks changes to the Event_Parameters property.

Test Concept: Write to the Event_Parameters property and verify that the Event_Type tracks.

Configuration Requirements: The IUT shall be configured with an Event Enrollment object, E1, having a writable Event_Parameters property that will accept a value with a different algorithm choice. The tester shall choose a valid Event_Parameters value, EP1, with an event type value, ET1, that the IUT will accept where the ET1 differs from the Event_Type already existing in E1.

Test Steps:

1. VERIFY Event_Type <> ET1
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = E1,
   'Property Identifier' = Event_Parameters,
   'Property Value' = EP1
3. RECEIVE BACnet-SimpleACK-PDU
4. VERIFY Event_Type = ET1

135.1-2009*i*-16 Revise DeviceCommunicationControl Test.

Rationale

The tests for DeviceCommunicationControl require changes due to changes in Addendum 135-2008*m*-8.

Addendum 135.1-2009*i*-16

[Change **Clause 9.24.2.2**, p 283]

9.24.2.2 Missing Password

Purpose: To verify the correct execution of DeviceCommunicationControl service procedure when a password is required but not provided. If the IUT does not provide password protection*, then* this test case shall be omitted.

Test Steps:

1. TRANSMIT DeviceCommunicationControl-Request,
   'Enable/Disable' = DISABLE,
2. *IF (Protocol_Revision >= 7) THEN*
   RECEIVE BACnet-Error-PDU,
   Error Class = SECURITY,
   Error Code = PASSWORD_FAILURE
   ELSE
   ~~(~~RECEIVE BACnet-Error-PDU,
   Error Class =SECURITY,
   Error Code =PASSWORD_FAILURE~~)~~ |
   (BACnet-Error-PDU,
   Error Class =SERVICES,
   Error Code =MISSING_REQUIRED_PARAMETER)
3. VERIFY (Device, X), *System_Status = (any valid value)*
   ~~(any required non-array property) = (the value for this property specified in the EPICS)~~

[Add new **Clause 9.24.2.X3**, p 284]

9.24.2.X3 Restore by ReinitializeDevice with Invalid 'Reinitialized State of Device'

Purpose: To verify that the communications are not restored when a ReinitializeDevice request is received containing one of the backup or restore related values for service parameter 'Reinitialized State of Device'.

Test Concept: Disable the IUT's communications for a period time, T, which shall be longer than it will take to complete the test. Verify that, while communications are disabled, the IUT correctly responds with a Result(-) when it receives a ReinitializeDevice request containing a backup or restore related values.

Test Steps:

1. TRANSMIT DeviceCommunicationControl-Request,
   'Enable/Disable' = DISABLE
   'Password' = (any appropriate password),
   'Time Duration' = (a value T >= 1, in minutes) | (no value)
2. RECEIVE BACnet-Simple-ACK-PDU
3. WAIT Internal Processing Fail Time
4. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP | ENDBACKUP | STARTRESTORE
   ENDRESTORE | ABORTRESTORE,
   'Password' = (any appropriate password)

5. IF (Protocol_Revision is present and Protocol_Revision >= 7) THEN
   RECEIVE BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = COMMUNICATION_DISABLED
   ELSE
   CHECK(that the IUT responded with BACnet-Error-PDU with an Error Class of SERVICES and an Error Code of COMMUNICATION_DISABLE, or the IUT did not respond at all)
6. TRANSMIT DeviceCommunicationControl-Request,
   'Enable/Disable' = ENABLE
   'Password' = (any appropriate password),
7. RECEIVE BACnet-Simple-ACK-PDU

[Change **Clause 9.27.1.1**, p 285]
[The negative test is moved to 9.27.2.X3]

9.27.1.1 COLDSTART with no Password
Purpose: To verify the correct execution of the ReinitializeDevice request service procedure when a COLDSTART is attempted and no password is ~~provided~~*required*.

Test Steps:

1. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = COLDSTART
2. ~~(~~RECEIVE BACnet-Simple-ACK-PDU
3. CHECK (Did the IUT perform a COLDSTART reboot?) ~~)~~ ~~|~~
~~(RECEIVE BACnet-Error-PDU,~~
~~Error Class = SECURITY,~~
~~Error Code = PASSWORD_FAILURE) |~~
~~(RECEIVE BACnet-Error-PDU,~~
~~Error Class = SERIVCES,~~
~~Error Code = MISSING_REQUIRED_PARAMETER)~~

Notes to Tester: ~~Two cases are possible. If the IUT requires the use of a password one of the specified errors shall be returned. If the IUT does not require the use of a password a simple acknowledgment shall be returned and the IUT shall reinitialize in the manner prescribed by the manufacturer.~~ External indications that the IUT has reinitialized, such as LEDs or startup message ~~traffic shall~~ *traffic, shall* be used to confirm reinitialization whenever possible.

[Change **Clause 9.27.1.3**, p 285]
[The negative test is moved to 9.27.2.X4]

9.27.1.3 WARMSTART with no Password
Purpose: To verify the correct execution of the ReinitializeDevice request service procedure when a warmstart is attempted and no password is ~~provided~~*required*.

Test Steps:

1. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = WARMSTART
2. ~~(~~RECEIVE BACnet-Simple-ACK-PDU
3. CHECK (Did the IUT perform a WARMSTART reboot?) ~~|~~
~~(RECEIVE BACnet-Error-PDU,~~
~~Error Class = SECURITY,~~
~~Error Code = PASSWORD_FAILURE) |~~
~~(RECEIVE BACnet-Error-PDU,~~
~~Error Class = SERIVCES,~~
~~Error Code = MISSING_REQUIRED_PARAMETER)~~

Notes to Tester: ~~Two cases are possible. If the IUT requires the use of a password one of the specified errors shall be returned. If the IUT does not require the use of a password a simple acknowledgment shall be returned and the IUT shall reinitialize in the manner prescribed by the manufacturer.~~ External indications that the IUT has reinitialized, such as LEDs or startup message ~~traffic shall~~ *traffic, shall* be used to confirm reinitialization whenever possible.

[Add new **Clause 9.27.2.X3**, p 285]

9.27.2.X3 COLDSTART with Missing Password

Purpose: To verify that the correct BACnet Error PDU is returned when a COLDSTART is attempted and a password is required but no password is provided.

Test Steps:

1. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = COLDSTART,
2. IF (Protocol_Revision >= 7) THEN
   RECEIVE BACnet-Error-PDU,
   Error Class = SECURITY,
   Error Code = PASSWORD_FAILURE
   ELSE
   (RECEIVE BACnet-Error-PDU,
   Error Class = SECURITY,
   Error Code = PASSWORD_FAILURE) |
   (BACnet-Error-PDU,
   Error Class = SERIVCES,
   Error Code = MISSING_REQUIRED_PARAMETER)
3. CHECK (The IUT did NOT perform a COLDSTART reboot)

Notes to Tester: External indications that the IUT has reinitialized, such as LEDs or startup message traffic, shall be used to confirm reinitialization whenever possible.

[Add new **Clause 9.27.2.X4**, p 285]

9.27.2.X4 WARMSTART with Missing Password

Purpose: To verify that the correct BACnet Error PDU is returned when a WARMSTART is attempted and a password is required but no password is provided.

Test Steps:

1. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = WARMSTART,
2. IF (Protocol_Revision >= 7) THEN
   RECEIVE BACnet-Error-PDU,
   Error Class = SECURITY,
   Error Code = PASSWORD_FAILURE
   ELSE
   RECEIVE BACnet-Error-PDU,
   Error Class = SECURITY,
   Error Code = PASSWORD_FAILURE |
   (BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = MISSING_REQUIRED_PARAMETER)
3. CHECK (The IUT did NOT perform a WARMSTART reboot)

Notes to Tester: External indications that the IUT has reinitialized, such as LEDs or startup message traffic, shall be used to confirm reinitialization whenever possible.

135.1-2009*i*-17 Add Alarm Re-acknowledgment Tests.

Rationale

Add in tests for the alarm re-acknowledgment functionality added in Addendum 135-2004*b*-8.

[Add new **Clauses 9.1.1.X...**, p.194]

9.1.1.X1 Successful Alarm Re-Acknowledgment of Confirmed Event Notifications

Purpose: To verify the successful re-acknowledgment of an event signaled by a ConfirmedEventNotification, including notification of other workstations and updating of the Acked_Transitions status.

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

Configuration Requirements: The IUT shall be configured with at least one event-initiating object that generates offnormal transitions and sends confirmed notifications. The Acked_Transitions property shall have the value B'111', indicating that all transitions have been acknowledged. The TD and at least one other BACnet device shall be recipients of the event notification.

Test Steps:

1. MAKE (a change that triggers the detection of an event in the IUT)
2. WAIT (Time_Delay)
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-initiating object),
   'Time Stamp' = (the current time or sequence number),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event),
   'Event Type' = (any valid event type),
   'Notify Type' = (the notify type configured for this event),
   'AckRequired' = TRUE,
   'From State' = NORMAL,
   'To State' = (any appropriate offnormal event state),
   'Event Values' = (the values appropriate to the event type)
4. TRANSMIT BACnet-SimpleACK-PDU
5. RECEIVE
   DESTINATION = (at least one device other than the TD),
   SOURCE = IUT,
   ConfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-initiating object),
   'Time Stamp' = (the timestamp or sequence number received in step 3),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event),
   'Event Type' = (any valid event type),
   'Notify Type' = (the notify type configured for this event),

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

6. TRANSMIT BACnet-SimpleACK-PDU
7. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = B'011'
8. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (the time stamp conveyed in the notification),
   'Acknowledgment Source' = (a character string)
   'Time of Acknowledgment' = (any of the forms specified for this parameter)
9. RECEIVE BACnet-Simple-ACK-PDU
10. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (the current time or sequence number),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (the event type included in step 3),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (the 'To State' used in step 3)
11. TRANSMIT BACnet-SimpleACK-PDU
12. RECEIVE
    DESTINATION = (at least one device other than the TD),
    SOURCE = IUT,
    ConfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (the timestamp or sequence number received in step 10),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (the event type included in step 3),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (the 'To State' used in step 3)
13. TRANSMIT BACnet-SimpleACK-PDU
14. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = B'111'
15. TRANSMIT AcknowledgeAlarm-Request,
    'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
    'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
    'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
    'Time Stamp' = (the time stamp conveyed in the notification),
    'Acknowledgment Source' = (a character string)
    'Time of Acknowledgment' = (any of the forms specified for this parameter)
16. RECEIVE BACnet-SimpleACK-PDU
17. BEFORE **Notification Fail Time**

RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-initiating object),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step 3),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step 3)

18. TRANSMIT BACnet-SimpleACK-PDU
19. RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-initiating object),
'Time Stamp' = (the timestamp or sequence number received in step 17),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step 3),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step 3)
20. TRANSMIT BACnet-SimpleACK-PDU
21. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = B'111'

Notes to Tester: The destination address used for the acknowledgment notification in steps 12 and 19 shall be the same address used in step 4. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant.

9.1.1.X2 Successful Alarm Re-Acknowledgment of Unconfirmed Event Notifications

Purpose: To verify the successful re-acknowledgment of an event signaled by an UnconfirmedEventNotification, including notification of other workstations and updating of the Acked_Transitions status.

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

Configuration Requirements: The IUT shall be configured with at least one event-initiating object that generates offnormal transitions and sends unconfirmed notifications. The Acked_Transitions property shall have the value B'111', indicating that all transitions have been acknowledged. The TD and at least one other BACnet device shall be recipients of the event notification.

Test Steps:

1. MAKE (a change that triggers the detection of an offnormal event in the IUT)
2. WAIT (Time_Delay)
3. BEFORE **Notification Fail Time**
RECEIVE UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-initiating object),

'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (any valid event type),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = (any appropriate offnormal event state),
'Event Values' = (the values appropriate to the event type)

4. RECEIVE
   DESTINATION = (at least one device other than the TD),
   SOURCE = IUT,
   UnconfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-initiating object),
   'Time Stamp' = (the timestamp or sequence number received in step 3),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event),
   'Event Type' = (any valid event type),
   'Notify Type' = (the notify type configured for this event),
   'AckRequired' = TRUE,
   'From State' = NORMAL,
   'To State' = (any appropriate offnormal event state),
   'Event Values' = (the values appropriate to the event type)
5. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = B'011'
6. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (the time stamp conveyed in the notification),
   'Acknowledgment Source' = (a character string)
   'Time of Acknowledgment' = (any of the forms specified for this parameter)
7. RECEIVE BACnet-SimpleACK-PDU
8. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-initiating object),
   'Time Stamp' = (the current time or sequence number),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event),
   'Event Type' = (the event type included in step 3),
   'Notify Type' = ACK_NOTIFICATION,
   'To State' = (the 'To State' used in step 3)
9. RECEIVE
   DESTINATION = (at least one device other than the TD),
   SOURCE = IUT,
   UnconfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-initiating object),
   'Time Stamp' = (the timestamp or sequence number received in step 8),

'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step 3),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step 3)

10. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = B'111'
11. TRANSMIT AcknowledgeAlarm-Request,
    'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
    'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
    'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
    'Time Stamp' = (the time stamp conveyed in the notification),
    'Acknowledgment Source' = (a character string)
    'Time of Acknowledgment' = (any of the forms specified for this parameter)
12. RECEIVE BACnet-SimpleACK-PDU
13. BEFORE **Notification Fail Time**
    RECEIVE UnconfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (the current time or sequence number),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (the event type included in step 3),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (the 'To State' used in step 3)
14. RECEIVE
    DESTINATION = (at least one device other than the TD),
    SOURCE = IUT,
    UnconfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (the timestamp or sequence number received in step 13),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (the event type included in step 3),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (the 'To State' used in step 3)
15. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = B'111'

Notes to Tester: The destination address used for the acknowledgment notification in steps 9 and 14 shall be the same address used in step 4. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant.

135.1-2009*i*-18. Modify I-Am Tests.

Rationale

Modify the I-Am tests to allow the I-Am to be unicast. The wait for the I-Am is also modified to allow for longer periods for the generation of the I-Am as might be the case if the device is acting as a gateway and staggers I-Am responses.

[Change **Clause 4.5.9**, p. 8]

4.5.9 Timers

This section defines timer values that are used to determine when a test has failed because an appropriate response has not been observed by the TD. A Real value in seconds must be provided for each timer. See *Clause* 6.3.

Fail Times: ↵
{↵
Notification Fail Time: ❑↵
Internal Processing Fail Time: ❑↵
Minimum ON/OFF Time: ❑↵
Schedule Evaluation Fail Time: ❑↵
External Command Fail Time: ❑↵
Program Object State Change Fail Time: ❑↵
Acknowledgement Fail Time: ❑↵
Slave Proxy Confirm Interval: ❑↵
Unconfirmed Response Fail Time: ❑↵
}↵

[Add new **Clause 6.3.X**, p. 30]

6.3.X Unconfirmed Response Fail Time

The **Unconfirmed Response Fail Time** is the elapsed time, in seconds, between when the IUT receives an unconfirmed request and when a test is considered to have failed because the expected unconfirmed request from the IUT has not been received. For example, this is the maximum amount of time that the IUT would take to initiate an I-Am request in response to a Who-Is request.

[Change **Clause 9.33.1.1**, p. 296]

9.33.1.1 Local Broadcast, General Inquiry

Purpose: To verify that the IUT can correctly respond to a local broadcast Who-Is service request that does not restrict device ranges.

Test Steps:

1. TRANSMIT
   DESTINATION = LOCAL BROADCAST,
   Who-Is-Request
2. ~~WAIT~~ **~~Internal Processing Fail Time~~**
3. *BEFORE* ***Unconfirmed Response Fail Time***
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | LOCAL BROADCAST | *TD*
   I-Am-Request,

'I Am Device Identifier' = (the IUT's Device object),
'Max APDU Length Accepted' = (the value specified in the EPICS),
'Segmentation Supported' = (the value specified in the EPICS),
'Vendor Identifier' = (the identifier registered for this vendor)

[Change **Clause 9.33.1.2**, p. 296]

9.33.1.2 Global Broadcast, General Inquiry

Purpose: To verify that the IUT can correctly respond to a global broadcast Who-Is request that does not restrict device ranges.

Test Steps:

1. TRANSMIT
   DESTINATION = GLOBAL BROADCAST,
   Who-Is-Request
2. ~~WAIT~~ **~~Internal Processing Fail Time~~**
3. *BEFORE* ***Unconfirmed Response Fail Time***
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | LOCAL BROADCAST | *TD*
   I-Am-Request,
   'I Am Device Identifier' = (the IUT's Device object),
   'Max APDU Length Accepted' = (the value specified in the EPICS),
   'Segmentation Supported' = (the value specified in the EPICS),
   'Vendor Identifier' = (the identifier registered for this vendor)

[Change **Clause 9.33.1.4**, p. 297]

9.33.1.4 Local Broadcast, Specific Device Inquiry with IUT Device Instance Equal to Low Limit of Device Range

Purpose: To verify that the IUT correctly recognizes the low limit of the specified device range.

Test Steps:

1. TRANSMIT
   DESTINATION = LOCAL BROADCAST,
   Who-Is-Request,
   'Device Instance Range Low Limit' = (The Device object instance number of the IUT),
   'Device Instance Range High Limit' = (any value H: H > the Device object instance number of the IUT)
2. ~~WAIT~~ **~~Internal Processing Fail Time~~**
3. *BEFORE* ***Unconfirmed Response Fail Time***
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | LOCAL BROADCAST | *TD*
   I-Am-Request,
   'I Am Device Identifier' = (the IUT's Device object),
   'Max APDU Length Accepted' = (the value specified in the EPICS),
   'Segmentation Supported' = (the value specified in the EPICS),
   'Vendor Identifier' = (the identifier registered for this vendor)

[Change **Clause 9.33.1.5**, p. 297]

9.33.1.5 Local Broadcast, Specific Device Inquiry with IUT Device Instance Equal to High Limit of Device Range

Purpose: To verify that the IUT correctly recognizes the high limit of the specified device range.

Test Steps:

1. TRANSMIT
   DESTINATION = LOCAL BROADCAST,
   Who-Is-Request,
   'Device Instance Range Low Limit' = (any value L: 0 ≤ L < the Device object instance number of the IUT),
   'Device Instance Range High Limit' = (the Device object instance number of the IUT)
2. ~~WAIT **Internal Processing Fail Time**~~
3. ~~*BEFORE* ***Unconfirmed Response Fail Time***~~
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | LOCAL BROADCAST | *TD*
   I-Am-Request,
   'I Am Device Identifier' = (the IUT's Device object),
   'Max APDU Length Accepted' = (the value specified in the EPICS),
   'Segmentation Supported' = (the value specified in the EPICS),
   'Vendor Identifier' = (the identifier registered for this vendor)

[Change **Clause 9.33.1.6**, p. 297]

9.33.1.6 Local Broadcast, Specific Device Inquiry with IUT Inside of the Device Range

Purpose: To verify that the IUT responds to Who-Is requests when it is included within the specified device range.

Test Steps:

1. TRANSMIT
   DESTINATION = LOCAL BROADCAST,
   Who-Is-Request,
   'Device Instance Range Low Limit' = (any value L: 0 ≤ L < the Device object instance number of the IUT),
   'Device Instance Range High Limit' = (any value H: H > the Device object instance number of the IUT)
2. ~~WAIT **Internal Processing Fail Time**~~
3. ~~*BEFORE* ***Unconfirmed Response Fail Time***~~
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | LOCAL BROADCAST | *TD*
   I-Am-Request,
   'I Am Device Identifier' = (the IUT's Device object),
   'Max APDU Length Accepted' = (the value specified in the EPICS),
   'Segmentation Supported' = (the value specified in the EPICS),
   'Vendor Identifier' = (the identifier registered for this vendor)

[Change **Clause 9.33.2.1**, p. 274]

9.33.2.1 General Inquiry, Global Broadcast from a Remote Network

Purpose: To verify the ability of the IUT to recognize the origin of a globally broadcast Who-Is service request and respond such that the device originating the request receives the response.

Test Steps:

1. TRANSMIT
   DESTINATION = GLOBAL BROADCAST,
   SNET = (any remote network number),
   SADR = (any MAC address valid for the specified network),
   Who-Is-Request
2. ~~WAIT **Internal Processing Fail Time**~~
3. ~~*BEFORE* ***Unconfirmed Response Fail Time***~~

RECEIVE
DESTINATION = GLOBAL BROADCAST |
REMOTE BROADCAST (to the network specified by SNET in step 1) |
TD,
I-Am-Request,
'I Am Device Identifier' = (the IUT's Device object),
'Max APDU Length Accepted' = (the value specified in the EPICS),
'Segmentation Supported' = (the value specified in the EPICS),
'Vendor Identifier' = (the identifier registered for this vendor)

[Change **Clause 9.33.2.2**, p. 274]

9.33.2.2 General Inquiry, Remote Broadcast

Purpose: To verify the ability of the IUT to recognize the origin of a remotely broadcast Who-Is service request and respond such that the device originating the request receives the response.

Test Steps:

1. TRANSMIT
   DESTINATION = LOCAL BROADCAST,
   SNET = (any remote network number),
   SADR = (any MAC address valid for the specified network),
   Who-Is-Request
2. ~~WAIT **Internal Processing Fail Time**~~
3. *BEFORE **Unconfirmed Response Fail Time***
   RECEIVE
   DESTINATION = GLOBAL BROADCAST |
   REMOTE BROADCAST (to the network specified by SNET in step 1)
   | *TD*,
   I-Am-Request,
   'I Am Device Identifier' = (the IUT's Device object),
   'Max APDU Length Accepted' = (the value specified in the EPICS),
   'Segmentation Supported' = (the value specified in the EPICS),
   'Vendor Identifier' = (the identifier registered for this vendor)

[Change **Clause 9.33.2.3**, p. 299]

9.33.2.3 General Inquiry, Directed to a Remote Device

Reason for Change: The original test incorrectly allowed a Local Broadcast response instead of the Remote Broadcast response.

Purpose: To verify that the IUT responds with ~~a broadcast~~ *an* I-Am service ~~request even if the Who-Is service request was not transmitted with a broadcast address.~~ *that is of the form global broadcast, remote broadcast or unicast.*

Test Steps:

1. TRANSMIT
   DESTINATION = IUT,
   SNET = (any remote network number),
   SADR = (any MAC address valid for the specified network),
   Who-Is-Request
2. ~~WAIT **Internal Processing Fail Time**~~
3. *BEFORE **Unconfirmed Response Fail Time***
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | ~~LOCAL BROADCAST~~

REMOTE BROADCAST (to the network specified by SNET in step 1) | TD,
I-Am-Request,
'I Am Device Identifier' = (the IUT's Device object),
'Max APDU Length Accepted' = (the value specified in the EPICS),
'Segmentation Supported' = (the value specified in the EPICS),
'Vendor Identifier' = (the identifier registered for this vendor)

[Change **Clause 13.5.3**, p. 455]
[Step renumbering is not shown in change marking to improve clarity.]

13.5.3 Proxy Test

Purpose: This test verifies that an IUT correctly proxies for slaves that are listed in its Slave_Address_Binding property.

Test Concept: Configure the IUT so that it will proxy for a slave device and wait for the IUT to find and confirm the slave. Issue Who-Is requests in all forms to ensure that the IUT correctly proxies the I-Am responses for the slave device. The test should be repeated with the TD connected to the MS/TP segment, and with the TD connected to a different BACnet network.

Configuration Requirements: The MS/TP network shall contain a slave device at address A1 with a device identifier of D1. The IUT shall be configured to perform slave proxying. The test starts after the IUT has successfully found and confirmed the slave device. This test shall be repeated once with the TD connected to the MS/TP network and once with the TD connected to a different BACnet network.

BACnet Reference Clauses: 12.11.39, 12.11.40, 12.11.41, 12.11.42, and 16.10.2.

Test Steps:

1. TRANSMIT
   DESTINATION = GLOBAL BROADCAST,
   Who-Is
~~2. WAIT **Internal Processing Fail Time**~~
2. *BEFORE **Unconfirmed Response Fail Time***
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | LOCAL BROADCAST | *TD*
   SOURCE = A1
   I-Am-Request,
   'I Am Device Identifier' = (the slave's Device object's Object_Identifer),
   'Max APDU Length Accepted' = (the slave's value for this property),
   'Segmentation Supported' = FALSE,
   'Vendor Identifier' = (the slave's value for this property)
3. TRANSMIT DESTINATION = A1, Who-Is
~~5. WAIT **Internal Processing Fail Time**~~
4. *BEFORE **Unconfirmed Response Fail Time***
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | LOCAL BROADCAST | *TD*
   SOURCE = A1
   I-Am-Request,
   'I Am Device Identifier' = (the slave's Device object's Object_Identifer),
   'Max APDU Length Accepted' = (the slave's value for this property),
   'Segmentation Supported' = FALSE,
   'Vendor Identifier' = (the slave's value for this property)
5. TRANSMIT DESTINATION = GLOBAL BROADCAST, Who-Is
~~8. WAIT **Internal Processing Fail Time**~~
6. *BEFORE **Unconfirmed Response Fail Time***
   RECEIVE

DESTINATION =	GLOBAL BROADCAST | LOCAL BROADCAST | *TD*
SOURCE =	A1
I-Am-Request,	
'I Am Device Identifier' =	(the slave's Device object's Object_Identifer),
'Max APDU Length Accepted' =	(the slave's value for this property),
'Segmentation Supported' =	FALSE,
'Vendor Identifier' =	(the slave's value for this property)

135.1-2009*i*-19. Add A-side Trend Tests.

Rationale

The section adds tests for the A-side logging BIBBs.

[Add new **Clauses 8.21.X...**, p. 168]

8.21.X1 Reading a Range of Items Using Any Valid Range in Response to ConfirmedEventNotifications of the BUFFER_READY Event Type

Purpose: To verify that the IUT can initiate a series of ReadRange service requests that access a contiguous set of log records by responding to ConfirmedEventNotification messages with an 'Event Type' of BUFFER_READY.

Test Concept: The TD contains a log object that has two logical sets of log records, S1and S2. The log continues to accumulate log records until logical record sets S3 and S4 are added to the buffer. The TD sends BUFFER_READY event notifications to the IUT about the addition of S2 and S3. The IUT accesses the records in S2 and S3 by sending a series of ReadRange requests, using any valid range. The test then verifies that the IUT has accessed a contiguous set of records in S2, S3, and possibly some portion of S4. The size of sets S1, S2, S3, and S4 shall each be small enough that none of them causes the reference server to send an event notification.

The test verifies that the IUT does not simply read the entire buffer each time, nor miss samples in the Log_Buffer.

Configuration Requirements: The TD contains a log object, L1, that has a Buffer_Size property of sufficient number that it can contain the set of records S1, S2, S3, and S4, where the size of each set is large enough that the TD cannot return the entire set with a single ReadRange response (this behavior can be achieved by disabling the TD's ability to transmit segmented messages or by limiting the number of segments the TD can transmit). The TD is configured such that it adds entries to its Log_Buffer for the duration of the test and will send a ConfirmedEventNotification of type BUFFER_READY to the IUT after the addition of S2 and S3.

Test Steps:

1. MAKE (L1 contain record sets S1 + S2)
2. TRANSMIT ConfirmedEventNotification-Request,
   'Subscriber Process Identifier' = (any valid value selected for the IUT),
   'Initiating Device Identifier' = TD,
   'Event Object Identifier' = (the object detecting the event),
   'Time Stamp' = (any valid value),
   'Notification Class' = (any valid notification class),
   'Priority' = (any valid value),
   'Event Type' = BUFFER_READY,
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = NORMAL,
   'Event Values' = (L1, Log_Buffer),
   (any Unsigned32 value that indicates the end of S1),
   (any Unsigned32 value that indicates the addition of S2)
3. RECEIVE Simple-Ack-PDU
4. BEFORE (a time interval specified by the vendor)
   WHILE (not all records from S2 have been returned by TD (see notes to tester))
   RECEIVE ReadRange-Request,
   'Object Identifier' = (L1),
   'Property Identifier' = (Log_Buffer),
   'Range' = (any valid range)

TRANSMIT ReadRange-Ack,
'Object Identifier' = (L1),
'Property Identifier' = (Log_Buffer),
'ItemData' = (a set of records appropriate for this response)
5. MAKE (L1 contain records S1 + S2 + S3)
6. TRANSMIT ConfirmedEventNotification-Request,
'Subscriber Process Identifier' = (any valid value selected for the IUT),
'Initiating Device Identifier' = TD,
'Event Object Identifier' = (the object detecting the event),
'Time Stamp' = (any valid value),
'Notification Class' = (any valid notification class),
'Priority' = (any valid value),
'Event Type' = BUFFER_READY,
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = NORMAL,
'Event Values' = (L1, Log_Buffer),
(any Unsigned32 value that indicates the end of S2),
(any Unsigned32 value that indicates the addition of S3)
7. RECEIVE Simple-Ack-PDU
8. MAKE (L1 contain records S1 + S2 + S3 + S4 (see note to tester))
9. BEFORE (a time interval specified by the vendor)
WHILE (not all records from S3 have been returned by D1)
RECEIVE ReadRange-Request,
'Object Identifier' = (L1),
'Property Identifier' = (Log_Buffer),
'Range' = (any valid range)
TRANSMIT ReadRange-Ack,
'Object Identifier' = (L1),
'Property Identifier' = (Log_Buffer),
'ItemData' = (a set of records appropriate for this response)
10. CHECK (the records from all steps form a contiguous set, including all records in S2, S3, and possibly some portion of S4)

Note to Tester: Before step 4, the IUT may have pre-read some of S2. In such cases, the IUT shall not be failed for not reading those samples during step 4.
Note to Tester: The addition of S4 in step 8 can occur anytime after step 6 but before step 10.

8.21.X2 Reading a Range of Items Using Any Valid Range in Response to UnconfirmedEventNotifications of the BUFFER_READY Event Type

Purpose: To verify that the IUT can initiate a series of ReadRange service requests that access a contiguous set of log records by responding to UnconfirmedEventNotification messages with an 'Event Type' of BUFFER_READY.

The test concept, configuration requirements, and test steps are identical to those in 8.21.X1, except UnconfirmedEventNotification requests are issued instead of ConfirmedEventNotification requests, and the IUT does not acknowledge receiving the notifications.

8.21.X3 Reading a Range of Items Using Any Valid Range

Purpose: To verify that the IUT can initiate one or more ReadRange requests that access a tester-specified portion of log records, using any valid range.

Test Concept: The TD contains a Trend Log object that has a logical set of log records, S1. The tester selects a portion of S1 to be returned, and causes the IUT to request those records, using any valid range. The test then verifies that the IUT can display the records in a manner consistent with those that the TD returns.

Configuration Requirements: The TD contains a log object, L1, which has a set of records, S1. The IUT is configured to display the returned set of log records.

Test Steps:

1. MAKE (L1 contain the set of records S1)
2. MAKE (the IUT request a range of samples from L1)
3. WHILE (not all records from the tester-selected range have been requested)
    RECEIVE ReadRange-Request,
        'Object Identifier' = (L1),
        'Property Identifier' = (Log_Buffer),
        'Range' = (any valid range)
    TRANSMIT ReadRange-Ack,
        'Object Identifier' = (L1),
        'Property Identifier' = (Log_Buffer),
        'ItemData' = (a set of records appropriate for this response)
4. CHECK (the records returned in step 3 include the tester-selected range)
5. MAKE (the IUT display the tester-selected range)
6. CHECK (the records in step 5 are consistent with the records returned in step 3)

Note to Tester: The IUT is not required to display records containing log-status values.

8.21.X4 Presents Log Records Containing a Specific Datatype

Purpose: To verify that the IUT can initiate one or more ReadRange requests that access and present a tester-specified portion of log records having a specific datatype, using any valid range. It is a generic test used to test data presentation requirements.

Test Concept: Run test in Clause 8.21.X3 and verify that the data presentation meets the criteria specified by the BIBB being tested.

Note to Tester: The values presented by the IUT may differ from the values transmitted on the wire due to rounding, truncation, formatting, language, conversion, etc.

Note to Tester: The IUT is not required to display records containing log-status values.

135.1-2009*i*-20. Make the EPICS Definition Generic.

Rationale

Rework the description of the EPICS to be generic and not require updates when object types, services, or BIBBs are added to the standard.

[Change **Clause 4.5.2.1**, p. 5]

4.5.2.1 BIBBs Supported

This section indicates which BIBBs are supported. The syntax is shown below. Each BIBB shall be listed, one per line between the curly braces. An empty list indicates that no BIBBs are supported.

BIBBs Supported: ↵
{↵
❑↵
}↵

~~The BIBBs may be any of:~~

~~DS-RP-A~~	~~DS-RP-B~~	
~~DS-RPM-A~~	~~DS-RPM-B~~	
~~DS-RPC-A~~	~~DS-RPC-B~~	
~~DS-WP-A~~	~~DS-WP-B~~	
~~DS-WPM-A~~	~~DS-WPM-B~~	
~~DS-COV-A~~	~~DS-COV-B~~	
~~DS-COVP-A~~	~~DS-COVP-B~~	
~~DS-COVU-A~~	~~DS-COVU-B~~	
~~AE-N-A~~	~~AE-N-I-B~~	~~AE-N-E-B~~
~~AE-ACK-A~~	~~AE-ACK-B~~	
~~AE-ASUM-A~~	~~AE-ASUM-B~~	
~~AE-ESUM-A~~	~~AE-ESUM-B~~	
~~AE-INFO-A~~	~~AE-INFO-B~~	
~~AE-LS-A~~	~~AE-LS-B~~	
~~SCHED-A~~	~~SCHED-I-B~~	~~SCHED-E-B~~
~~T-VMT-A~~	~~T-VMT-I-B~~	~~T-VMT-E-B~~
~~T-ATR-A~~	~~T-ATR-B~~	
~~DM-DDB-A~~	~~DM-DDB-B~~	
~~DM-DOB-A~~	~~DM-DOB-B~~	
~~DM-DCC-A~~	~~DM-DCC-B~~	
~~DM-PT-A~~	~~DM-PT-B~~	
~~DM-TM-A~~	~~DM-TM-B~~	
~~DM-TS-A~~	~~DM-TS-B~~	
~~DM-UTC-A~~	~~DM-UTC-B~~	
~~DM-RD-A~~	~~DM-RD-B~~	
~~DM-BR-A~~	~~DM-BR-B~~	
~~DM-R-A~~	~~DM-R-B~~	
~~DM-LM-A~~	~~DM-LM-B~~	
~~DM-OCD-A~~	~~DM-OCD-B~~	
~~DM-VT-A~~	~~DM-VT-B~~	
~~NM-CE-A~~	~~NM-CE-B~~	
~~NM-RC-A~~	~~NM-RC-B~~	

The BIBBs may be any of those BIBBs described in Annex K. The format in the EPICS shall be to use the short acronym described in the title for each BIBB. For example: A device which supports 'Data Sharing - ReadProperty - B' should include 'DS-RP-B' in this section.

For example:

BIBBS Supported: : ↵
{↵
DS-RP-B↵
DS-WP-B↵
DS-RPM-B↵
DM-DOB-B↵
DM-DDB-B↵
DM-DDB-A↵
}↵

[Change **Clause 4.5.3**, p. 6]

4.5.3 Application Services Supported

This section indicates which standard application services are supported. The syntax is shown below. Each supported service shall be listed between curly braces one service per line, followed by the words "Initiate" or "Execute" to indicate whether the service can be initiated, executed, or both.

BACnet Standard Application Services Supported: ↵
{↵
❑ Initiate↵
❑ Execute↵
❑ Initiate Execute↵
}

The standard services may be any of *the services listed in the Clause 21 production BACnetServicesSupported.*~~÷~~

~~AcknowledgeAlarm RemoveListElement ConfirmedTextMessage~~
~~ConfirmedCOVNotification CreateObject UnconfirmedTextMessage~~
~~UnconfirmedCOVNotification DeleteObject TimeSynchronization~~
~~ConfirmedEventNotification ReadProperty UTCTimeSynchronization~~
~~UnconfirmedEventNotification ReadPropertyConditional Who-Has~~
~~GetAlarmSummary ReadPropertyMultiple I-Have~~
~~GetEnrollmentSummary ReadRange Who-Is~~
~~GetEventInformation WriteProperty I-Am~~
~~LifeSafetyOperation WritePropertyMultiple VT-Open~~
~~SubscribeCOV DeviceCommunicationControl VT-Close~~
~~SubscribeCOVProperty ConfirmedPrivateTransfer VT-Data~~
~~AtomicReadFile UnconfirmedPrivateTransfer RequestKey~~
~~AtomicWriteFile ReinitializeDevice Authenticate~~
~~AddListElement~~

The format should match the title of each corresponding services section in the standard minus the text 'Service'. For example, if the device supports the AcknowledgeAlarm service, the text 'AcknowledgeAlarm' should be included in this section of the EPICS.

For example:

BACnet Standard Application Services Supported: ↵
{↵
Who-Is Initiate Execute↵
I-Am Initiate Execute↵
Who-Has Execute↵
I-Have Initiate ↵
ReadProperty Execute↵
ReadPropertyMultiple Execute↵
WriteProperty Execute↵
}

[Change **Clause 4.5.4**, p. 6]

4.5.4 Object Types Supported

This section indicates which standard object types are supported. The syntax is shown below. Each supported object type shall be listed between curly braces one object type per line, optionally followed by the words "Createable", "Deleteable", or both to indicate that dynamic creation or deletion is supported.

Standard Object Types Supported: ↵
{↵
❑↵
❑ Createable↵
❑ Deleteable↵
❑ Createable Deleteable↵
}↵

The standard objects may be any of *the objects listed in the Clause 21 production, BACnetObjectTypesSupported.*~~:~~

~~Access Door~~	~~Binary Output~~	~~Group~~	~~Multi-state Value~~
~~Accumulator~~	~~Binary Value~~	~~Life Safety Point~~	~~Notification Class~~
~~Analog Input~~	~~Calendar~~	~~Life Safety Zone~~	~~Program~~
~~Analog Output~~	~~Command~~	~~Load Control~~	~~Pulse Converter~~
~~Analog Value~~	~~Device~~	~~Loop~~	~~Schedule~~
~~Averaging~~	~~Event Enrollment~~	~~Multi-state Input~~	~~Structured View~~
~~Binary Input~~	~~File~~	~~Multi-state Output~~	~~Trend Log~~

The format should be the title of each corresponding object section in the standard minus the text Object Type. For example, if the device supports the object access door, the text 'Access Door' should be included in this section.

For example:

BACnet Standard Application Services Supported: ↵
{↵
Analog Value Createavle Deleteable↵
Analog Input↵
Device↵
}

135.1-2009*i*-20. Clarify Priority in the GetEnrollmentSummary Priority Filter Test.

Rationale

Rework the description of the EPICS to be generic and not require updates when object types, services, or BIBBs are added to the standard.

[Change **Clause 9.7.2.4**, p 210]

9.7.2.4 Priority Filter
Purpose: To verify that the IUT can execute the GetEnrollmentSummary request when the 'Priority Filter' is used.

Configuration Requirements: If possible, the IUT shall be configured with one or more event-generating objects at each of four different priority levels. *For the purpose of this test, the priority of an event-generating object is the priority of the object's most recent transition.* The priority levels shall be 0, X_{low}, X_{high}, 255, where $10 < X_{low} < 100$ and $100 < X_{high} < 255$. If only a subset of these priorities can be supported at one time*, then* as many of them as possible shall be configured.

Test Steps:

1. TRANSMIT GetEnrollmentSummary-Request,
   'Acknowledgment Filter' = ALL,
   'MinPriority ' = 0,
   'MaxPriority' = 0
2. RECEIVE GetEnrollmentSummary-ACK,
   'List of Enrollment Summaries' = (all configured event-generating objects with a priority in the specified range)
3. TRANSMIT GetEnrollmentSummary-Request,
   'Acknowledgment Filter' = ALL,
   'MinPriority' = 0,
   'MaxPriority' = 255
4. RECEIVE GetEnrollmentSummary-ACK,
   'List of Enrollment Summaries' = (all configured event-generating objects with a priority in the specified range)
5. TRANSMIT GetEnrollmentSummary-Request,
   'Acknowledgment Filter' = ALL,
   'MinPriority' = Xlow,
   'MaxPriority' = Xhigh
6. RECEIVE GetEnrollmentSummary-ACK,
   'List of Enrollment Summaries' = (all configured event-generating objects with a priority in the specified range)

135.1-2009*i*-22. Add Non-documented Property and Read-Only Property Tests.

Rationale

Many of the tests rely on accurate documentation of the IUT in the EPICS. These new tests check for common errors in EPICS.

[Change **Clause 7.1**, p. 31]

7.1 Read Support for Properties in the Test Database

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 read using BACnet ReadProperty and ReadPropertyMultiple services. The test is performed once using ReadProperty and once using ReadPropertyMultiple*, if supported*. When verifying array properties, the whole array 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 {
            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.

[Add new **Clause 7.1.X**, p. 31]

7.1.X Non-documented Property Test

Purpose: To verify that all properties contained in every object are documented in the EPICS.

Test Concept: For each object in the EPICS database, attempt to read each standard property that the EPICS does not document as being part of the object.

Test Steps:

```
1.  REPEAT X = (a tester selected set of objects) DO {
        REPEAT Y = (0 through 511) DO {
            IF (the property Y is not in the EPICS for object X) THEN
                TRANSMIT ReadProperty-Request,
                    'Object Identifier' =     X,
                    'Property Identifier' =  Y
                RECEIVE BACnet-Error-PDU,
                    Error Class =PROPERTY,
                    Error Code =UNKNOWN_PROPERTY
            }
        }
```

Notes to Tester: The objects selected by the tester should include one instance of each supported object type. Where some instances of an object type differ in the set of supported properties, the allowable value ranges for a property, or the writability of a property, then one instance of each variant of that object type should be selected.

[Add new **Clause 7.2.X**, p. 32]

7.2.X Read-only Property Test

Purpose: To verify that properties marked as read-only in the EPICS are in fact read-only.

Test Concept: To each read-only (not writable and not conditionally writable) property in the EPICS, write the value of the property as read from the device and verify that an error is returned. Write another value that is within the acceptable range for the datatype and verify that an error is returned. If the property is a list and the IUT supports AddListElement, attempt to modify the property with AddListElement and verify that an error is returned. If the IUT does not support the WriteProperty service, then this test shall be skipped.

Test Steps:

1. REPEAT X = (a tester selected set of objects) DO {
   REPEAT Y = (all read-only properties in object X) DO {
   IF (the property is not an array) THEN
   READ Z = X
   TRANSMIT WriteProperty-Request,
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Property Value' = Z
   RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = WRITE_ACCESS_DENIED

   TRANSMIT WriteProperty-Request,
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Property Value' = (any value meeting the range requirements of 7.2.1 except Z)
   RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = WRITE_ACCESS_DENIED

   IF (the IUT supports AddListElement and the property is a list) THEN
   TRANSMIT AddListElement-Request,
   'Object Identifier' = X,
   'Property Identifier' = Y,
   List of Elements' = (any element value meeting the range requirements of 7.2.1 excluding those in Z)
   RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = WRITE_ACCESS_DENIED
   ELSE
   READ LEN = X, Array_Index = 0
   IF (LEN > 0) THEN
   READ Z = X, Array Index = 1
   TRANSMIT WriteProperty-Request,
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Property Value' = Z
   'Array Index' = 1

```
            RECEIVE BACnet-Error-PDU,
                Error Class =           PROPERTY,
                Error Code =            WRITE_ACCESS_DENIED

            TRANSMIT WriteProperty-Request,
                'Object Identifier' =   X,
                'Property Identifier' = Y,
                'Property Value' =      (any value meeting the range requirements of 7.2.1 except Z)
                'Array Index' =         1
            RECEIVE BACnet-Error-PDU,
                Error Class =           PROPERTY,
                Error Code =            WRITE_ACCESS_DENIED

            IF (the IUT supports AddListElement and the property is an array of lists) THEN
                TRANSMIT AddListElement-Request,
                    'Object Identifier' =   X,
                    'Property Identifier' = Y,
                    'Array Index' =         1
                    List of Elements' =     (any elements value meeting the range requirements of 7.2.1excluding
                                            those in Z)
                RECEIVE BACnet-Error-PDU,
                    Error Class =           PROPERTY,
                    Error Code =            WRITE_ACCESS_DENIED
            ELSE
                TRANSMIT WriteProperty-Request,
                    'Object Identifier' =   X,
                    'Property Identifier' = Y,
                    'Property Value' =      (any value meeting the range requirements of 7.2.1)
                RECEIVE BACnet-Error-PDU,
                    Error Class =           PROPERTY,
                    Error Code =            WRITE_ACCESS_DENIED
        }
    }
```

Notes to Tester: The objects selected by the tester should include one instance of each supported object type. Where some instances of an object type differ in the set of supported properties, the allowable value ranges for a property, or the writability of a property, then one instance of each variant of that object type should be selected.

Notes to Tester: When modifying a property, it is expected that an error class of PROPERTY with an error code of WRITE_ACCESS_DENIED will be returned, but the IUT may instead return an error_class of PROPERTY with an error_code of VALUE_OUT_OF_RANGE, or an error_class of RESOURCES with an error_code of NO_SPACE_TO_WRITE_PROPERTY.

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

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

HISTORY OF REVISIONS

Summary of Changes to the Standard
…
Addendum *i* to ANSI/ASHRAE 135.1-2009 Approved by the ASHRAE Standards Committee January 29, 2011; by the ASHRAE Board of Directors February 2, 2011; and by the American National Standards Institute February 3, 2011. 1. Improve Schedule Object Restoration Tests. 2. Add Test to Check Range of the Present_Value of Multi-state Objects. 3. Add Test for SubscribeCOV Service Execution Without a Lifetime Parameter. 4. Update Test for Processing of ReadProperty Service Responses. 5. Add Tests for Processing of GetEventInformation Service Responses. 6. Add Tests for Fallback from ReadPropertyMultiple to ReadProperty. 7. Allow Priorities in WriteProperty and WritePropertyMultiple Tests. 8. Add Test for Writing Array Size. 9. Clarify Test for Writing with a Value that is Out of Range. 10. Update Test for Writing with an Invalid Datatype. 11. Relax ReadPropertyMultiple Error Test. 12. Add Test for Unicast Who-Is. 13. Revise Unknown Network Layer Message Test. 14. Add New Trend Log Tests. 15. Add Event_Type Test. 16. Revise DeviceCommunicationControl Test. 17. Add Alarm Re-acknowledgement Tests. 18. Modify I-Am Tests. 19. Add A-side Trend Tests. 20. Make the EPICS Definition Generic. 21. Clarify Priority in the GetEnrollmentSummary Priority Filter Test. 22. Add Non-documented Property and Read-Only Property Tests.

POLICY STATEMENT DEFINING ASHRAE'S CONCERN FOR THE ENVIRONMENTAL IMPACT OF ITS ACTIVITIES

ASHRAE is concerned with the impact of its members' activities on both the indoor and outdoor environment. ASHRAE's members will strive to minimize any possible deleterious effect on the indoor and outdoor environment of the systems and components in their responsibility while maximizing the beneficial effects these systems provide, consistent with accepted standards and the practical state of the art.

ASHRAE's short-range goal is to ensure that the systems and components within its scope do not impact the indoor and outdoor environment to a greater extent than specified by the standards and guidelines as established by itself and other responsible bodies.

As an ongoing goal, ASHRAE will, through its Standards Committee and extensive technical committee structure, continue to generate up-to-date standards and guidelines where appropriate and adopt, recommend, and promote those new and revised standards developed by other responsible organizations.

Through its *Handbook*, appropriate chapters will contain up-to-date standards and design considerations as the material is systematically revised.

ASHRAE will take the lead with respect to dissemination of environmental information of its primary interest and will seek out and disseminate information from other responsible organizations that is pertinent, as guides to updating standards and guidelines.

The effects of the design and selection of equipment and systems will be considered within the scope of the system's intended use and expected misuse. The disposal of hazardous materials, if any, will also be considered.

ASHRAE's primary concern for environmental impact will be at the site where equipment within ASHRAE's scope operates. However, energy source selection and the possible environmental impact due to the energy source and energy transportation will be considered where possible. Recommendations concerning energy source selection should be made by its members.