BACnet® TESTING LABORATORIES
ADDENDA

Addendum aq to
BTL Test Package 16.1

Revision 6
Revised September 30, 2020

Approved by the BTL Working Group on July 9, 2020.
Approved by the BTL Working Group Voting Members on September 30, 2020.
Published on October 1, 2020.

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

FOREWORD

The purpose of this addendum is to present current changes being made to the BTL Test Package. These modifications are the result of change proposals made pursuant to the continuous maintenance procedures and of deliberations within the BTL-WG Committee. The changes are summarized below.

In the following document, language to be added to existing clauses within the BTL Test Package 16.1 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout

In contrast, changes to BTL Specified Tests also contain a yellow highlight to indicate the changes made by this addendum. When this addendum is applied, all highlighting will be removed. Change markings on tests will remain to indicate the difference between the new test and an existing 135.1 test. If a test being modified has never existed in 135.1, the applied result should not contain any change markings. When this is the case, square brackets will be used to describe the changes required for this test.

Each addendum can stand independently unless specifically noted via dependency within the addendum. If multiple addenda change the same test or section, each future released addendum that changes the same test or section will note in square brackets whether or not those changes are reflected.

BTL-16.1aq-1: Tests for FAULT_LISTED Algorithm- BTLWG-698

Overview:

Addendum 135-2012*aq*-3 at Protocol_Revision 18 added new FAULT_LISTED algorithm to vertical transport objects that provide fault reporting, and to the Event Enrollment object.

Changes:

[In BTL Checklist, change Alarm and Event Management - Notification - Internal - B and Alarm and Event Management - Notification - External - B]

Alarm and Event Management - Notification - Internal - B		
		…
	C[3, 8]	Implements the CHANGE_OF_RELIABILITY - FAULT_LISTED algorithm
		…
[1] Required if EventNotifications with service parameter AckRequired = True can be issued. [2] At least one of these options must be supported to claim support for this BIBB. [3] At least one of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible. [4] At least one of these options must be supported to claim support for this BIBB. The BACnetDateTime form of the timestamp is the recommended option. [5] Contact BTL for interim tests for this algorithm. [6] Protocol_Revision 16 or higher must be claimed. [7] Protocol_Revision 17 or higher must be claimed. [8] Protocol_Revision 18 or higher must be claimed.		
Alarm and Event Management - Notification - External - B		
		…
	C[1, 5]	Implements the CHANGE_OF_RELIABILITY - FAULT_LISTED algorithm
		…
[1] One of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible. [2] Contact BTL for interim tests for this algorithm. [3] Protocol_Revision 16 or higher must be claimed. [4] Protocol_Revision 17 or higher must be claimed. [5] Protocol_Revision 18 or higher must be claimed.		

[In BTL Test Plan, change 5.2.36]

5.2.36 Implements the CHANGE_OF_RELIABILITY - FAULT_LISTED Algorithm

The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and supports the specified algorithm.

~~Contact BTL for interim tests for this algorithm.~~

BTL - 8.4.17.X1.1 - NORMAL to FAULT Transition (ConfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

BTL - 8.4.17.X1.2 - FAULT-to-FAULT transition (ConfirmedEventNotification)		
	Test Conditionality	If the IUT supports only one fault condition, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 8.5.17.X1.1 - NORMAL to FAULT Transition (UnconfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 8.5.17.X1.2 - FAULT-to-FAULT transition (UnconfirmedEventNotification)		
	Test Conditionality	If the IUT supports only one fault condition, this test shall be skipped.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, change 5.3.24]

5.3.24 Implements the CHANGE_OF_RELIABILITY - FAULT_LISTED Algorithm

The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and supports the specified algorithm.

~~Contact BTL for interim tests for this algorithm.~~

BTL - 8.4.17.X1.1 NORMAL to FAULT Transition (ConfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 8.4.17.X1.2 FAULT-to-FAULT transition (ConfirmedEventNotification)		
	Test Conditionality	If the IUT supports only one fault condition, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 8.5.17.X1.1 NORMAL to FAULT Transition (UnconfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 8.5.17.X1.2 FAULT-to-FAULT transition (UnconfirmedEventNotification)		
	Test Conditionality	If the IUT supports only one fault condition, this test shall be skipped.
	Test Directives	
	Testing Hints	

[In BTL Specified Tests, add a new test in this section]

8.4.17.X1 CHANGE_OF_RELIABILITY – FAULT_LISTED Tests (ConfirmedEventNotification)

8.4.17.X1.1 NORMAL to FAULT Transition (ConfirmedEventNotification)
Reason for Change: No tests exist.

Purpose: This test case verifies the correct operation of the FAULT_LISTED event algorithm for objects transitioning from NORMAL to FAULT event states.

Test Concept: The test concept corresponds to 8.5.17.X1.1.

Configuration Requirements: The configuration requirements are identical to those in 8.5.17.X1.1, except that the 'Issue Confirmed Notifications' parameter shall have a value of TRUE.

Test Steps: The test steps for this test case are identical to the test steps in 8.5.17.X1.1, except that the UnconfirmedEventNotification requests are ConfirmedEventNotification requests and the TD acknowledges receiving the notifications.

Notes to Tester: The passing results for this test case are identical to the ones in 8.5.17.X1.1, except that the event notifications shall be conveyed using a ConfirmedEventNotification service request.

8.4.17.X1.2 FAULT-to-FAULT transition (ConfirmedEventNotification)
Reason for Change: No tests exist.

Purpose: This test case verifies the correct operation of the FAULT_LISTED event algorithm for objects transitioning from FAULT to FAULT event states.

Test Concept: The test concept corresponds to 8.5.17.X1.2.

Configuration Requirements: The configuration requirements are identical to those in 8.5.17.X1.2, except that the 'Issue Confirmed Notifications' parameter shall have a value of TRUE.

Test Steps: The test steps for this test case are identical to the test steps in 8.5.17.X1.2, except that the UnconfirmedEventNotification requests are ConfirmedEventNotification requests and the TD acknowledges receiving the notifications.

Notes to Tester: The passing results for this test case are identical to the ones in 8.5.17.X1.2, except that the event notifications shall be conveyed using a ConfirmedEventNotification service request.

8.5.17.X1 CHANGE_OF_RELIABILITY - FAULT_LISTED Tests (UnconfirmedEventNotification)

8.5.17.X1.1 NORMAL to FAULT Transition (UnconfirmedEventNotification)
Reason for Change: No tests exist.

Purpose: This test case verifies the correct operation of the FAULT_LISTED event algorithm for objects transitioning from NORMAL to FAULT event states.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_LISTED algorithm. Ensure that no fault conditions exist in the object. Set pMonitoredList to FV1, a non-empty list of supported faults. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredList to an empty list. Verify the correct transition is generated.

Configuration Requirements: O1 is configured to detect faults and to report those using unconfirmed event notifications. O1 is initially configured to have no fault conditions present, and has an Event_State of NORMAL.

Test Steps:

1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY pCurrentState = NORMAL
3. IF (pMonitoredList is writable) THEN
   WRITE pMonitoredList = FV1
   ELSE
   MAKE (pMonitoredList = FV1)
4. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request,
   'Process Identifier' = (any valid process Identifier),
   'Initiating Device Identifier' = IUT
   'Event Object Identifier' = O1
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the notification class configured for O1),
   'Priority' = (the value configured for the transition),
   'Event Type' = CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),

'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = FAULT,
'Event Values' = (FAULT_LISTED,
(T, T, ? ?),
(A list of valid values for properties required to be reported
for O1, and 0 or more other properties of O1))

5. VERIFY pCurrentReliability = FAULTS_LISTED
6. VERIFY pCurrentState = FAULT
7. IF (pMonitoredList is writable) THEN
   WRITE pMonitoredList = (an empty list)
   ELSE
   MAKE (pMonitoredList = (an empty list)
8. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request,
   'Process Identifier' = (any valid process Identifier),
   'Initiating Device Identifier' = IUT
   'Event Object Identifier' = O1
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the notification class configured for O1),
   'Priority' = (the value configured for the transition),
   'Event Type' = CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = FAULT,
   'To State' = NORMAL,
   'Event Values' = (NO_FAULT_DETECTED,
   (F, F, ? ?),
   (A list of valid values for properties required to be reported
   for O1, and 0 or more other properties of O1))
9. pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY pCurrentState = NORMAL

8.5.17.X1.2 FAULT-to-FAULT transition (UnconfirmedEventNotification)
Reason for Change: No tests exist.

Purpose: This test case verifies the correct operation of the FAULT_LISTED event algorithm for objects transitioning from FAULT to FAULT event states.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_LISTED algorithm. Ensure that a fault condition, FV1, exists in the object. Set pMonitoredList to FV2, a non-empty list different from FV1. Verify the correct transition is generated.

Configuration Requirements: O1 is configured to detect faults and to report those using unconfirmed event notifications. O1 is initially configured to have a fault by having pMonitoredList contain a non-empty list, FV1, and has an Event_State of FAULT.

Test Steps:

1. VERIFY pCurrentReliability = FAULT_LISTED
2. VERIFY pCurrentState = FAULT
3. IF (pMonitoredList is writable) THEN
   WRITE pMonitoredList = FV2
   ELSE
   MAKE (pMonitoredList = FV2)
4. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request,
   'Process Identifier' = (any valid process Identifier),

'Initiating Device Identifier' =	IUT
'Event Object Identifier' =	O1
'Time Stamp' =	(any valid time stamp),
'Notification Class' =	(the notification class configured for O1),
'Priority' =	(the value configured for the transition),
'Event Type' =	CHANGE_OF_RELIABILITY,
'Message Text' =	(optional, any valid message text),
'Notify Type' =	ALARM | EVENT,
'AckRequired' =	TRUE | FALSE,
'From State' =	FAULT,
'To State' =	FAULT,
'Event Values' =	(FAULT_LISTED, (T, T, ? ?), (A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1))

5. VERIFY pCurrentReliability = FAULTS_LISTED
6. VERIFY pCurrentState = FAULT

BTL-16.1aq-2: Add Testing for Elevator Object Types - BTLWG-699

Overview:

The existing test package does not have any coverage for Elevator Group, Escalator, and Lift object types.

Changes:

[In BTL Checklist, completely replace the entries for Elevator Group, Lift and Escalator objects]

….		
Elevator Group Object		
	R[1]	Base Requirements
	O	Supports Landing_Call_Control property
[1] Protocol_Revision 18 or higher must be claimed		
Lift Object		
	R[1]	Base Requirements
	S	Supports writable Out_Of_Service property
	O	Supports Energy_Meter_Ref and Energy_Meter properties
	O	Contains an object with Reliability_Evaluation_Inhibit property
[1]Protocol_Revision 18 or higher must be claimed		
Escalator Object		
	R[1]	Base Requirements
	S	Supports writable Out_Of_Service property
	O	Supports Energy_Meter_Ref and Energy_Meter properties
	O	Contains an object with Reliability_Evaluation_Inhibit property
[1] Protocol_Revision 18 or higher must be claimed		
….		

[In BTL Test Plan, modify section 3.58 Elevator Group Object]

3.58 Elevator Group Object

3.58.1 Base Requirements

~~Contact BTL for interim tests for this object~~. Base requirements must be met by any IUT that can contain Elevator Group objects.

BTL - 7.3.2.X45.1 - Machine_Room_ID property references a Positive Integer Value Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X45.2 - Linking of Lift and Escalator Objects under Group_Members property of the Elevator Group Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.58.2 Supports Landing_Call_Control Property

The IUT contains, or can be made to contain, an Elevator Group object that contains the Landing_Call_Control Property.

BTL - 7.3.2.X45.3 - Linking of Landing_Call_Control Property Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, modify section 3.59 Lift Object]

3.59 Lift Object

3.59.1 Base Requirements

~~Contact BTL for interim tests for this object~~. Base requirements must be met by any IUT that can contain Lift objects.

BTL - 7.3.2.X45.2 - Linking of Lift and Escalator Objects under Group_Members property of the Elevator Group Object		
	Test Conditionality	If the IUT contains at least one Elevator_Group object, this test may be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X46.1- Array Size of the Lift Object properties based on car door size		
	Test Conditionality	This test must be executed if two or more of the BACnetARRAY properties Car_Door_Text, Assigned_Landing_Calls, Making_Car_Call, Registered_Car_Call, Car_Door_Status, Car_Door_Command and Landing_Door_Status are present.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X46.2- Lift Properties Operational Test		
	Test Conditionality	Must be executed. Repeat the test for each supported method of control (modification of Making_Car_Call property, modification of Assigned_Landing_Calls)
	Test Directives	
	Testing Hints	

3.59.2 Supports writable Out_Of_Service property

The Out_Of_Service property in Lift objects contained in the IUT is either writable or can be modified by any other means.

BTL - 7.3.2.X46.3 - Out_Of_Service, Status_Flags for Lift Object		
	Test Conditionality	If the Out_Of_Service property is writable or can be modified by other means this test must be executed.
	Test Directives	
	Testing Hints	

3.59.3 Supports Energy_Meter_Ref and Energy_Meter Properties

The Energy_Meter_Ref and Energy_Meter properties are both present in at least one Lift object.

BTL - 7.3.2.X46.4 - Energy_Meter_Ref Property Tests		
	Test Conditionality	If the IUT does not contain a Lift object with both Energy_Meter_Ref and Energy_Meter properties, this test may be skipped.
	Test Directives	
	Testing Hints	

3.59.4 Contains an object with Reliability_Evaluation_Inhibit Property

The IUT contains, or can be made to contain, a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	

	Testing Hints	

[In BTL Test Plan, modify section 3.60 Escalator Object]

3.60 Escalator Object

3.60.1 Base Requirements

~~Contact BTL for interim tests for this object~~. Base requirements must be met by any IUT that can contain Escalator objects.

BTL - 7.3.2.X45.2 Linking of Lift and Escalator Objects under Group_Members property of the Elevator Group Object		
	Test Conditionality	If the IUT contains at least one Elevator_Group object, this test may be skipped.
	Test Directives	
	Testing Hints	

3.60.2 Supports Writable Out_Of_Service property

The Out_Of_Service property in Escalator objects contained in the IUT is either writable or can be modified by any other means.

BTL - 7.3.2.X47.1 - Out_Of_Service, Status_Flags for Escalator Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.60.3 Supports Energy_Meter_Ref and Energy_Meter Properties

The Energy_Meter_Ref and Energy_Meter properties are both present in at least one Escalator object.

BTL - 7.3.2.X46.4 Energy_Meter_Ref Property Tests		
	Test Conditionality	If the IUT does not contain an Escalator object with both Energy_Meter_Ref and Energy_Meter properties, this test may be skipped.
	Test Directives	
	Testing Hints	

3.60.4 Contains an object with Reliability_Evaluation_Inhibit Property

The IUT contains, or can be made to contain, a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	
	Testing Hints	

[In BTL Specified Tests, add new tests]

7.3.2.X45 Elevator Group Object Tests

7.3.2.X45.1 Machine_Room_ID property references a Positive Integer Value Object
Reason for Change: No tests exist.

Purpose: To verify that the Machine_Room_ID property of an Elevator Group object can only reference a Positive Integer Value object or an object with instance number of 4194303.

Test Concept: The Machine_Room_ID property of an Elevator Group object, EG1, is read to verify that it contains an object reference to a Positive Integer Value object, PIV, or an object with instance number of 4194303. If the property is writable, an attempt is made to write an object reference, O1, that is not a Positive Integer Value object and has an instance number 0-4194302 (inclusive) to verify that an error is returned.

Test Steps:

1. IF (Machine_Room_ID contains room identification number) THEN
   VERIFY (EG1), Machine_Room_ID = (PIV)
   ELSE
   VERIFY (EG1), Machine_Room_ID = (any object type, 4194303)
2. IF (Machine_Room_ID is writeable) THEN
   Transmit WriteProperty-Request
   'Object Identifier'= EG1,
   'Property Identifier'= Machine_Room_ID,
   'Property Value'= O1
   Receive BACnet-Error-PDU
   'Error Class'= PROPERTY,
   'Error Code'= VALUE_OUT_OF_RANGE

7.3.2.X45.2 Linking of Lift and Escalator Objects under Group_Members property of the Elevator Group Object
Reason for Change: No tests exist.

Purpose: This test verifies that objects in the Group_Members property of Elevator Group objects contain a reference back to the Elevator Group that has it listed as a member.

Test Concept: The Group_Members property of each Elevator Group object is read to identify member Lift and Escalator objects. The Elevator_Group property is read from each member Lift object and Escalator object to verify it contains a reference back to the appropriate Elevator Group object. The Elevator_Group property of the remaining Lift and Escalator objects are read to verify that it contains an object identifier instance of 4194303.

Configuration Requirements: If the IUT supports a Group_Members property that can be made to contain a reference to one or more Lift objects, than it shall be configured as such. If the IUT supports a Group_Members property that can be made to contain a reference to one or more Escalator objects, it shall be configured as such.

Test Steps:

1. REPEAT EGO = (each Elevator Group object in the IUT) {
   READ L1 = (EGO, Group_Members)
   IF (L1 is not empty) THEN
   REPEAT O1 = (each Lift or Escalator object in L1) {
   READ EGP = (O1, Elevator_Group)
   VERIFY EGP = EGO
   }
   }
3. REPEAT O1 = (each remaining Lift or Escalator object in the IUT) {
   READ EGP = (O1, Elevator_Group)
   VERIFY EGP = (any object type, 4194303)
   }

7.3.2.X45.3 Landing_Call_Control test
Reason for Change: No tests exist.

Purpose: To verify that writing to the Landing_Call_Control property updates the Landing_Call_Control and Landing_Calls properties in the Elevator Group object and updates the Assigned_Landing_Calls property of a linked Lift object

Test Concept: The Landing_Call_Control property of an Elevator Group object (EG1) is written with a value that represents a request to travel upwards from FN1. The Landing_Call_Control and Landing_Calls properties of EG1 and the Assigned_Landing_Calls property of the linked Lift object (L1) are checked to verify they updated correctly. The Landing_Call_Control property is written with a value that represents a request to travel downwards from FN2 and the aforementioned properties are checked again. The optional 'floor-text' parameter is used in one of the WRITE steps to verify the server will ignore this parameter when present. In the test steps, DF represents a valid destination floor.

Configuration Requirements: Lift object (L1) is contained in the Group_Members property of the Elevator Group object (EG1) and has a door at array index Y on the same side of the landing call. FN1 and FN2 values should be sufficiently far away from the current position of L1 to allow for reading of the property values. No other processes shall be generating landing calls during this test.

Test Steps:

1. WRITE EG1, Landing_Call_Control = (FN1, UP | DF (DF > FN1), "test string")
2. VERIFY EG1, Landing_Call_Control = (FN1, UP | DF, floor-text (optional))
3. VERIFY EG1, Landing_Calls = (FN1, UP | DF, floor-text (optional))
4. IF (L1 contains the Assigned_Landing_Calls property)
   VERIFY L1, Assigned_Landing_Calls, ARRAY INDEX (Y) = (FN1, UP)
5. WAIT (a time interval sufficient for the car to complete the call + **Internal Processing Fail Time**)
6. VERIFY EG1, Landing_Calls = ()
7. IF (L1 contains the Assigned_Landing_Calls property)
   VERIFY L1, Assigned_Landing_Calls, ARRAY_INDEX (Y) = ()
8. WRITE EG1, Landing_Call_Control = (FN2, DOWN | DF (DF < FN2))
9. VERIFY EG1, Landing_Call_Control = (FN2, DOWN | DF, floor-text (optional))
10. VERIFY EG1, Landing_Calls = (FN2, DOWN | DF, floor-text (optional))
11. IF (L1 contains the Assigned_Landing_Calls property)
    VERIFY L1, Assigned_Landing_Calls = (FN1, DOWN)
12. WAIT (a time interval sufficient for the car to complete the call + **Internal Processing Fail Time**)
13. VERIFY EG1, Landing_Calls = ()
14. IF (L1 contains the Assigned_Landing_Calls property)
    VERIFY L1, Assigned_Landing_Calls, ARRAY_INDEX (Y) = ()

Notes to Tester: If the Elevator Group contains more than 1 lift, the value written to Landing_Call_Control may get assigned to any other lift in the group based on the lift algorithm.

7.3.2.X46 Lift Object Tests

7.3.2.X46.1 Array Size of the Lift Object properties based on car door size.

Reason for Change: No tests exist.

Purpose: To verify that the size of the arrays for the Car_Door_Text, Assigned_Landing_Calls, Making_Car_Call, Registered_Car_Call, Car_Door_Status, Car_Door_Command and Landing_Door_Status properties are the same.

Test Concept: The array size for each of the above properties, if present, is read and the sizes are compared to verify they are all equal.

Test Steps:

1. VERIFY (L1), Car_Door_Text = (Number of car doors present in the Lift), ARRAY INDEX = 0
2. VERIFY (L1), Assigned_Landing_Calls = (Number of car doors present in Lift), ARRAY INDEX = 0
3. VERIFY (L1), Making_Car_Call = (Number of car doors present in the Lift), ARRAY INDEX = 0
4. VERIFY (L1), Registered_Car_Call = (Number of car doors present in the Lift), ARRAY INDEX = 0
5. VERIFY (L1), Car_Door_Status = (Number of car doors present in the Lift), ARRAY INDEX = 0
6. VERIFY (L1), Car_Door_Command = (Number of car doors present in the Lift), ARRAY INDEX = 0
7. VERIFY (L1), Landing_Door_Status = (Number of car doors present in the Lift), ARRAY INDEX = 0
8. CHECK (Array index 0 of all these properties shall be same)

7.3.2.X46.2 Lift Properties Operational Test

Reason for Change: No tests exist.

Purpose: To verify that the property values in the Lift object update when it responds to a call.

Test Concept: The test starts with the Lift object, L1, in the lowest floor that it serves, LF, and property values are checked. A request is made to move the lift to the highest floor that it serves, HF, and property values are checked while the lift is moving and again when the lift arrives at HF. If the IUT does not contain the property specified in the test step, that step shall be skipped. In the test steps, DSR is a specific array index corresponding to the car door servicing the request.

Configuration Requirements: At the start of the test, the lift corresponding to L1 is at LF and there are no active calls for L1. Throughout the test, L1 is in a normal operating state such that Car_Mode = NORMAL, Out_Of_Service = FALSE, and no other processes shall be attempting to control L1.

Test Steps:

1. READ LF = Car_Position
2. READ DS1 = Car_Door_Status
3. VERIFY Floor_Text = (any value), ARRAY INDEX = LF
4. VERIFY Floor_Text = (any value), ARRAY INDEX = HF
5. REPEAT N = (each array element) DO {
   VERIFY Assigned_Landing_Calls = {}, ARRAY INDEX = N
   }
6. REPEAT N = (each array element) DO {
   VERIFY Registered_Car_Calls = {}, ARRAY INDEX = N
   }
7. VERIFY Car_Moving_Direction <> UP | DOWN
8. VERIFY Car_Mode = NORMAL
9. VERIFY Next_Stopping_Floor = LF
10. VERIFY Passenger_Alarm = FALSE
11. VERIFY Reliability = NO_FAULT_DETECTED
12. VERIFY Out_Of_Service = FALSE
13. VERIFY Car_Drive_Status = STATIONARY | UNKNOWN
14. REPEAT N = (each array element) DO {
    VERIFY Landing_Door_Status = (a list containing an entry {LF, DS1[N]}), ARRAY INDEX = N
    }
15. MAKE (A command that will cause L1 to travel to HF)

    --Complete steps 16 – 19 before L1 reaches HF
16. IF (command was generated via Landing call) THEN
    VERIFY Assigned_Landing_Calls = (HF, DOWN), ARRAY INDEX = DSR
    }
    ELSE --command was generated via Car call
    VERIFY Making_Car_Call = (HF), ARRAY INDEX = DSR
    VERIFY Registered_Car_Calls = (HF), ARRAY INDEX = DSR
    VERIFY Car_Assigned_Direction = (UP)
    }
17. VERIFY Car_Moving_Direction = UP
18. VERIFY Next_Stopping_Floor = HF
19. VERIFY Car_Drive_Status <> STATIONARY
20. WAIT (for L1 to reach HF) + **Internal Processing Fail Time**
21. REPEAT N = (each array element) DO {
    VERIFY Registerd_Car_Calls = {}, ARRAY INDEX = N
    }
22. VERIFY Car_Position = HF
23. VERIFY Car_Moving_Direction <> UP | DOWN
24. VERIFY Next_Stopping_Floor = HF
25. READ DS2 = Car_Door_Status
26. REPEAT (N = each array element) DO{

VERIFY Landing_Door_Status = (a list containing an entry {LF, DS2[N]}), ARRAY INDEX = N
}

7.3.2.X46.3 Out_Of_Service, Status_Flags for Lift object
Reason for Change: No tests exist.

Purpose: To verify the interrelationship between Out_Of_Service and Status_Flags and that properties dictated by the standard to be writable when Out_Of_Service is TRUE are writable when Out_Of_Service is TRUE.

Test Concept: Out_Of_Service is set to TRUE and Status_Flags is checked to verify the Out_Of_Service flag is set. While Out_Of_Service is TRUE, each of the properties (represented by LP), if present in the object, is read to obtain the current property value, X, and written with a different property value, Y. The property value is read again to verify it changed to Y.

LP = (Assigned_Landing_Calls, Registered_Car_Call, Car_Position, Car_Moving_Direction, Car_Assigned_Direction, Car_Door_Status, Car_Door_Zone, Car_Load, Next_Stopping_Floor, Passenger_Alarm, Energy_Meter, Car_Drive_Status, Fault_Signals, Landing_Door_Status, Making_Car_Call, Car_Door_Command, and Car_Mode)

Test Steps:

1. WRITE Out_Of_Service = TRUE
2. VERIFY Out_Of_Service = TRUE
3. VERIFY Status_Flags = (?, ?, ?, TRUE)
4. REPEAT P = (each property in LP present in the object) DO{
   READ X = P
   WRITE P = Y
   WAIT **Internal Processing Fail Time**
   VERIFY (P =Y)
   }

7.3.2.X46.4 Energy_Meter_Ref Property Tests
Reason for Change: No tests exist.

Purpose: To verify linking of Energy_Meter property and Energy_Meter_Ref property.

Test Concept: If the Energy_Meter_Ref property of an object (O1) is present and initialized (contains an instance other than 4194303), then the Energy_Meter property, if present, shall have a value of 0.0. If Energy_Meter_Ref is present and is un-initialized, then the value of Energy_Meter property shall have any valid value.

Test Steps:

1. IF (Energy_Meter_Ref is present and initialized with instance other than 4194303) THEN
   VERIFY Energy_Meter = 0.0
   ELSE
   VERIFY Energy_Meter = (Any Valid Value)

7.3.2.X47 Escalator Object Tests

7.3.2.X47.1 Out_Of_Service, Status_Flags for Escalator object
Reason for Change: No tests exist.

Purpose: To verify the interrelationship between Out_Of_Service and Status_Flags and that properties dictated by the standard to be writable when Out_Of_Service is TRUE are writable when Out_Of_Service is TRUE.

Test Concept: Out_Of_Service is set to TRUE and Status_Flags is checked to verify the Out_Of_Service flag is set. While Out_Of_Service is TRUE, each of the properties (represented by EP), if present in the object, is read to obtain the current property value, X, and written with a different property value, Y. The property value is read again to verify it changed to Y.

EP = (Power_Mode, Operation_Direction, Escalator_Mode , Energy_Meter, Fault_Signals, and Passenger_Alarm)

Test Steps:

1. WRITE Out_Of_Service = TRUE
2. VERIFY Out_Of_Service = TRUE
3. VERIFY Status_Flags = (?, ?, ?, TRUE)
4. REPEAT P = (each property in LP present in the object) DO {
   READ X = P
   WRITE (P = Y)
   WAIT **Internal Processing Fail Time**
   VERIFY (P =Y)
   }

BTL-16.1aq-3: Add Testing for SubscribeCOVPropertyMultiple - BTLWG-119

Overview:

Addenda aq-2 added COVMultiple services: SubscribeCOVPropertyMultiple, ConfirmedCOVNotificationMultiple, UnconfirmedCOVNotificationMultiple services to all multiple values to be subscribed to and allow individual timestamps in each notification. This service can be used with any property of any object, so long as it is supported on the A and B sides.

Changes:

[In BTL Checklist, replace the data for DS-COVM-A and DS-COVM-B to add support for these BIBBs]

Data Sharing - Change Of Value Multiple - A		
	R	Base Requirements
	R	Subscribes with lifetimes up to 8 hours in duration
	O	Can cancel subscriptions
	C[1]	Can subscribe for confirmed notifications
	C[1]	Can subscribe for unconfirmed notifications
	O	Supports subscribing to timestamped notifications
	C[2]	Can subscribe to non-array properties
	C[2]	Can subscribe to array elements
	C[2]	Can subscribe to the size of an array
	C[2]	Can subscribe to whole arrays
	O	Can subscribe to list properties
	O	Can subscribe with a COV Increment
	C[3]	Can subscribe to NULL property values
	C[3]	Can subscribe to BOOLEAN property values
	C[3]	Can subscribe to Enumerated property values
	C[3]	Can subscribe to INTEGER property values
	C[3]	Can subscribe to Unsigned property values
	C[3]	Can subscribe to REAL property values
	C[3]	Can subscribe to Double property values
	C[3]	Can subscribe to Time property values
	C[3]	Can subscribe to Date property values
	C[3]	Can subscribe to CharacterString property values
	C[3]	Can subscribe to OctetString property values
	C[3]	Can subscribe to BitString property values
	C[3]	Can subscribe to BACnetObjectIdentifier property values
	C[3]	Can subscribe to Value_Source property values
	C[3]	Can subscribe to constructed property values
	C[3]	Can subscribe to proprietary property values of basic data types
[1] At least one of these options is required in order to claim conformance to this BIBB. [2] At least one of these options is required in order to claim conformance to this BIBB. [3] At least one of these options is required in order to claim conformance to this BIBB.		
Data Sharing - Change Of Value Multiple - B		
	R	Base Requirements
	R	Supports COVM Lifetimes up to 8 hours in duration
	R	Supports a minimum of 5 COV-multiple contexts with 5 COV-references per context
	C[1]	Supports COVM for non-array property
	C[1]	Supports COVM for array element
	C[1]	Supports COVM for the size of an array
	C[1]	Supports COVM for the whole array
	O	Supports COVM for list property
	C[2]	Supports COVM for NULL property values
	C[2]	Supports COVM for BOOLEAN property values
	C[2]	Supports COVM for Enumerated property values
	C[2]	Supports COVM for INTEGER property values
	C[2]	Supports COVM for Unsigned property values

	C[2]	Supports COVM for REAL property values
	C[2]	Supports COVM for Double property values
	C[2]	Supports COVM for Time property values
	C[2]	Supports COVM for Date property values
	C[2]	Supports COVM for CharacterString property values
	C[2]	Supports COVM for OctetString property values
	C[2]	Supports COVM for BitString property values
	C[2]	Supports COVM for BACnetObjectIdentifier property values
	C[2]	Supports COVM for Value_Source property values
	C[2]	Supports COVM for constructed property values
[1] At least one of these options is required in order to claim conformance to this BIBB. [2] At least one of these options is required in order to claim conformance to this BIBB.		

[In BTL Test Plan, replace sections 4.25 and 4.26 to add support for these BIBBs]

4.25 Data Sharing - Change Of Value Multiple - A

4.25.1 Base Requirements

Base requirements must be met by any IUT claiming conformance to this BIBB.

BTL - 8.X12.1.5 - Subscribe to Two Properties in a Single Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 8.X12.1.6 - Subscribe to Properties in Multiple Objects Using a Single Request		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.25.2 Subscribes with lifetimes up to 8 hours in duration

The IUT is capable of subscribing with a lifetime less than or equal to 28800 seconds (8 hours).

BTL - 8.X12.1.3 - Requests 8 Hour Lifetimes		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.25.3 Can Cancel Subscriptions

The IUT can explicitly cancel COV subscriptions (in contrast to just letting the subscription expire).

BTL - 8.X12.1.8 - Canceling a Subscription		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.25.4 Can Subscribe for Confirmed Notifications

The IUT can subscribe for, receive, and process confirmed Change of Value Multiple notifications.

BTL - 8.X12.1.1 - Confirmed Notifications Subscription		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 8.X12.2.2 - Unknown Subscription		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test with an Invalid Process Identifier, Invalid Monitored Object Identifier, and Invalid Monitored property reference

	Testing Hints	
BTL - 8.X12.2.1 - Change of Value Multiple Notification Arrives After Subscription Has Expired		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.25.5 Can Subscribe for Unconfirmed Notifications

The IUT can subscribe for, receive, and process unconfirmed Change of Value Multiple notifications.

BTL - 8.X12.1.2 - Unconfirmed Notifications Subscription		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.25.6 Supports Subscribing to Timestamped Notifications

The IUT can subscribe for and receive Timestamped Notifications.

BTL - 8.X12.1.4 - Subscribe to Timestamped Notifications		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.25.7 Can Subscribe to Non-array Properties

The IUT can subscribe for and receive a Change of Value Multiple notification that that references a non-array property and can correctly process the response.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing the property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property' = (any valid non-array property in the Monitored Object, which the vendor supports in a SubscribeCOVPropertyMultiple-Request)
	Testing Hints	

4.25.8 Can Subscribe to Array Elements

The IUT can subscribe for and receive a Change of Value Multiple notification that references a specific element of an array property and can correctly process the response.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing the property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any valid array property which the vendor supports in a SubscribeCOVPropertyMultiple-Request and array index with value different from 0)
	Testing Hints	

4.25.9 Can Subscribe to the Size of an Array

The IUT can subscribe for and receive a Change of Value Multiple notification that references the size of an array property and can correctly process the response.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing the property to monitor and which the vendor supports in a

		SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any valid array property which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and array index with value equal to 0)
	Testing Hints	

4.25.10 Can Subscribe to Whole Arrays

The IUT can subscribe for and receive a Change of Value Multiple notification for an array property and can correctly process the response.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing the property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any valid array property which the vendor supports in a SubscribeCOVPropertyMultiple-Request with no array index)
	Testing Hints	

4.25.11 Can Subscribe to List Properties

The IUT can subscribe for and receive a Change of Value Multiple notification that references a list property and can correctly process the response

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing the list property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any valid list property which the vendor supports in a SubscribeCOVPropertyMultiple-Request)
	Testing Hints	

4.25.12 Can Subscribe with a COV Increment

The IUT can subscribe with the parameter 'COV Increment'.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and ensure that IUT generates a SubscribeCOVPropertyMulitple-Request which contains 'COV Increment' parameter,
	Testing Hints	

4.25.13 Can Subscribe to NULL Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from property that contains a NULL value.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a NULL value)

	Testing Hints	Schedule_Default of the Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object are standard properties that should accept a written NULL.

4.25.14 Can Subscribe to BOOLEAN Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from BOOLEAN property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a BOOLEAN value)
	Testing Hints	

4.25.15 Can Subscribe to Enumerated Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from Enumerated property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain an Enumerated value)
	Testing Hints	

4.25.16 Can Subscribe to INTEGER Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from INTEGER property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain an Integer value)
	Testing Hints	

4.25.17 Can Subscribe to Unsigned Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from Unsigned property values.

BTL - 8.X12.1.7 - Change of Value Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain an Unsigned value)
	Testing Hints	

4.25.18 Can Subscribe to REAL Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from REAL property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a REAL value)
	Testing Hints	

4.25.19 Can Subscribe to Double Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from Double property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a Double value)
	Testing Hints	

4.25.20 Can Subscribe to Time Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from Time property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a Time value)
	Testing Hints	

4.25.21 Can Subscribe to Date Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from Date property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a Date value)
	Testing Hints	

4.25.22 Can Subscribe to CharacterString Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from CharacterString property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier'

		= (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a CharacterString value)
	Testing Hints	

4.25.23 Can Subscribe to OctetString Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from OctetString property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain an OctetString value)
	Testing Hints	

4.25.24 Can Subscribe to BitString Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from BitString property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a BitString value)
	Testing Hints	

4.25.25 Can Subscribe to BACnetObjectIdentifier Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from BACnetObjectIdentifier property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a BACnetObjectIdentifier value)
	Testing Hints	

4.25.26 Can Subscribe to Value_Source Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from Value_Source property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object which contains a Value_Source property and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = Value_Source
	Testing Hints	

4.25.27 Can Subscribe to Constructed Property Values

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from constructed property values.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubcribeCOVPropertyMultiple-Request that can contain a constructed property value)
	Testing Hints	

4.25.28 Can Subscribe to Proprietary Property Values of Basic Data Types

The IUT can subscribe for, receive, and process Change of Value Multiple notifications from proprietary property values of basic data types.

BTL - 8.X12.1.7 - Change of Value Multiple Notification		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Object' = (any valid object containing a property to monitor and which the vendor supports in a SubscribeCOVPropertyMultiple-Request) and 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVPropertyMultiple-Request that can contain a proprietary value of basic data types)
	Testing Hints	

4.26 Data Sharing - Change Of Value Multiple - B

4.26.1 Base Requirements

Base requirements must be met by any IUT claiming conformance to this BIBB.

BTL - 9.X41.1.1 - Supports Non-Timestamped Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Execute this test using a property that supports non-timestamped notifications
	Testing Hints	
BTL - 9.X41.1.2 - Supports Timestamped Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Execute this test using a property that supports timestamped notifications
	Testing Hints	
BTL - 9.X41.1.5 - Supports Subscriptions Multiple Properties Using Multiple Requests		
	Test Conditionality	Must be executed.
	Test Directives	Select objects and properties which support COV-multiple notifications
	Testing Hints	
BTL - 9.X41.1.9 - Canceling Subsets of COVM Subscriptions		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.X41.1.10 - Canceling Expired or Non-Existing Subscriptions		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.X41.1.8 - Updating Existing Subscriptions		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

BTL - 9.X41.1.7 - Supports Client-Supplied COV Increment		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.X41.2.1 - The Monitored Object Does Not Support COVM Notification		
	Test Conditionality	Must be executed, unless all objects support SubscribeCOVPropertyMultiple on at least one of its properties.
	Test Directives	Apply the test to a property in an object that does not support COVM (on any property).
	Testing Hints	
BTL - 9.X41.2.2 - The Monitored Property Does Not Support COVM Notification		
	Test Conditionality	Must be executed, unless all objects support SubscribeCOVPropertyMultiple on all properties.
	Test Directives	Apply the test to a property for which the IUT does not support COVM, which is contained in an object that does support COVM (on a different property).
	Testing Hints	
BTL - 9.X41.2.3 - Monitored Object Does Not Exist		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 9.X41.2.4 - Monitored Property Does Not Exist		
	Test Conditionality	Must be executed
	Test Directives	Be sure to test at least one property identifier that is within the ASHRAE allocated range for standard property identifiers, but that has not yet been defined.
	Testing Hints	
BTL - 9.X41.2.5 - Array Index Provided But Property is Not an Array		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 9.X41.2.6 - Array Index Provided is Out Of Range		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 9.X41.2.7 - No Space To Add List Element		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 9.X41.2.8 - The Lifetime Parameter is Out Of Range		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 9.X41.2.9 - The Max Notification Delay Parameter is Out Of Range		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.X41.2.10 - The Max Notification Delay is Greater Than the Lifetime		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.26.2 Supports COVM Lifetimes Up to 8 Hours in Duration

The IUT will accept COVM subscriptions with lifetimes up to 8 hours.

BTL - 9.X41.1.11 - Subscription Expiration Test		
	Test Conditionality	Must be executed.
	Test Directives	Execute this test using a Lifetime of 8 hours.

	Testing Hints	

4.26.3 Supports a Minimum of 5 COV-Multiple Contexts with 5 COV-References per Context

The IUT supports 5 or more concurrent COVM subscriptions

BTL - 9.X41.1.6 - Ensuring 5 Concurrent COV-Multiple Contexts With 5 COV-References per Context		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.26.4 Supports COVM for Non-Array Property

The IUT supports COVM notifications for at least one non-array property

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple
	Testing Hints	

4.26.5 Supports COVM for Array Element

The IUT supports COVM notifications for at least one array element.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple
	Testing Hints	

4.26.6 Supports COVM for the Size of an Array

The IUT supports COVM notifications for at least one index 0 of an array

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple
	Testing Hints	

4.26.7 Supports COVM for the Whole Array

The IUT supports COVM notifications for at least one whole array

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple
	Testing Hints	

4.26.8 Supports COVM for List Property

The IUT supports COVM notifications for at least one list property

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVPropertyMultiple
	Testing Hints	

4.26.9 Supports COVM for NULL Property Values

The IUT supports COVM notifications for at least one property that contains a NULL value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	Schedule_Default of the Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object are standard properties that can contain or accept a written NULL.
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	Schedule_Default of the Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object are standard properties that can contain or accept a written NULL.

4.26.10 Supports COVM for BOOLEAN Property Values

The IUT supports change of value notifications for at least one BOOLEAN property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype.

		Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.11 Supports COVM for Enumerated Property Values

The IUT supports change of value notifications for at least one Enumerated property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.12 Supports COVM for INTEGER Property Values

The IUT supports change of value notifications for at least one INTEGER property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.13 Supports COVM for Unsigned Property Values

The IUT supports change of value notifications for at least one Unsigned Property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype.

		Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.14 Supports COVM for REAL Property Values

The IUT supports change of value notifications for at least one REAL property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.15 Supports COVM for Double Property Values

The IUT supports change of value notifications for at least one Double property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.16 Supports COVM for Time Property Values

The IUT supports change of value notifications for at least one Time property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype.

		Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.17 Supports COVM for Date Property Values

The IUT supports change of value notifications for at least one Date property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.18 Supports COVM for CharacterString Property Values

The IUT supports change of value notifications for at least one CharacterString property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.19 Supports COVM for OctetString Property Values

The IUT supports change of value notifications for at least one property with value of type OctetString.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype.

		Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.20 Supports COVM for BitString Property Values

The IUT supports change of value notifications for at least one property with value of type BitString.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.21 Supports COVM for BACnetObjectIdentifier Property Values

The IUT supports change of value notifications for at least one property with value of type BACnetObjectIdentifier.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.22 Supports COVM for Value_Source Property Values

The IUT supports change of value notifications for at least one Value_Source property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.

	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

4.26.23 Supports COVM for Constructed Property Values

The IUT supports change of value notifications for at least one constructed property value.

BTL - 9.X41.1.3 - Confirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	
BTL - 9.X41.1.4 - Unconfirmed Change of Value Notification From Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 property of the specified datatype. Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVM on one or more of its properties.
	Testing Hints	

[In BTL Specified Tests, add the following tests]

8.X12 SubscribeCOVPropertyMultiple Service Initiation Tests

8.X12.1 Positive SubscribeCOVPropertyMultiple Service Initiation Tests

8.X12.1.1 Confirmed Notifications Subscription

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify the client can subscribe to confirmed notifications using the SubscribeCOVPropertyMultiple service.

Test Concept: The IUT is made to subscribe for confirmed notifications.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVPropertyMultiple-Request),
2. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L,
   'Max Notification Delay' = (any valid notification delay),
   'List of COV Subscription Specifications' = (any valid list of subscriptions)
3. TRANSMIT BACnet-SimpleAck-PDU

8.X12.1.2 Unconfirmed Notifications Subscription

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify the client can subscribe to unconfirmed notifications using the SubscribeCOVPropertyMultiple service.

Test Concept: The IUT is made to subscribe for unconfirmed notifications.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVPropertyMultiple-Request),
2. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = FALSE,
   'Lifetime' = L,
   'Max Notification Delay' = (any valid notification delay),
   'List of COV Subscription Specifications' = (any valid list of subscriptions)
3. TRANSMIT BACnet-SimpleAck-PDU

8.X12.1.3 Requests 8 Hour Lifetimes

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify that the IUT is able to provide a lifetime which is less than or equal to 8 hours for any SubscribeCOVPropertyMultiple request it generates.

Test Concept: The tester selects any of the possible COVM subscriptions that the IUT is able to generate and it is configured to use a lifetime less than or equal to 8 hours. The IUT is made to send the subscription, and the lifetime is verified to be less than or equal to 8 hours.

Test Steps:

1. MAKE (the IUT send the selected SubscribeCOVPropertyMultiple-Request),
2. RECEIVE SubscribeCOVPropertyMultiple-Request,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any value <= 28800),
   'Max Notification Delay' = (any valid delay between 1 and 3600),
   'List of COV Subscription Specifications' = (a valid list of COV Specifications)
3. TRANSMIT BACnet-SimpleACK-PDU

8.X12.1.4 Subscribe to Timestamped Notifications

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify the client can subscribe to timestamped notifications using the SubscribeCOVPropertyMultiple service.

Test Concept: A subscription for timestamped COVM notifications is established with Lifetime L for property P1 of Object O1. L shall be less than 8 hours but large enough to complete the test.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVPropertyMultiple-Request),
2. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = L,
   'Max Notification Delay' = (any valid notification delay),
   'List of COV Subscription Specifications' = (any valid list with at least 1 entry where 'Timestamped' is TRUE)
3. TRANSMIT BACnet-SimpleAck-PDU

8.X12.1.5 Subscribe to Two Properties in a Single Object

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify that the IUT can subscribe to 2 or more properties from a single object.

Test Concept: A subscription for COVM notifications is established for properties from a single object.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVPropertyMultiple-Request),

2. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = L,
   'Max Notification Delay' = (any valid notification delay),
   'List of COV Subscription Specifications' = (a valid list of 2 or more properties from a single object)
3. TRANSMIT BACnet-SimpleAck-PDU

8.X12.1.6 Subscribe to Properties in Multiple Objects Using a Single Request

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify the client can subscribe to properties from multiple objects.

Test Concept: A subscription for notifications is established for properties from 2 or more objects.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVPropertyMultiple-Request),
2. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (PID: any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = L,
   'Max Notification Delay' = (any valid notification delay),
   'List of COV Subscription Specifications' = (PROPS: any valid list of properties from 2 or more objects)
3. TRANSMIT BACnet-SimpleAck-PDU

8.X12.1.7 Change of Value Multiple Notification

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify that the IUT accepts COVM notifications for properties which it subscribed to.

Test Concept: A subscription for COVM notifications is established, a notification is sent to the IUT, and the vendor defined actions are verified.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVPropertyMultiple-Request),
2. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (ID1: any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (L: any valid lifetime),
   'Max Notification Delay' = (any valid delay between 1 and 3600),
   'List of COV Subscription Specifications' = (PROPS: any valid list of subscriptions)
3. TRANSMIT BACnet-SimpleACK-PDU
4. IF (the subscription was for confirmed notifications) THEN
   TRANSMIT ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = TD,
   'Time Remaining' = (any value ~=L),
   'Timestamp' = (any valid value, or absent if subscribed to non-timestamped notifications),
   'List of COV Notifications' = (values appropriate to each entry in PROPS)
   RECEIVE BACnet-SimpleACK-PDU
   ELSE
   TRANSMIT UnconfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = TD,
   'Time Remaining' = (any value ~= L),
   'Timestamp' = (any valid value, or absent if subscribed to non-timestamped notifications),

'List of COV Notifications' = (values appropriate to each entry in PROPS)

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

8.X12.1.8 Canceling a Subscription

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify the client can cancel a COVM subscription.

Test Concept: A subscription for COVM notifications is established with a lifetime L, which is long enough to complete the test. The client is made to cancel the subscription by sending a SubscribeCOVPropertyMultiple request with Lifetime, and Max Notification Delay absent.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVPropertyMultiple-Request),
2. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (the process identifier used in step 1),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = L,
   'Max Notification Delay' = (any valid delay between 1 and 3600),
   'List of COV Subscription Specifications' = (PROPS: a valid list of COV Subscription Specifications)
3. TRANSMIT BACnet-SimpleAck-PDU
4. IF confirmed notifications were subscribed for THEN
   TRANSMIT ConfirmedCOVNotificationMultiple-Request
   'Subscriber Process Identifier' = PID,
   'Initiating Device Identifier' = (TD's device identifier),
   'Time Remaining' = (a value ~= L),
   'Timestamp' = (a valid value, or absent if Time Of Change was not requested in the subscription)
   'List of COV Notifications' = (a valid list containing an entry for each entry in PROPS)
   RECEIVE BACnet-SimpleAck-PDU
   ELSE
   TRANSMIT UnconfirmedCOVNotificationMultiple-Request
   'Subscriber Process Identifier' = PID,
   'Initiating Device Identifier' = (TD's device identifier),
   'Time Remaining' = (a value ~= L),
   'Timestamp' = (a valid value, or absent if Time Of Change was not requested in the subscription)
   'List of COV Notifications' = (a valid list containing an entry for each entry in PROPS)
5. MAKE (the IUT cancel the subscription)
6. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (the process identifier used in step 2),
   'Issue Confirmed Notifications' = (the same value used in step 2),
   -- 'Lifetime' = (absent)
   -- 'Max Notification Delay' = (absent)
   'List of COV Subscription Specifications' = (PROPS, or an empty list)
7. TRANSMIT BACnet-SimpleAck-PDU

8.X12.2 Negative SubscribeCOVPropertyMultiple Service Initiation Tests

8.X12.2.1 Change of Value Multiple Notification Arrives After Subscription Has Expired

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify that an appropriate error is returned if a COVM notification arrives after the subscription time period has expired.

Test Concept: A subscription for COVM notifications is established and then cancelled or allowed to expire. A ConfirmedCOVNotificationMultiple is then sent to the IUT to verify it returns either the appropriate error or a Simple-Ack.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVPropertyMultiple-Request),
2. RECEIVE SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (ID1: any valid process identifier),
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = (L: any valid lifetime),
   'Max Notification Delay' = (any valid notification delay),
   'List of COV Subscription Specifications' = (PROPS: any valid list of COV subscriptions)
3. TRANSMIT BACnet-SimpleACK-PDU
4. TRANSMIT ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = TD,
   'Time Remaining' = (a value ~= L),
   'Timestamp' = (any appropriate value or absent if it is not a timestamped subscription)
   'List of COV Notifications' = (values appropriate to the properties in PROPS)
5. RECEIVE BACnet-SimpleACK-PDU
6. IF (the IUT can cancel the subscription) THEN
   MAKE (the IUT cancel the subscription),
   RECEIVE SubscribeCOVPropertyMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = (absent)
   'Max Notification Delay' = (absent)
   'List of COV Subscription Specifications' = (PROPS or an empty list)
   ELSE
   WAIT (2 * L seconds)
7. TRANSMIT ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = TD,
   'Time Remaining' = (a value ~= L),
   'Timestamp' = (any appropriate value or absent if it is not a timestamped subscription)
   'List of COV Notifications' = (values appropriate to the properties in PROPS)
8. RECEIVE BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = UNKNOWN_SUBSCRIPTION |
   (BACnet-SimpleAck-PDU)

8.X12.2.2 Unknown Subscription

Reason for Change: Added new test to support DS-COVM-A testing.

Purpose: To verify that an appropriate response is returned if a COVM notification arrives that contains arguments or parameters which do not match any current subscriptions.

Test Concept: The TD sends a ConfirmedCOVNotificationMultiple-Request which does not correspond to any existing subscriptions. Verify that the IUT responds with either an error message or a Simple-ACK.

Configuration Requirements: At the start of the test, the IUT shall have no outstanding COVM subscriptions with TD using process identifier ID2.

Test Steps:
1. TRANSMIT ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID2,
   'Initiating Device Identifier' = TD,
   'Time Remaining' = (any valid value),
   'List of COV Notifications' = (any valid list of property notifications)

2. RECEIVE
   BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = (UNKNOWN_SUBSCRIPTION) |
   (BACnet-SimpleACK-PDU)

9.X41 SubscribeCOVPropertyMultiple Service Execution Tests

9.X41.1 Positive SubscribeCOVPropertyMultiple Service Execution Tests

9.X41.1.1 Supports Non-Timestamped Notifications

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT can execute a COVM Notification without providing a timestamp

Test Concept: A subscription for COVM notifications, with the Timestamped parameter set to FALSE. Verify that the IUT sends the appropriate COVM notification in response.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (ID1: any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = L,
   'Max Notification Delay' = (any valid delay between 1 and 3600),
   'List of COV Subscription Specifications' = (any valid list with 'Timestamped' set to FALSE in all entries)
2. RECEIVE BACnet-SimpleACK-PDU
3. IF (the subscription was for confirmed notifications) THEN
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = TD,
   'Time Remaining' = (a value ~= L),
   -- 'Timestamp' = (absent)
   'List of COV Notifications' = (values appropriate to the properties subscribed to)
   TRANSMIT BACnet-SimpleACK-PDU
   ELSE
   WHILE (notifications have not been received for all subscribed to items)
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = TD,
   'Time Remaining' = (a value ~= L),
   -- 'Timestamp' = (absent)
   'List of COV Notifications' = (values appropriate to some or all of the properties subscribed to)

9.X41.1.2 Supports Timestamped Notifications

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT can execute a COVM Notification providing a timestamp

Test Concept: A subscription for COVM notifications with the Timestamped parameter set to TRUE for at least 1 entry in the list of subscriptions, and FALSE for at least 1 entry in the list of subscriptions, is sent to the IUT for properties for which the IUT supports COVM. Verify that the IUT sends the appropriate COVM notification in response.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (ID1: any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (L: a valid lifetime),
   'Max Notification Delay' = (any valid delay between 1 and 3600),
   'List of COV Subscription Specifications' = (any valid list of properties which exist in the IUT for which the IUT supports COVM with Timestamped set to TRUE for at least one, and FALSE for at least one)
2. RECEIVE BACnet-SimpleACK-PDU
3. IF (the subscription was for confirmed notifications) THEN
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (a value ~= L),
   'Timestamp' = (an appropriate timestamp)
   'List of COV Notifications' = (values appropriate to the properties subscribed to along with 'Time of Change' values only for those for which timestamps were requested)
   TRANSMIT BACnet-SimpleACK-PDU
   ELSE
   WHILE (notifications have not been received for all subscribed to items)
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (a value ~= L),
   'Timestamp' = (an appropriate timestamp)
   'List of COV Notifications' = (values appropriate to some or all of the properties subscribed to along with 'Time of Change' values only for those for which timestamps were requested)

Notes to Tester: If the IUT only supports COVM for one property in one object, then the subscription shall be for the single property with Timestamped set to TRUE.

9.X41.1.3 Confirmed Change of Value Notification From Property Value

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT initiates a ConfirmedCOVMultipleNotification service request when a subscribed to property changes.

Test Concept: A COVM subscription is made which contains a subscription to property P1 in object O1. The value of P1 is changed and it is verified that the IUT sends a COVM notification.

Test Steps:
1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (ID1: any valid process identifier),
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = (L: any valid lifetime),
   'Max Notification Delay' = (any valid value),
   'List of COV Subscription Specifications' = (PROPS: a valid list of properties for which the IUT supports COVM including P1 in O1)
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,

'Initiating Device Identifier' = IUT,
'Time Remaining' = (a value ~= L),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent),
'List of COV Notifications' = (values appropriate to the subscribed to properties)

4. TRANSMIT BACnet-SimpleACK-PDU
5. MAKE (a change to P1 that should cause a COVM notification)
6. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (a value greater than 0 and less than L),
   'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent),
   'List of COV Notifications' = (a list consisting of a valid value for P1 and values for any co-reported properties as described in clause 13.1)
7. TRANSMIT BACnet-SimpleACK-PDU

9.X41.1.4 Unconfirmed Change of Value Notification From Property Value

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT initiates an UnconfirmedCOVMultipleNotification service request when a subscribed to property changes.

Test Concept: A COVM subscription is made which contains a subscription to property P1 in object O1. The value of P1 is changed and it is verified that the IUT send a COVM notification.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (ID1: any valid process identifier),
   'Issue Confirmed Notifications' = FALSE,
   'Lifetime' = L,
   'Max Notification Delay' = (MND: any valid value)
   'List of COV Subscription Specifications' = (a valid list of properties for which the IUT supports COVM including P1 in O1)
2. RECEIVE BACnet-SimpleACK-PDU
3. WHILE (notifications have not been received for all subscribed to items)
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (a value ~= L),
   'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent)
   'List of COV Notifications' = (values appropriate to some or all of the properties subscribed to)
4. MAKE (a change to the P1 that should cause a COVM notification)
5. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (a value greater than 0 and less than the requested lifetime),
   'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent),
   'List of COV Notifications' = (a list consisting of a valid value for P1 and values for any co-reported properties as described in clause 13.1)

9.X41.1.5 Supports Subscriptions to Multiple Properties Using Multiple Requests

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify the server adds new subscriptions to existing COVM contexts when requested.

Test Concept: A subscription for COVM notifications is established for property P1 of object O1. A second subscription is sent using the same COVM context for property P2 in object O2. Verify that the IUT's Active_COV_Multiple_Subscriptions property is correctly updated after each subscription.

Configuration Requirements: There are no active COVM subscription for properties in the IUT. If the IUT cannot be configured to have 2 properties which support COVM subscriptions, then this test shall be skipped.

Test Steps:

1. CHECK (the IUT's Active_COV_Multiple_Subscriptions property is empty)
2. TRANSMIT SubscribeCOVPropertyMultiple-Request
   - 'Subscriber Process Identifier' = (ID1: any valid process identifier),
   - 'Issue Confirmed Notifications' = TRUE | FALSE,
   - 'Lifetime' = L,
   - 'Max Notification Delay' = (any valid notification delay),
   - 'List of COV Subscription Specifications' = {
     - ('Monitored Object' = O1,
     - 'List of COV References' = {
       - ('Monitored Property' = P1,
       - 'COV Increment' = (any valid increment, or empty if P1 is not numeric),
       - 'Timestamped' = TRUE | FALSE)}
     -)
   - }
3. RECEIVE BACnet-SimpleAck-PDU
4. IF (confirmed notifications were requested) THEN
   - BEFORE **Notification Fail Time**
     - RECEIVE ConfirmedCOVNotificationMultiple-Request,
       - 'Subscriber Process Identifier' = ID1,
       - 'Initiating Device Identifier' = IUT,
       - 'Time Remaining' = (any value ~= L),
       - 'Timestamp' = (an appropriate timestamp)
       - 'List of COV Notifications' = (a list of values of length 1 indicating P1's value)
   - TRANSMIT BACnet-SimpleACK-PDU

   ELSE
   - BEFORE **Notification Fail Time**
     - RECEIVE UnconfirmedCOVNotificationMultiple-Request,
       - 'Subscriber Process Identifier' = ID1,
       - 'Initiating Device Identifier' = IUT,
       - 'Time Remaining' = (any value ~= L),
       - 'Timestamp' = (an appropriate timestamp, or absent if timestamps not requested)
       - 'List of COV Notifications' = (a list of values of length 1 indicating P1's value)
5. VERIFY Active_COV_Multiple_Subscriptions = (a list with one entry for COVM context ID1 with 1 entry for the subscription to P1)
6. TRANSMIT SubscribeCOVPropertyMultiple-Request
   - 'Subscriber Process Identifier' = ID1,
   - 'Issue Confirmed Notifications' = TRUE | FALSE,
   - 'Lifetime' = L,
   - 'Max Notification Delay' = (any valid notification delay),
   - 'List of COV Subscription Specifications' = {
     - ('Monitored Object' = O2,
     - 'List of COV References' = {
       - ('Monitored Property' = P2,
       - 'COV Increment' = (any valid increment, or empty if P2 is not

numeric),
'Timestamped' = TRUE | FALSE)}
)
}

7. RECEIVE BACnet-SimpleAck-PDU
8. IF confirmed notifications were requested THEN
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (any value ~= L),
   'Timestamp' = (an appropriate timestamp, or absent if timestamps not requested)
   'List of COV Notifications' = (a list of values of length 1 indicating P2's new value)
   TRANSMIT BACnet-SimpleACK-PDU
   ELSE
   WHILE (notifications have not been received for all subscribed to items)
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (any value ~= L),
   'Timestamp' = (an appropriate timestamp, or absent if timestamps not requested)
   'List of COV Notifications' = (a list of values of length 1 indicating P2's new value)
9. VERIFY Active_COV_Multiple_Subscriptions = (a list with one entry for COVM context ID1 with 2 entries for P1 and P2)

Notes to Tester: Objects O1 and O2 can be the same object, and properties P1 and P2 can be the same property, but (O1, P1) must be different than (O2, P2).

9.X41.1.6 Ensuring 5 Concurrent COV-Multiple Contexts With 5 COV-References Per Context

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT can support 5 COV-multiple contexts with 5 COV-references each.

Test Concept: Subscriptions for COVM notifications are made using process identifiers PID1 through PID5. The required post subscription notifications are verified. Once all subscriptions are made, the Active_COV_Multiple_Subscriptions is verified to contain all subscriptions.

Configuration Requirements: The IUT has no active COVM subscriptions.

Test Steps:

1. REPEAT (X=PID1 to PID5) {
   TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = X,
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (L, any value large enough to complete the test),
   'Max Notification Delay' = (any valid value),
   'List of COV Subscription Specifications' = (any valid list of properties for which the IUT supports COVM)
   RECEIVE BACnet-SimpleACK-PDU
2. IF (if confirmed notifications were requested) THEN
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = X,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (a value ~= L),

'Timestamp' = (any appropriate timestamp, if subscribed to timestamped notifications),
'List of COV Notifications' = (values appropriate to the subscribed to properties)
TRANSMIT BACnet-SimpleACK-PDU
ELSE
WHILE (notifications have not been received for all subscribed to items)
BEFORE **Notification Fail Time**
RECEIVE UnconfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = X,
'Initiating Device Identifier' = IUT,
'Time Remaining' = (any valid value),
'Timestamp' = (any appropriate timestamp, if subscribed to timestamped notifications)
'List of COV Notifications' = (values appropriate to some or all of the properties subscribed to)
}
3. VERIFY Active_COV_Multiple_Subscriptions = (a list of 5 COVM contexts as subscribed to)

9.X41.1.7 Supports Client-Supplied COV Increment

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT abides by client supplied COV increments from SubscribeCOVPropertyMultiple requests.

Test Concept: A subscription for COVM notifications is made to a numeric property P1 which supports COVM in object O1. The COV Increment, N, is specified in the subscription request. Verify that the COV Increment N is stored in the COVM context for this subscription. The value of P1 is changed by less than the COV Increment and the TD waits to ensure the IUT does not generate a notification. The value of P1 is changed such that the total change is more than N and it is verified that the IUT sends a notification within the delay time.

Configuration Requirements: If the property being subscribed to has a related COV_Increment property in the object, then the value of N should be significantly different than the value of the COV_Increment property. If the object does not have a COV_Increment property, then N shall be significantly different than the device's internal COV Increment.

Test Steps:
1. TRANSMIT SubscribeCOVPropertyMultiple-Request
'Subscriber Process Identifier' = (ID1: any valid process identifier),
'Issue Confirmed Notifications' = TRUE | FALSE,
'Lifetime' = L,
'Max Notification Delay' = (MND: any valid value),
'List of COV Subscription Specifications' = {('Monitored Object' = O1,
'List of COV References' = { (
'Monitored Property' = P1,
'COV Increment' = N,
'Timestamped' = TRUE | FALSE)}
) }
2. RECEIVE BACnet-SimpleACK-PDU
3. VERIFY Active_COV_Multiple_Subscriptions = (a list containing a COVM context for ID1 containing 1 entry for P1 with a COV_ Increment of N)
4. MAKE (P1's value change by less than COV Increment)
5. WAIT **Notification Fail Time** + MND
5. CHECK (verify that the IUT did not transmit a notification message for the monitored property)
6. MAKE (P1's value change such that the total change to P1 is slightly more than N)
7. IF (the subscription was for confirmed notifications) THEN
BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = ID1,
'Initiating Device Identifier' = IUT,
'Time Remaining' = (any valid value greater than 0 and less than L),

'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent)
'List of COV Notifications' = (a list of values of length 1 indicating P1's new value)
TRANSMIT BACnet-SimpleACK-PDU
ELSE
WHILE (notifications have not been received for all subscribed to items)
BEFORE **Notification Fail Time**
RECEIVE UnconfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = (ID1),
'Initiating Device Identifier' = IUT,
'Time Remaining' = (any valid value greater than 0 and less than L),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent)
'List of COV Notifications' = (a list of values of length 1 indicating P1's new value)

9.X41.1.8 Updating Existing Subscriptions

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT supports resubscriptions to extend the lifetime of COVM contexts.

Test Concept: A COVM subscription is made for 1 or more properties in the IUT. The IUT shall be made to transmit a notification to the TD and the Time Remaining value is validated. Before the subscription expires, the TD resubscribes with a different, and longer, lifetime and the new lifetime is verified in the resultant COVM notification.

Test Steps:
1. TRANSMIT SubscribeCOVPropertyMultiple-Request
'Subscriber Process Identifier' = ID1,
'Issue Confirmed Notifications' = TRUE | FALSE,
'Lifetime' = L,
'Max Notification Delay' = (MND: any valid value)
'List of COV Subscription Specifications' = (PROPS: a valid list of subscriptions)
2. RECEIVE BACnet-SimpleACK-PDU
3. IF (the subscription was for confirmed notifications) THEN
BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = IUD1,
'Initiating Device Identifier' = IUT,
'Time Remaining' = (TR: TR ~= L),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped Notifications, otherwise absent)
'List of COV Notifications' = (values appropriate to each entry in PROPS)
TRANSMIT BACnet-SimpleACK-PDU
ELSE
WHILE (notifications have not been received for all subscribed to items)
BEFORE **Notification Fail Time**
RECEIVE UnconfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = ID1,
'Initiating Device Identifier' = IUT,
'Time Remaining' = (TR:TR ~= L),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent),
'List of COV Notifications' = (values appropriate to some or all entries in PROPS)
4. MAKE (a change to a monitored property, P1, that should cause a COVM notification)
5. WAIT N seconds, where L > N > the resolution of the IUT's COVM lifetime timer
6. IF (the subscription was for confirmed notifications) THEN
BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = ID1,
'Initiating Device Identifier' = IUT,

'Time Remaining' = (TR: 0 < TR < (L - N)),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent)
'List of COV Notifications' = (a list of values of length 1 indicating P1's new value)
TRANSMIT BACnet-SimpleACK-PDU
ELSE
WHILE (notifications have not been received for all subscribed to items)
BEFORE **Notification Fail Time**
RECEIVE UnconfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = ID1,
'Initiating Device Identifier' = IUT,
'Time Remaining' = (TR: 0 < TR < (L - N)),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent),
'List of COV Notifications' = (values appropriate to some or all entries in PROPS)

7. TRANSMIT SubscribeCOVPropertyMultiple-Request
'Subscriber Process Identifier' = ID1,
'Issue Confirmed Notifications' = (the same value used previously),
'Lifetime' = (L2: where L < L2 <= 28800),
'Max Notification Delay' = MND,
'List of COV Subscription Specifications' = PROPS
8. RECEIVE BACnet-SimpleACK-PDU
9. IF (the subscription was for confirmed notifications) THEN
BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = ID1,
'Initiating Device Identifier' = IUT,
'Time Remaining' = (TR2: TR ~= L2),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent)
'List of COV Notifications' = (values appropriate to each entry in PROPS)
TRANSMIT BACnet-SimpleACK-PDU
ELSE
WHILE (notifications have not been received for all subscribed to items)
BEFORE **Notification Fail Time**
RECEIVE UnconfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = (the same identifier used in step 2),
'Initiating Device Identifier' = IUT,
'Time Remaining' = (TR2: TR2 ~= L2),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped notifications, otherwise absent),
'List of COV Notifications' = (values appropriate to some or all entries in PROPS)

9.X41.1.9 Canceling Subsets of COVM Subscriptions

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly cancels COVM subscriptions for some, not all, of the properties subscribed to in a COVM context.

Test Concept: A subscription for COVM notifications is established for multiple properties within the IUT. Before the subscriptions expire, one of the subscriptions is cancelled. Verify that the IUT's Active_COV_Multiple_Subscriptions property only contains an entry for the remaining subscriptions.

Configuration Requirements: There are no active COVM subscription for properties in the IUT. If the IUT cannot be configured to have 2 properties which support COVM subscriptions, then this test shall be skipped.

Test Steps:

1. VERIFY Active_COV_Multiple_Subscriptions = ()
2. TRANSMIT SubscribeCOVPropertyMultiple-Request
'Subscriber Process Identifier' = (any valid process identifier),

'Issue Confirmed Notifications' = FALSE,
'Lifetime' = L,
'Max Notification Delay' = (any valid notification delay),
'List of COV Subscription Specifications' = (a list of 2 or more properties for which the IUT supports COVM)

3. RECEIVE BACnet-SimpleACK-PDU
4. WHILE (notifications have not been received for all subscribed to items)
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (any value ~= L),
   'Timestamp' = (an appropriate timestamp, or absent if not requested)
   'List of COV Notifications' = (values appropriate to some or all of the properties subscribed to)
5. VERIFY Active_COV_Multiple_Subscriptions = (a list with 1 COVM context containing all properties subscribed to)
6. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = ID1,
   'Issue Confirmed Notifications' = FALSE
   'Lifetime' = (absent)
   'Max Notification Delay' = (absent)
   'List of COV Subscription Specifications' = (CANCELLED: a subset of the properties subscribed to)
7. RECEIVE BACnet-SimpleACK-PDU
8. VERIFY Active_COV_Multiple_Subscriptions = (a list with 1 COVM context containing all remaining properties subscribed to, excluding those in CANCELLED)

9.X41.1.10 Canceling Expired or Non-Existing Subscriptions

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify the IUT does not return an error when the client cancels a COVM subscription that doesn't match any of the COV contexts in the IUT's list of active subscriptions.

Test Concept: Send a SubscribeCOVPropertyMultiple request to cancel a subscription for property P1 in object O1, which is not in the list of subscriptions in the IUT's Active_COV_Multiple_Subscriptions property. Verify that the IUT sends a BACnet-SimpleACK-PDU in response.

Configuration Requirements: The IUT is configured with 1 or more COVM subscriptions. One of the subscriptions is using a process identifier ID1 and includes a subscription to property P1 in object O1. Property P2 in object O2 shall not be included in the subscriptions for ID1 (but may in subscriptions using a different process identifier). Where possible P2 in O2 should be a property for which the IUT supports COVM subscriptions.

Test Steps:

1. READ COVM_LIST = Active_COV_Multiple_Subscriptions
2. CHECK (COVM_LIST contains an COVM context with a process identifier of ID1 and includes a subscription to property P1 in object O1)
3. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier which is not ID1),
   'Issue Confirmed Notifications' = (the value matching the entry for ID1),
   -- 'Lifetime' = (absent),
   -- 'Max Notification Delay' = (absent),
   'List of COV Subscription Specifications' = (a list with 1 entry matching the subscription details for P1 in O1)
4. RECEIVE BACnet-SimpleACK-PDU
5. VERIFY Active_COV_Multiple_Subscriptions = COVM_LIST
6. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = ID1,
   'Issue Confirmed Notifications' = (the value matching the entry for ID1),
   -- 'Lifetime' = (absent),
   -- 'Max Notification Delay' = (absent),
   'List of COV Subscription Specifications' = (a list with 1 entry referencing P2 in O2)

7. RECEIVE BACnet-SimpleACK-PDU
8. VERIFY Active_COV_Multiple_Subscriptions = COVM_LIST

9.X41.1.11 Subscription Expiration Test

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT removes subscriptions from the list of active subscriptions once the subscription lifetime has elapsed.

Test Concept: A COVM subscription is made for 1 or more properties in the IUT. One of the subscribed to properties is made to change and it is verified that the IUT transmits a notification to the TD containing an accurate Time Remaining value. The tester then waits for the subscription to expire, it is verified that Active_COV_Multiple_Subscriptions is updated. The property is changed again and it is verified that the IUT does not send a notification.

Configuration Requirements: No existing subscription exists for ID1 for the TD. A value for L is chosen which is long enough to complete the initial test steps, but which is short enough to wait for it to expire.

Test Steps:
1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = ID1,
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = L,
   'Max Notification Delay' = 0,
   'List of COV Subscription Specifications' = (a valid list of subscriptions)
2. RECEIVE BACnet-SimpleACK-PDU
3. IF (the subscription was for confirmed notifications) THEN
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (a value ~= L),
   'Timestamp' = (an appropriate timestamp, if subscribed to timestamped Notifications, otherwise absent),
   'List of COV Notifications' = (values appropriate to the properties subscribed to)
   TRANSMIT BACnet-SimpleACK-PDU
   ELSE
   WHILE (notifications have not been received for all subscribed to items)
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' ~= (a value approximately equal to, but not greater than, the requested subscription lifetime),
   'Timestamp' = (an appropriate timestamp, if subscribed to timestamped Notifications, otherwise absent),
   'List of COV Notifications' = (values appropriate to some or all of the properties subscribed to)
4. MAKE (a change to a monitored property, P1, that should cause a COVM notification)
5. WAIT N seconds, where L > N > the resolution of the IUT's COVM lifetime timer
6. IF (the subscription was for confirmed notifications) THEN
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotificationMultiple-Request,
   'Subscriber Process Identifier' = ID1,
   'Initiating Device Identifier' = IUT,
   'Time Remaining' = (TR: 0 < TR < (L - N)),
   'Timestamp' = (an appropriate timestamp, if subscribed to timestamped Notifications, otherwise absent)
   'List of COV Notifications' = (a list of values of length 1 indicating P1's value)
   TRANSMIT BACnet-SimpleACK-PDU

ELSE
BEFORE **Notification Fail Time**
RECEIVE UnconfirmedCOVNotificationMultiple-Request,
'Subscriber Process Identifier' = ID1,
'Initiating Device Identifier' = IUT,
'Time Remaining' = (TR: 0 < TR < (L - N)),
'Timestamp' = (an appropriate timestamp, if subscribed to timestamped Notifications, otherwise absent)
'List of COV Notifications' = (values appropriate to the properties subscribed to)

7. WAIT L seconds
8. MAKE (a change to a monitored property that would cause a COVM notification if there were an active subscription)
9. CHECK (verify that the IUT did not transmit a COVM notification message for the modified property)
10. VERIFY Active_COV_Multiple_Subscriptions = (a list which does not contain a COVM context for ID1)

9.X41.2 Negative SubscribeCOVPropertyMutliple Service Execution Tests

9.X41.2.1 The Monitored Object Does Not Support COVM Notification

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the monitored object does not support COVM notifications.

Test Concept: A subscription for COVM notifications is made which includes a request for property P1 in object O1. where O1 does not support COVM All requested subscriptions before O1 are selected such that they would succeed if O1 were not in the list. It is verified that the IUT returns the correct error indicating O1 and P1 as the first failed element encountered.

Configuration Requirements: The object, O1, shall not support COVM notification for any of its properties. If the IUT cannot be configured in this manner, then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid lifetime),
   'Max Notification Delay' = (any valid value smaller than the lifetime),
   'List of COV Subscription Specifications' = (a list of subscriptions for properties in O1 and optionally other objects with P1 being the first property requested from O1)
2. RECEIVE BACnet-Error-PDU,
   'First-Failed-Subscription' = {
   'Monitored Object Identifier' = O1,
   'Monitored Property Reference = P1,
   'Error Class' = OBJECT,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
   }

9.X41.2.2 The Monitored Property Does Not Support COVM Notification

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the monitored property does not support COVM notifications.

Test Concept: A subscription for COVM notifications is made which includes a request for property P1 in object O1. where P1 does not support COVM All requested subscriptions before P1 are selected such that they would succeed if P1 were not in the list. It is verified that the IUT returns the correct error indicating O1 and P1 as the first failed element encountered.

Configuration Requirements: The object, O1, shall support COVM notification for any of its properties. If the IUT does not support objects for which COVM is supported for only a subset of the properties, then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid lifetime),
   'Max Notification Delay' = (any valid value smaller than the lifetime),
   'List of COV Subscription Specifications' = (a list of subscriptions for properties in O1 and optionally other objects with P1 being the first property which cannot be subscribed to)
2. RECEIVE BACnet-Error-PDU,
   'First-Failed-Subscription' = {
   'Monitored Object Identifier' = O1,
   'Monitored Property Reference' = P1,
   'Error Class' = PROPERTY,
   'Error Code' = NOT_COV_PROPERTY
   }

9.X41.2.3 Monitored Object Does Not Exist

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the monitored object does not exist.

Test Concept: A subscription for COVM notifications is made which includes a request for property P1 in object O1. where O1 does not exist but would support COVM for P1 if it did. All requested subscriptions before O1 are selected such that they would succeed if O1 were not in the list. It is verified that the IUT returns the correct error indicating O1 and P1 as the first failed element encountered.

Configuration Requirements: The object, O1, shall be of a type for which the IUT supports COVM notifications for property P1. If the IUT cannot be configured in this manner, then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid lifetime),
   'Max Notification Delay' = (any valid value smaller than the lifetime),
   'List of COV Subscription Specifications' = (a list of subscriptions for properties in O1 and optionally other objects with P1 being the first property requested from O1)
2. RECEIVE BACnet-Error-PDU,
   'First-Failed-Subscription' = {
   'Monitored Object Identifier' = O1,
   'Monitored Property Reference' = P1,
   'Error Class' = OBJECT,
   'Error Code' = UNKNOWN_OBJECT
   }

9.X41.2.4 Monitored Property Does Not Exist

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the monitored property does not exist.

Test Concept: A subscription for COVM notifications is made which includes a request for property P1 in object O1. where P1 does not exist in O1. All requested subscriptions before P1 in O1 are selected such that they would succeed if P1 in O1 were not in the list. It is verified that the IUT returns the correct error indicating O1 and P1 as the first failed element encountered.

Configuration Requirements: The object, O1, shall exist, shall not contain P1 and be of a type for which the IUT supports COVM notifications. If the IUT cannot be configured in this manner, then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid lifetime),
   'Max Notification Delay' = (any valid value smaller than the lifetime),
   'List of COV Subscription Specifications' = (a list of subscriptions for properties in O1 and optionally other objects with P1 being the first property which cannot be subscribed to)
2. RECEIVE BACnet-Error-PDU,
   'First-Failed-Subscription' = {
   'Monitored Object Identifier' = O1,
   'Monitored Property Reference' = P1,
   'Error Class' = PROPERTY,
   'Error Code' = UNKNOWN_PROPERTY
   }

9.X41.2.5 Array Index Provided But Property is Not an Array

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the monitored property is not an array but an array index is provided.

Test Concept: A subscription for COVM notifications is made which includes a request for property P1, with an array index, in object O1 where the IUT supports COVM for P1 in O1 but P1 is not an array. All requested subscriptions before P1 in O1 are selected such that they would succeed if P1 in O1 were not in the list. It is verified that the IUT returns the correct error indicating O1 and P1 as the first failed element encountered.

Configuration Requirements: The property P1 shall be one which supports COVM and is not an array. If the IUT cannot be configured in this manner, then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid lifetime),
   'Max Notification Delay' = (any valid value smaller than the lifetime),
   'List of COV Subscription Specifications' = (a list of subscriptions for properties in O1 and optionally other objects with P1 being the first property which cannot be subscribed to. An array index shall be included in the entry for P1)
2. RECEIVE BACnet-Error-PDU,
   'First-Failed-Subscription' = {
   'Monitored Object Identifier' = O1,
   'Monitored Property Reference' = P1,
   'Error Class' = PROPERTY,
   'Error Code' = PROPERTY_IS_NOT_AN_ARRAY

}

9.X41.2.6 Array Index Provided Is Out Of Range

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the monitored property an array but the provided array index is outside the range of the array.

Test Concept: A subscription for COVM notifications is made which includes a request for an array property P1, with an array index, in object O1 where the IUT supports COVM for P1. All requested subscriptions before P1 in O1 are selected such that they would succeed if P1 in O1 were not in the list. It is verified that the IUT returns the correct error indicating O1 and P1 as the first failed element encountered.

Configuration Requirements: If the IUT does not support COVM on any array properties, then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid lifetime),
   'Max Notification Delay' = (any valid value smaller than the lifetime),
   'List of COV Subscription Specifications' = (a list of subscriptions for properties in O1 and optionally other objects with P1 being the first property which cannot be subscribed to. The array index included in the entry for P1 shall be larger than the number of entries in P1)
2. RECEIVE BACnet-Error-PDU,
   'First-Failed-Subscription' = {
   'Monitored Object Identifier' = O1,
   'Monitored Property Reference' = P1,
   'Error Class' = PROPERTY,
   'Error Code' = INVALID_ARRAY_INDEX
   }

9.X41.2.7 No Space to Add List Element

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when there is no space for a subscription.

Test Concept: Repeatedly subscribe to the same object each time with a different Process Identifier until the device runs out of resources and returns the appropriate error.

Configuration Requirements: If the device cannot be configured such that the maximum number of subscriptions the IUT can accept is less than 10000, then this test shall be skipped.

Test Steps:

REPEAT PID = (1 through the maximum number of subscriptions the IUT can accept plus 1 or until the IUT returns an Error-PDU) {

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = PID,
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid lifetime large enough to complete the test),
   'Max Notification Delay' = (any valid value smaller than the lifetime),

'List of COV Subscription Specifications' = (any valid list of subscriptions)
2. RECEIVE BACNET-SimpleACK-PDU
| (BACnet-Error-PDU,
'Error Class' = RESOURCES,
'Error Code' = NO_SPACE_TO_ADD_LIST_ELEMENT)
}

9.X41.2.8 The Lifetime Parameter is Out Of Range

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the Lifetime parameter is out of range.

Configuration Requirements: If the device supports lifetimes across the full range of valid lifetimes then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
'Subscriber Process Identifier' = (any valid process identifier),
'Issue Confirmed Notifications' = TRUE | FALSE,
'Lifetime' = (a value larger than that supported by the IUT
'Max Notification Delay' = (any valid value smaller than the lifetime),
'List of COV Subscription Specifications' = (any valid list of subscriptions)
2. RECEIVE BACnet-Error-PDU,
'Error Class' = SERVICES,
'Error Code' = VALUE_OUT_OF_RANGE

9.X41.2.9 The Max Notification Delay Parameter is Out Of Range

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the Max Notification Delay parameter is out of range.

Configuration Requirements: If the device supports Max Notification Delays across the full range of valid values then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
'Subscriber Process Identifier' = (any valid process identifier),
'Issue Confirmed Notifications' = TRUE | FALSE,
'Lifetime' = (any valid value large enough to complete the test),
'Max Notification Delay' = (a value larger than supported by the IUT),
'List of COV Subscription Specifications' = (any valid list of subscriptions)
2. RECEIVE BACnet-Error-PDU,
'Error Class' = SERVICES,
'Error Code' = VALUE_OUT_OF_RANGE

9.X41.2.10 The Max Notification Delay is Greater Than the Lifetime

Reason for Change: Added new test to support DS-COVM-B testing.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the Max Notification Delay parameter is greater than the Lifetime parameter.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (a value supported by the IUT but within the normal range of Max Notification Delay)
   'Max Notification Delay' = (a value greater than the lifetime),
   'List of COV Subscription Specifications' = (any valid list of subscriptions)
2. RECEIVE BACnet-Error-PDU,
   'Error Class' = SERVICES,
   'Error Code' = VALUE_OUT_OF_RANGE