BACnet® TESTING LABORATORIES
ADDENDA

Addendum misc1 to
BTL Test Package 16.1

Revision 4
Revised 6/18/2020

Approved by the BTL Working Group on April 16, 2020.
Approved by the BTL Working Group Voting Members on August 13, 2020.
Published on August 14, 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-16.1misc1-1: Remove Allowance to Skip 7.3.1.X4 - BTLWG-546

Overview:

The test description allows for the test to be skipped when there is no need for this allowance.
The test description includes requirements on Event_Enable that are unnecessary.

Changes:

[From 135.2013q, modify test 7.3.1.X4 and place into BTL Specified Tests]

7.3.1.X4 Event_Message_Texts Tests
Reason For Change:Removed the allowance for the test to be skipped.

Purpose: To verify that the value of the Event_Message_Texts property is updated when an object generates an event notification.

Test Concept: Read the Event_Message_Texts from the object. Transition the object through each event state which is enabled in the object saving the Message Text parameter from the received notification. Verify that the Event_Message_Texts updates with the Event_Message_Texts value received from the notification.

Configuration Requirements: The IUT shall be configured with an event-generation object, O1 which shall be in a NORMAL Event_State at the beginning of the test *and if Event_Enable is configurable it shall have all bits set to TRUE for which the object supports transitions*. ~~If the algorithm of the object does not support NORMAL to NORMAL transitions, then the TO-OFFNORMAL bit of the Event_Enable shall be TRUE. If the IUT does not contain any objects which can transition to any offnormal state, then this test shall be skipped.~~

Test Steps:

1. READ EMT = Event_Message_Texts
2. IF (Event_Enable is (TRUE, ?, ?) *and O1 can generate TO_OFFNORMAL transitions*) THEN {
3. IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value that is offnormal)
   ELSE
   MAKE (pMonitoredValue a value that is offnormal)
4. WAIT (pTimeDelay)
5. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid timestamp),
   'Notification Class' = (the class corresponding to the object being tested),
   'Priority' = (the configured TO_OFFNORMAL priority),
   'Event Type' = (any valid event type),
   'Message Text' = (M: any valid value placed into EMT[1]),
   'Notify Type' = Notify_Type,
   'AckRequired' = (the configured value for the TO_OFFNORMAL transition),
   'From State' = NORMAL,
   'To State' = (any valid offnormal state),
   'Event Values' = (values appropriate to the event type)
6. VERIFY Event_Message_Texts = EMT
   }
7. IF (Event_Enable is (?, ?, TRUE) *and O1 can generate TO_NORMAL transitions*) THEN {
8. IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value that will result in a TO_NORMAL transition)
   ELSE
   MAKE (pMonitoredValue a value that will result in a TO_NORMAL transition)

9. WAIT (pTimeDelayNormal)
10. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = O1,
    'Time Stamp' = (any valid timestamp),
    'Notification Class' = (the class corresponding to the object being tested),
    'Priority' = (the configured TO_NORMAL priority),
    'Event Type' = (any valid event type),
    'Message Text' = (M: any valid value placed into EMT[3]),
    'Notify Type' = Notify_Type,
    'AckRequired' = (the configured value for the TO_NORMAL transition),
    'From State' = (any valid value),
    'To State' = NORMAL,
    'Event Values' = (values appropriate to the event type)
11. VERIFY Event_Message_Texts = EMT
    }
12. IF (Event_Enable is (?, TRUE, ?) *and O1 can generate TO_FAULT transitions*) THEN {
13. MAKE (a condition exist that will cause O1 to generate a TO-FAULT transition)
14. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = O1,
    'Time Stamp' = (any valid timestamp),
    'Notification Class' = (the class corresponding to the object being tested),
    'Priority' = (the configured TO_FAULT priority),
    'Event Type' = (IF (Protocol_Revision < 13) THEN
    (any valid event type),
    ELSE
    CHANGE_OF_RELIABILITY),
    'Message Text' = (M: any valid value placed into EMT[2]),
    'Notify Type' = Notify_Type,
    'AckRequired' = (the configured value for the TO_FAULT transition),
    'From State' = (any valid value),
    'To State' = FAULT,
    'Event Values' = (values appropriate to the event type)
15. VERIFY Event_Message_Texts = EMT
    }

BTL-16.1misc1-2: Unicast I-Am Test Fix - BTLWG-592

Overview:

The BTL Test Plan incorrectly references the version of 9.33.2.3 in 135.1 instead of the modified version which exists in BTL Specified Tests.

Changes:

[In BTL Test Plan, in section 8.2.1, modify test reference to 9.33.2.3 to be from BTL Specified Tests instead of 135.1-2013]

~~135.1-2013~~ *BTL* - 9.33.2.3 - General Inquiry, Directed to a Remote Device		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	.

BTL-16.1misc1-3: Correcting DS-V-A and DS-M-A Test Sections - BTLWG-640

Overview:

PR10 added some primitive value objects (among other objects) to Tables K-1 and K-5 in the BACnet standard. The sections for DS-V-A and DS-M-A in the BTL Test Plan have empty subsections to "Support Primitive Value Objects." This work item corrects this problem.

Changes:

[Amend the BTL Checklist to remove references to testing Primitive Value Objects]

Data Sharing - View - A		
	R	Base Requirements
	R	Supports DS-RP-A
	~~BTL-R[1]~~	~~Supports Primitive Value Objects~~
	~~[1]Required if the device implements protocol revision 10 or higher~~	

Data Sharing - Modify - A		
	R	Base Requirements
	R	Supports DS-WP-A
	~~BTL-R[1]~~	~~Supports Primitive Value Objects~~
	~~[1]Required if the device implements protocol revision 10 or higher~~	

[Remove section 4.11.3 of the BTL Test Plan entirely]

4.11 Data Sharing - View - A

4.11.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for <u>each</u> of the standard object types and associated properties specified by DS-V-A. Note: If the IUT also supports DS-AV-A, this test should be omitted because it is a subset of the DS-V-A test.

4.11.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

~~4.11.3 Supports Primitive Value Objects~~

~~Presentation supports all Object Types defined at Protocol_Revision less than or equal to itself.~~

[Remove section 4.13.3 of the BTL Test Plan entirely]

4.13 Data Sharing - Modify - A

4.13.1 Base Requirements

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

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for each of the required object types listed in the table in the BIBB definition. Repeat for each of the required properties listed in the table in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition. Note: If the IUT also supports DS-AM-A, this test may be omitted because it is a subset of the DS-AM-A test.
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	This test should be executed at priority 8 only, i.e. $PR_1 = 8$. Note: If the IUT also supports DS-AM-A, this test may be omitted because it is a subset of the DS-AM-A test.

4.13.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

~~4.13.3 Supports Primitive Value Objects~~

~~Modification is supported for all Object Types defined at Protocol_Revision less than or equal to itself.~~

BTL-16.1misc1-4: Add Testing of DS-COV-B for Access Point Objects - BTLWG-702

Overview:

Access Point and Credential Data Input object types each support COV reporting, and which each specify parameters in the COVNotification that are unique to the object type. This proposal specifies an approach for the support of testing those which is similar as is used in the Loop object type, which similarly has parameters in the COVNotification that are unique to the object type.

Changes:

[In BTL Checklist, completely replace the following sections.]

4.9.21 Can Subscribe for COV from Access Point Objects

The IUT can subscribe for, receive, and process Change of Value notifications from Access Point objects.

BTL - 9.2.1.X6 - ConfirmedCOVNotification from Access Point		
	Test Conditionality	Either 9.2.1.X6 or 9.3.1.X7 must be executed
	Test Directives	Test at least one instance where the object type is an Access Point.
	Testing Hints	
BTL - 9.3.1.X7 - UnconfirmedCOVNotification from Access Point		
	Test Conditionality	Either 9.2.1.X6 or 9.3.1.X7 must be executed
	Test Directives	Test at least one instance where the object type is an Access Point.
	Testing Hints	

4.9.22 Can Subscribe for COV from Credential Data Input Objects

The IUT can subscribe for, receive, and process Change of Value notifications from Credential Data Input objects.

BTL - 9.2.1.X7 - ConfirmedCOVNotification from Credential Data Input		
	Test Conditionality	Either 9.2.1.X7 or 9.3.1.X8 must be executed
	Test Directives	Test at least one instance where the object type is a Credential Data Input.
	Testing Hints	
BTL - 9.3.1.X8 - UnconfirmedCOVNotification from Credential Data Input		
	Test Conditionality	Either 9.2.1.X7 or 9.3.1.X8 must be executed
	Test Directives	Test at least one instance where the object type is a Credential Data Input.
	Testing Hints	

[In BTL Test Plan, add two sections in 4.10 for the Access Point, and Credential Data Input object types.

4.10.30 Supports COV for Access Point Objects

The IUT supports change of value notifications for at least one object of type Access Point.

BTL - 8.2.X12 – Change of Value Notification from an Access Point object		
	Test Conditionality	This may be skipped if **BTL - 8.3.X15** is executed against an Access Point object.
	Test Directives	Test at least one instance where the object type is an Access Point object.
	Testing Hints	
BTL - 8.3.X15 - Change of Value Notification from an Access Point object		
	Test Conditionality	This may be skipped if **BTL - 8.2.X12** is executed against an Access Point object.

	Test Directives	Test at least one instance where the object type is an Access Point object.
	Testing Hints	

4.10.31 Supports COV for Credential Data Input Objects

The IUT supports change of value notifications for at least one object of type Credential Data Input.

BTL - 8.2.X13 – Change of Value Notification from a Credential Data Input Object		
	Test Conditionality	This may be skipped if **BTL - 8.3.X16** is executed against a Credential Data Input object.
	Test Directives	Test at least one instance where the object type is a Credential Data Input object.
	Testing Hints	
BTL - 8.3.X16 - Change of Value Notification from a Credential Data Input Object		
	Test Conditionality	This may be skipped if **BTL - 8.2.X13** is executed against a Credential Data Input object.
	Test Directives	Test at least one instance where the object type is a Credential Data Input object.
	Testing Hints	

[In BTL Specified Tests, add four new tests in existing section 8.2,9.2 for the Access Point, and Credential Data Input. Also add four tests into existing section 8.3,9.3 for the UnconfirmedCOVNotification versions of those.]

8.2 ConfirmedCOVNotification Service Initiation Tests

8.2.X12 Change of Value Notification from an Access Point object

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags and Access_Event_Time properties of Access Point objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Access_Event_Time and Status_Flags of the monitored object is changed, and a notification shall be received. The properties may be changed using the WriteProperty service or by another means. For some implementations it may be necessary to write to the Out_Of_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable. For implementations where it is not possible to write Out_Of_Service or change the Status_Flags by any other means, step 5,6,7 shall be skipped.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PI: any value > 0 chosen by the TD),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,

'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (the initial Access_Event, Status_Flags, Access_Event_Tag, Access_Event_Time, Access_Event_Credential and Access_Event_Authentication_Factor if X has an Access_Event_Authentication_Factor property)

4. TRANSMIT BACnet-SimpleACK-PDU
5. IF (Out_Of_Service is writable) THEN
   WRITE X,Out_Of_Service = TRUE
   ELSE
   MAKE (Status_Flags = any value that differs from initial Status_Flags)
6. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the initial Access_Event, new Status_Flags, initial Access_Event_Tag, Access_Event_Time, Access_Event_Credential and Access_Event_Authentication_Factor if X has a Access_Event_Authentication_Factor property)
7. TRANSMIT BACnet-SimpleACK-PDU
8. IF (Access_Event_Time is now writable) THEN
   WRITE Access_Event_Time = (any value that differs from initial Access_Event_Time)
   ELSE
   MAKE (Access_Event_Time = any value that differs from initial Access_Event_Time)
9. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the new values of Access_Event, Access_Event_Tag, Access_Event_Time, Access_Event_Credential, and Access_Event_Authentication_Factor if X has Access_Event_Authentication_Factor property)
10. TRANSMIT BACnet-SimpleACK-PDU
11. TRANSMIT SubscribeCOV-Request,
    'Subscriber Process Identifier' = PI,
    'Monitored Object Identifier' = X
12. RECEIVE BACnet-SimpleACK-PDU
13. IF (Out_Of_Service is writable) THEN
    WRITE Out_Of_Service = FALSE
14. CHECK (verify that no notification message has been transmitted)

8.2.X13 Change of Value Notification from an Credential Data Input object

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags and Update_Time properties of Credential Data Input objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Status_Flags and Update_Time properties of the monitored object is changed, and a notification shall be received. The properties may be changed using the WriteProperty service or by another means. For some implementations it may be necessary to write to the Out_Of_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable. For implementations where it is not possible to write Out_Of_Service or change the Status_Flags by any other means, step 5,6,7 shall be skipped

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PI: any value > 0chosen by the TD),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the initial Present_Value, initial Status_Flags, and Update_Time (most recent update time when the Present_Value was updated))
4. TRANSMIT BACnet-SimpleACK-PDU
5. IF (Out_Of_Service is writable) THEN
   WRITE X, Out_Of_Service = TRUE
   ELSE
   MAKE (Status_Flags = any value that differs from initial Status_Flags)

6. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the initial Present_Value, new Status_Flags, and Update_Time (most recent update time when the Present_Value was updated))
7. TRANSMIT BACnet-SimpleACK-PDU
8. IF (Present_Value is now writable) THEN
   WRITE X, Present_Value = (any value that differs from initial Present_Value)
   ELSE
   MAKE (Present_Value = any value that differs from initial Present_Value)
9. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the new Present_Value, new Status_Flags, and Update_Time (most recent update time when the Present_Value was updated))

10. TRANSMIT BACnet-SimpleACK-PDU
11. Verify Update_Time received in step 7.
12. TRANSMIT SubscribeCOV-Request,
    'Subscriber Process Identifier' = PI,
    'Monitored Object Identifier' = X
13. RECEIVE BACnet-SimpleACK-PDU
14. IF (Out_Of_Service is writable) THEN
    WRITE Out_Of_Service = FALSE
15. CHECK (verify that no notification message has been transmitted)

8.3 UnconfirmedCOVNotification Service Initiation Tests

8.3.X15 Change of Value Notification from an Access Point object

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Status_Flag and Access_Event_Time properties of Access Point objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.X12 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services.

8.3.X16 Change of Value Notification from an Credential Data Input object

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Status_Flags and Update_Time properties of Credential Data Input objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.X13 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services.

9.2 ConfirmedCOVNotification Service Execution Tests

9.2.1 Positive ConfirmedCOVNotification Service Execution Tests

9.2.1.X6 ConfirmedCOVNotification from Access Point Object

Purpose: To verify that the IUT can execute ConfirmedCOVNotification requests from Access Point objects.

Test Steps:

1. RECEIVE SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PI: any valid process identifier value > 0),
   'Monitored Object Identifier' = (X: any Access Point object),
   'Issue Confirmed Notifications ' = TRUE,
   'Lifetime' = (a value greater than one minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining in the subscription),
   'List of Values' = (any valid set of values)
4. RECEIVE BACnet-SimpleACK-PDU
5. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

9.2.1.X7 ConfirmedCOVNotification from Credential Data Input Object

Purpose: To verify that the IUT can execute ConfirmedCOVNotification requests from Credential Data Input objects.

Test Steps:

1. RECEIVE SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PI: any valid process identifier value > 0),
   'Monitored Object Identifier' = (X: any Credential Data Input object),
   'Issue Confirmed Notifications ' = TRUE,
   'Lifetime' = (a value greater than one minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT ConfirmedCOVNotification-Request,

'Subscriber Process Identifier' = PI,
'Initiating Device Identifier' = TD,
'Monitored Object Identifier' = X,
'Time Remaining' = (the time remaining in the subscription),
'List of Values' = (any valid set of values)

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

9.3 UnconfirmedCOVNotification Service Execution Tests

9.3.1.X7 UnconfirmedCOVNotification from Access Point Object

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

Test Steps:

1. RECEIVE SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PI: any valid process identifier value > 0),
   'Monitored Object Identifier' = (X: any Access Door object),
   'Issue Confirmed Notifications ' = FALSE,
   'Lifetime' = (a value greater than one minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining in the subscription),
   'List of Values' = (any valid set of values)
4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

9.3.1.X8 UnconfirmedCOVNotification from Credential Data Input Object

Purpose: To verify that the IUT can execute UnconfirmedCOVNotification requests from Credential Data Input objects.

Test Steps:

1. RECEIVE SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PI: any valid process identifier value > 0),
   'Monitored Object Identifier' = (X: any Access Door object),
   'Issue Confirmed Notifications ' = FALSE,
   'Lifetime' = (a value greater than one minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining in the subscription),
   'List of Values' = (any valid set of values)
4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

BTL-16.1misc1-5: Unconfirmed COV Notification Test Changes - BTLWG-704

Overview:

SSPC has clarified that all COVNotifications which arise from SubscribeCOV and SubscribeCOVProperty are sent to the subscriber, rather than broadcast so the last sentence of 9.11.1.X12 should be removed.

Changes:

[Change 9.11.1.X12 in BTL Specified Tests]
[The last sentence of the Test Steps section, “The MAC address used for the notification message shall be such that the TD is one of the recipients” has been removed]

9.11.1.X12 Unconfirmed Change of Value Notification from Property Value

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Property Value.

Test Steps: The steps for this test case are identical to the test steps in 9.11.1.X11 except that the SubscribeCOVProperty service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services.

BTL-16.1misc1-6: A Side Slave Proxy Tests - BTLWG-710

Overview:

This proposal contains coverage for the Slave Proxy A side.

Changes:
[Replace Test Plan entry 8.29]

8.29 Device Management - Slave Proxy - View and Modify - A

8.29.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for each of the Slave Proxy properties in the Device object, and in the Network Port object, if the claimed Protocol_Revision is 17 or greater.
	Testing Hints	
BTL - 8.18.X1 - Reading and Presenting Large List Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for each of {Manual_Slave_Address_Binding, Slave_Address_Binding} properties in the Device object, and in the Network Port object, if the claimed Protocol_Revision is 17 or greater.
	Testing Hints	
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for each of the Slave Proxy properties in the Device object, and in the Network Port object, if the claimed Protocol_Revision is 17 or greater.
	Testing Hints	
BTL - 8.22.X1 - Accepting Input and Modifying Large List Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for each of {Manual_Slave_Address_Binding, Slave_Address_Binding} properties in the Device object, and in the Network Port object, if the claimed Protocol_Revision is 17 or greater.
	Testing Hints	

[Add clause 8.18.X1 into BTL Specified Tests]

8.18.X1 Reading and Presenting Large List Properties

Reason for Change: there is no appropriate test for reading and presenting large list property values as required by DM-SP-VM-A.

Purpose: This test case verifies that the IUT is capable of reading and presenting large list properties using ReadRange. It is a generic test used to test data presentation requirements.

Configuration: For this test, the tester shall choose a list property, P1, from an object, O1. The TD shall be configured to not support segmentation. The value is P1 shall be too large to read via ReadProperty or ReadPropertyMultiple.

Test Steps:

1. MAKE (the IUT read P1)
2. WHILE (the complete list has not been read)
   RECEIVE ReadRange-Request,
   'Object Identifier' = O1,

'Property Identifier' = P1,
'Range' = (any valid value for P1)
TRANSMIT BACnet-ComplexACK-PDU,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Result Flags' = (values consistent with the request),
'Item Count' = (values consistent with the request),
'Item Data' = (values consistent with the request)

3. CHECK (that the IUT presents a list of values that is consistent with the values received in step 2)

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

Notes to Tester: If the IUT has not already determined that the value cannot be read using ReadProperty or ReadPropertyMultiple, the IUT may initiate a ReadProperty or ReadPropertyMultiple. If this occurs, the IUT shall pass the test only if it automatically falls back to using ReadRange upon receipt of the correct BACnetReject-PDU from the TD, indicating that the response is too large.

[Add clause 8.22.X1 into BTL Specified Tests]

8.22.X1 Accepting Input and Modifying Large List Properties

Reason for Change: there is no appropriate test for modifying large list property values as required by DM-SP-VM-A.

Purpose: This test case verifies that the IUT is capable of accepting user input and using it to modify large list properties where AddListElement and RemoveListElement will be required. It is a generic test used to test data-input requirements.

Configuration: For this test, the tester shall choose a list property, P1, from an object, O1. The TD shall be configured to not support segmentation. The list property shall be configured with a value such that it cannot be read or written without the use of ReadRange and AddListElement/RemoveListElement.

Test Steps:

-- test adding elements into the list

1. MAKE (the IUT accept 1 or more new entries, {E1..En} for P1 from the user)
2. RECEIVE AddListElement-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1
   'List Of Elements' = (E1, .., En)
3. TRANSMIT BACnet-SimpleACK-PDU

-- test removing elements from the list

4. MAKE (the IUT delete 1 or more entries, {E1..Ex} for P1 from the user)
5. RECEIVE RemoveListElement-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1
   'List Of Elements' = (E1, .., Ex)
6. TRANSMIT BACnet-SimpleACK-PDU

Notes to Tester: The value accepted by the IUT may differ from the value transmitted on the wire due to rounding, truncation, formatting, language conversion, etc.

Notes to Tester: If the IUT has not already determined that the value cannot be transmitted using WriteProperty or WritePropertyMultiple, the IUT may initiate a WriteProperty or WritePropertyMultiple. If this occurs, the IUT shall pass the test only if it automatically falls back to using AddListElement and RemoveListElement upon receipt of the correct BACnetReject-PDU from the TD, indicating that the write request is too large.

BTL-16.1misc1-7: New EPICS Consistency Test for Read-Only Properties - BTLWG-746

Overview:

If a property is documented in the standard as read-only there is currently no testing in the BTL test package to verify that the property is not writable.

Changes:

[In BTL Specified Tests, append to 5. EPICS CONSISTENCY TESTS]

(p) For each property that is required to be read-only, that property shall not be marked as writable, or conditionality writable, in the EPICS.

BTL-16.1misc1-8: Multi-State Range Test - BTLWG-810

Overview:

There is currently no range testing of Multi-State Present_Value in the Test Package even though there does exist a test in 135.1.

Changes:

[In BTL Test Plan - Add entry into 3.14.1, and 3.16.1 which are the Base Requirements in Multi-State Input, and Multi-State Value sections]

135.1-2013 - 7.3.1.15 Number_Of_States Range Test		
	Test Conditionality	If the IUT does not contain an object with a writable, or conditionally writable Present_Value, this test shall be skipped.
	Test Directives	
	Testing Hints	

[In BTL Test Plan - Add entry into 3.15.1 which is the Base Requirements in the Multi-State Output section]

135.1-2013 - 7.3.1.15 Number_Of_States Range Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

BTL-16.1misc1-9: Clarify Calendar Entry WeekNDay Day Of Week Test - BTLWG-840 [CR-406]

Overview:

CR-0406 noted issues with wording of this test in the response. The issues are:

- **D_1 and D_2 / V_1 and V_2 are not clearly unique values (although one could argue the dates are unique to each other per the table)**
- **Checking that the value is anything but V_2 is not helpful, especially if there are other Time-Value pairs with the same value which may become active as a result of step 5.**

BTLWG-200 has some fix for this and is already included in Test Plan 16.1. It has the original fix I proposed, which is to make V_2 a unique value. This is not always possible, for example when the Present_Value is a binary datatype. Since V_2 is at a different Date than V_1, it only needs to be different than the expected value at the selected Date for that part of the test.

[In BTL Specified Tests, update test 7.3.2.23.10.3.7, containing a fix for the CR]

7.3.2.23.10.3.7 Revision 4 Calendar Entry WeekNDay Day Of Week Test

Reason for Change: Added clarifying text to table 7-16.1.

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

BACnet Reference Clause: 12.24.8.

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

Test Concept: The IUT's local date and time are changed to values that are selected by the TD based on the criteria in Table 7-10. The value of the Present_Value property is monitored to verify that the scheduled write operations occur.

Configuration Requirements: The IUT shall be configured to contain a Schedule object with an Exception_Schedule containing a BACnetCalendarEntry with a WeekNDay entry specifying the day of the week. The criteria for the dates used in the test are given in Table 7-10. The local date and time shall be set such that the Present_Value property has a value other than V_1.

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

Date	Criteria	Value
D_1	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies only DayOfWeek, 2C. Date falls on the specified day of the week, and 2D. Higher eventPriority than any coincident BACnetSpecialEvents.	V_1
D_2	1. Date occurs during Effective_Period, 2A. BACnetSpecialEvent incorporates calendarEntry: WeekNDay, 2B. calendarEntry: WeekNDay specifies only DayOfWeek, and 2C. Date does not fall on the specified day of the week.	V_2 (*a value* different from ~~any other values~~ *the Present_Value expected at D_2*)

Test Steps:

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

BTL-16.1misc1-10: Improve Stop_When_Full Test - BTLWG-854

Overview:

Modified existing test case to check IUT will return negative response when TRUE is written to Enable and the Log_Buffer is full.

Changes:

[In BTL Specified Tests, modify test 7.3.2.24.6.1 with the below changes]

7.3.2.24.6.1 Stop_When_Full TRUE Test

Reason For Change: This test was revised for negative response.

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

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

Test Concept: The logging object is configured to acquire data by whatever means. Data is collected until more than Buffer_Size records have been collected and Enable is verified to be FALSE.
Attempt to write TRUE to Enable and verify that the IUT does not accept it due to Log_Buffer being full.

Configuration Requirements: *The IUT shall be configured with Object1 where* Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured with the latest possible date and time, in order that it occur after the end of the test. Stop_When_Full, if configurable, shall be set to TRUE. Enable shall be set to FALSE.

Test Steps:
1. **WRITE Record_Count = 0**
2. **WRITE Enable = TRUE**
3. **WHILE (Record_Count < Buffer_Size) DO {}**
4. **WAIT** Internal Processing Fail Time
5. **VERIFY Enable = FALSE**
6. ***TRANSMIT WriteProperty-Request,***
   'Object Identifier' = Object1,
   'Property Identifier' = Enable,
   'Property Value' = TRUE
7. ***RECEIVE BACnet-Error-PDU,***
   Error Class = OBJECT,
   Error Code = LOG_BUFFER_FULL.
8. ***VERIFY Enable = FALSE***