# BACnet® TESTING LABORATORIES
# ADDENDA

# Addendum bi to
# BTL Test Package 18.1

**Revision 3**
**Revised 11/17/2021**

Approved by the BTL Working Group on October 7, 2021
Approved by the BTL Working Group Voting Members on November 10, 2021
Published on November 19, 2021

**[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.

BTL-18.1-bi-1: Change DeviceCommunicationsControl for Audit Reporting [BTLWG-676, CR-0492]..... 2

BTL-18.1-bi-2: Add Audit Log and Audit Reporter Testing [BTLWG-420, BTLWG-422] ..... 7

In the following document, language to be added to existing clauses within the BTL Test Package 18.1 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-18.1-bi-1: Change DeviceCommunicationsControl for Audit Reporting [BTLWG-676, CR-0492]**

**Overview:**
Protocol Revision (PR) 20, the DISABLE parameter option in Device Communication Control service is deprecated. Based on CR0492 and the above addendum, some of the test steps are modified accordingly.

**Changes:**

# BTL Checklist Changes

**No changes**

# BTL Test Plan Changes

[In BTL Test Plan, Add new tests]

### 8.14.1 Base Requirements

~~There are no base requirements for this BIBB.~~

[Add new Test BTL-9.24.1.11 under section 8.14.1 in the Test Plan]

| | | |
|---|---|---|
| ***BTL-9.24.1.11 - Ensure that DISABLE option is not supported by IUT claiming PR >= 20*** | | |
| | ***Test Conditionality*** | *If the IUT claims Protocol Revision < 20, this test shall be skipped.* |
| | ***Test Directives*** | *If the IUT does not support an internal clock this test shall be tested with indefinite time duration.* |
| | ***Testing Hints*** | |

[Move test reference 135.1-2019 – 9.24.2.1 to BTL-9.24.2.1 under section 8.14.2 in the Test plan ]
[Move test reference 135.1-2019 – 9.24.2.2 to BTL-9.24.2.1 under section 8.14.2 in the Test plan]

### 8.14.2 Supports Receiving a DeviceCommunicationControl Service Request with a Password

| | | |
|---|---|---|
| ~~***135.1-2019 - 9.24.2.1 - Invalid Password***~~ ***BTL-9.24.2.1 - Invalid Password*** | | |
| | **Test Conditionality** | Must be executed |
| | **Test Directives** | |
| | **Testing Hints** | |
| ~~***135.1-2019 - 9.24.2.2 - Invalid Password***~~ ***BTL-9.24.2.2 - Missing Password*** | | |
| | **Test Conditionality** | Must be executed |
| | **Test Directives** | |
| | **Testing Hints** | |

[Remove test 135.1-2019 - 9.24.1.1 under 8.14.3 in the Test Plan]
[Remove test 135.1-2019 - 9.24.1.3 under 8.14.3 in the Test Plan]
[Add test 135.1-2019 - 9.24.1.6 under 8.14.3 in the Test Plan]
[Add test 135.1-2019 - 9.24.1.8 under 8.14.3 in the Test Plan]

### 8.14.3 Supports Receiving a DeviceCommunicationControl Service Request with no Password

The IUT does not require, or can be made to not require, a password parameter in a DeviceCommunicationControl service request.

| | | |
|---|---|---|
| ~~**135.1-2019 - 9.24.1.3 - Finite Time Duration**~~ | | |
| | ~~**Test Conditionality**~~ | ~~If the IUT does not support an internal clock this test may be skipped and test 9.24.1.1 shall be executed.~~ |
| | ~~**Test Directives**~~ | ~~The service request shall not contain a password.~~ |
| | ~~**Testing Hints**~~ | |
| ~~**135.1-2019 - 9.24.1.1 - Indefinite Time Duration Restored by DeviceCommunicationControl**~~ | | |
| | ~~**Test Conditionality**~~ | ~~If the IUT does not support indefinite time duration, this test shall be skipped.~~ |

| | | |
|---|---|---|
| | ~~**Test Directives**~~ | ~~The service request shall not contain a password.~~ |
| | ~~**Testing Hints**~~ | |
| ***135.1-2019 - 9.24.1.8 - Finite Time Duration, Disable Initiation*** | | |
| | ***Test Conditionality*** | *If the IUT does not support an internal clock this test shall be skipped* |
| | ***Test Directives*** | *The service request shall not contain a password.* |
| | ***Testing Hints*** | |
| ***135.1-2019 - 9.24.1.6 - Indefinite Time Duration, Disable-Initiation, Restored by DeviceCommunicationControl*** | | |
| | ***Test Conditionality*** | *If the IUT does not support indefinite time duration, this test shall be skipped.* |
| | ***Test Directives*** | *The service request shall not contain a password.* |
| | ***Testing Hints*** | |

[Add test 135.1-2019 - 9.24.1.8 under 8.14.4 in the Test Plan]
[Modify test conditionality of test reference 135.1-2019 – 9.24.1.3 under 8.14.4 in the Test Plan]
[Modify test conditionality of test reference 135.1-2019 – 9.24.1.4 under 8.14.4 in the Test Plan]

**8.14.4 Supports Receiving a DeviceCommunicationControl Service Request with a Finite Duration**

The IUT will accept, or can be made to accept, a DeviceCommunicationControl Service request with a Time Duration parameter.

| | | |
|---|---|---|
| **135.1-2019 - 9.24.1.3 - Finite Time Duration** | | |
| | **Test Conditionality** | ~~Must be executed.~~ *If the IUT claims Protocol_Revision >= 20, this test shall be skipped.* |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.24.1.4 - Finite Time Duration Restored by DeviceCommunicationControl** | | |
| | **Test Conditionality** | ~~Must be executed.~~ *If the IUT claims Protocol_Revision >= 20, this test shall be skipped.* |
| | **Test Directives** | |
| | **Testing Hints** | |
| ***135.1-2019 - 9.24.1.8 - Finite Time Duration, Disable Initiation*** | | |
| | ***Test Conditionality*** | *Must be executed.* |
| | ***Test Directives*** | |
| | ***Testing Hints*** | |

[Modify test conditionality of test reference 135.1-2019 – 9.24.1.1 under 8.14.5 in the Test Plan]
[Add test 135.1-2019 - 9.24.1.6 under 8.14.5 in the Test Plan]

**8.14.5 Supports Receiving a DeviceCommunicationControl Service Request with an Indefinite Duration**

The IUT will accept, or can be made to accept, a DeviceCommunicationControl Service request with no Time Duration parameter.

| | | |
|---|---|---|
| **135.1-2019 - 9.24.1.1 - Indefinite Time Duration Restored by DeviceCommunicationControl** | | |
| | **Test Conditionality** | ~~Must be executed~~ *If the IUT claims Protocol_Revision >= 20, this test shall be skipped.* |
| | **Test Directives** | |
| | **Testing Hints** | |
| ***135.1-2019 - 9.24.1.6 - Indefinite Time Duration, Disable-Initiation, Restored by DeviceCommunicationControl*** | | |
| | ***Test Conditionality*** | *Must be executed.* |
| | ***Test Directives*** | |
| | ***Testing Hints*** | |

[Modify test conditionality of test reference 135.1-2019 – 9.24.1.2 under 8.14.6 in the Test Plan ]
[Modify test conditionality of test reference 135.1-2019 – 9.24.1.5 under 8.14.6 in the Test Plan]
[Add test 135.1-2019 - 9.24.1.7 under 8.14.6 in the test plan]
[Add test 135.1-2019 - 9.24.1.12 under 8.14.6 in the test plan]

**8.14.6 Supports DM-RD-B**

The IUT also supports the DM-RD-B BIBB.

| | | |
|---|---|---|
| **135.1-2019 - 9.24.1.2 - Indefinite Time Duration Restored by ReinitializeDevice** | | |
| | **Test Conditionality** | *If the IUT claims Protocol_Revision >= 20, this test shall be skipped.* If the IUT does not support indefinite Time Duration, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.24.1.5 - Finite Time Duration Restored by ReinitializeDevice** | | |
| | **Test Conditionality** | *If the IUT claims Protocol_Revision >= 20, this test shall be skipped.* If the IUT does not support an internal clock, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| ***135.1-2019 - 9.24.1.7 - Indefinite Time Duration, Disable-Initiation, Restored by ReinitializeDevice*** | | |
| | ***Test Conditionality*** | *If the IUT does not support indefinite Time Duration, this test shall be skipped.* |
| | ***Test Directives*** | |
| | ***Testing Hints*** | |
| ***BTL - 9.24.1.12 - Disable of Service Initiation Restored by ReinitializeDevice*** | | |
| | ***Test Conditionality*** | *If the IUT does not support an internal clock, this test shall be skipped.* |
| | ***Test Directives*** | |
| | ***Testing Hints*** | |
| ***BTL - 9.24.2.3 - Restore by ReinitializeDevice with Invalid 'Reinitialized State of Device'*** | | |
| | Test Conditionality | ~~***Must be executed.***~~ *If the IUT claims Protocol_Revision >= 20, this test shall be skipped.* |
| | Test Directives | *If the IUT does not support an internal clock this test shall be tested with indefinite time duration.* |
| | Testing Hints | |

# Test Changes

[Add test 9.24.1.11 under section 9.24.1 in BTL specified Tests]
**9.24.1.11 Ensure that DISABLE option is not supported by IUT claiming PR >= 20**
*Reason for change: This is new test which address the requirement of 135-2016bi-2 and CR-0492.*

Purpose: To verify that IUT claiming Protocol Revision (PR) greater than or equal to 20, does not accept Enable/Disable' parameter equal to DISABLE in the DeviceCommunicationControl request.

Test Concept: Send DeviceCommunicationControl request with 'Enable/Disable' parameter equal to DISABLE to IUT. Then IUT is verified that it correctly responds with a Result(-).

Configuration Requirements: If the IUT does not support an internal clock this test shall be tested with indefinite time duration.

Test Steps:

1. TRANSMIT DeviceCommunicationControl-Request
   'Time Duration' = (a value T>1, in minutes) | (no value)
   'Enable/Disable'= DISABLE,
   'Password' = Any appropriate password if required
2. RECEIVE BACnet-Error-PDU
   Error Class = SERVICES,
   Error Code = SERVICE_REQUEST_DENIED
3. VERIFY (Device, X), System_Status = (any valid value)

[Add test 9.24.1.12 under section 9.24.1 in BTL specified Tests]
**9.24.1.12 Disable of Service Initiation Restored by ReinitializeDevice**
*Reason for Change: This is new test which was missed earlier.*

Purpose: To verify the correct execution of the DeviceCommunicationControl service when DISABLE_INITIATION is requested with a finite time duration. Communication is restored using the ReinitializeDevice service.

Configuration Requirements: The IUT shall be configured to initiate client requests.

Test Steps:

1. MAKE (a condition that would normally cause the IUT to initiate requests)
2. CHECK (that the IUT is initiating requests)
3. TRANSMIT DeviceCommunicationControl-Request,
   'Time Duration' = (a value in minutes > time required to execute all test steps),
   'Enable/Disable' = DISABLE_INITIATION,
   'Password' = (any appropriate password if required)
4. RECEIVE BACnet-SimpleACK-PDU
5. MAKE (a condition that would normally cause IUT to initiate requests)
6. CHECK (that the IUT has stopped initiating requests)
7. VERIFY (any supported property) = (any valid value)
8. TRANSMIT Who-Is-Request
9. RECEIVE I-Am-Request
10. TRANSMIT ReinitializeDevice-Request,
    'Reinitialized State of Device' = WARMSTART,
    'Password' = (any appropriate password)
11. RECEIVE BACnet-Simple-ACK-PDU
12. CHECK (Did the IUT perform a WARMSTART reboot?)
13. MAKE (a condition that would normally cause the IUT to initiate requests)
14. CHECK (that the IUT is initiating requests)

[Modify test step 1 in test 135.1-2019 – 9.24.2.1 and move into BTL specified tests]

**9.24.2.1 Invalid Password**
*Reason for Change: Modify the parameter value from DISABLE to DISABLE_INITIATION in step 1*

Purpose: To verify the correct execution of DeviceCommunicationControl service procedure when an invalid password is provided. If the IUT does not provide password protection this test case shall be omitted.

Test Steps:

1. TRANSMIT DeviceCommunicationControl-Request,
   'Enable/Disable' = *DISABLE_INITIATION,* ~~DISABLE~~
   'Password' = (any invalid password)
2. RECEIVE BACnet-Error-PDU,
   Error Class = SECURITY,
   Error Code = PASSWORD_FAILURE
3. VERIFY (Device, X), System_Status = (any valid value)

[Modify test step 1 in test 135.1-2019 – 9.24.2.2 and move into BTL specified tests]
**9.24.2.2 Missing Password**
*Reason for Change: Modify the parameter value from DISABLE to DISABLE_INITIATION in step 1.*

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

Test Steps:
1. TRANSMIT DeviceCommunicationControl-Request,
   'Enable/Disable' = *DISABLE_INITIATION* ~~*DISABLE*~~
2. IF (Protocol_Revision >= 7) THEN
   RECEIVE BACnet-Error-PDU,
   Error Class = SECURITY,

```
            Error Code = PASSWORD_FAILURE
ELSE
      RECEIVE BACnet-Error-PDU,
            Error Class = SECURITY,
            Error Code = PASSWORD_FAILURE |
      (RECEIVE BACnet-Error-PDU,
            Error Class = SERVICES,
            Error Code = MISSING_REQUIRED_PARAMETER)
```

3. VERIFY (Device, X), System_Status = (any valid value)

**BTL-18.1-bi-2: Add Audit Log and Audit Reporter Testing [BTLWG-420, BTLWG-422]**

**Overview:**

Add testing for the Audit Log and Audit Reporter objects and associated BIBBs:

Audit Reporting-Logging-A (AR-L-A)
Audit Reporting-Reporter-B (AR-R-B)
Audit Reporting-Reporter-Simple-B (AR-R-S-B)
Audit Reporting-Forwarder-B (AR-F-B)
Audit Reporting-View-A (AR-V-A)
Audit Reporting-Advanced View and Modify-A (AR-AVM-A)

**Changes:**

# BTL Checklist Changes

[ In the Checklist section, deletions should be shown in ~~strikethrough~~, and additions in *italics*]
[ If a complete new section is being added in, do not use italics]

[Replace the Audit Reporter and Audit Log object sections]

| | | |
|---|---|---|
| **Audit Reporter Object** | | |
| | R | Base Requirements |
| | | |
| **Audit Log Object** | | |
| | R | Base Requirements |
| | | |

[Replace the Audit Reporting sections]

| | | |
|---|---|---|
| **Audit Reporting-Logging-A** | | |
| | R | Base Requirements |
| | R | Supports matching and combining of audit notifications |
| | R | Supports hierarchical audit logging |
| | R | Supports execution of AuditLogQuery |
| | | |
| **Audit Reporting-Reporter-B** | | |
| | R | Base Requirements |
| | R | Supports operation target auditing |
| | C[1] | Supports operation source auditing |
| | C[2] | Generates UnconfirmedAuditNotifications |
| | C[2] | Generates ConfirmedAuditNotifications |
| | O | Supports Delaying of Audit Notifications |
| [1] Required if the IUT is able to operate as a BACnet client.<br>[2] At least one of these options must be supported. | | |
| **Audit Reporting-Reporter-Simple-B** | | |
| | R | Base Requirements |
| | R | Supports operation target auditing |
| | C[1] | Supports operation source auditing |
| | C[2] | Generates UnconfirmedAuditNotifications |
| | C[2] | Generates ConfirmedAuditNotifications |
| | O | Supports Delaying of Audit Notifications |
| [1] Required if the IUT is able to operate as a BACnet client.<br>[2] At least one of these options must be supported. | | |
| **Audit Reporting-Forwarder-B** | | |
| | R | Base Requirements |

| | | |
|---|---|---|
| **Audit Reporting-View-A** | | |
| | R | Base Requirements |
| | C[1] | Supports reading Audit Logs using AuditLogQuery |
| | C[1] | Supports reading Audit Logs using ReadRange |
| [1] At least one of these options is required. | | |
| **Audit Reporting-Advanced View and Modify-A** | | |
| | R | Base Requirements |
| | R | Supports reading Audit Logs using AuditLogQuery |
| | R | Supports reading Audit Logs using ReadRange |
| | R | Supports DS-RP-A |
| | R | Supports DS-WP-A |
| | R | Supports DM-OCD-A |

## BTL Test Plan Changes

[ Replace section 3.63, Audit Reporter object]

# 3.63 Audit Reporter Object

## 3.63.1 Base Requirements

Base requirements must be met by any IUT that can contain Audit Reporter objects.

| **Verify Checklist** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Verify that at least one of Audit Reporting-Reporting-B or Audit Reporting-Forwarder-B is claimed. |
| | **Testing Hints** | |

[ Replace section 3.64, Audit Log object]

# 3.64 Audit Log Object

## 3.64.1 Base Requirements

Base requirements must be met by any IUT that can contain Audit Log objects.

| **Verify Checklist** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Verify that at least one of Audit Reporting-Logging-A or Audit Reporting-Forwarder-B is claimed. |
| | **Testing Hints** | |

[ Replace section 13 ]

# 13.1 Audit Reporting-Logging-A

## 13.1.1 Base Requirements

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

| **Verify Checklist** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Verify that support for the Audit Log object is claimed. |
| | **Testing Hints** | |
| **BTL - 7.3.2.X61.1 - One Audit Log Holds all of an Objects History Test** | | |
| | **Test Conditionality** | Must be executed. |

| | | |
|---|---|---|
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.1 - Reading All Items in the List** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 7.3.2.24.1 - Enable Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 7.3.2.24.7 - Buffer_Size Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 7.3.2.24.8 - Record_Count Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 7.3.2.24.9 - Total_Record_Count Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.2 - Reading Items by Position with Positive Count** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.3 - Reading Items by Position with Negative Count** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.4 - Reading Items by Time** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.4.1 - Reading Items by Time with Negative Count** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.9 - Reading Items by Sequence with Positive Count** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 9.21.1.10 - Reading Items by Sequence with Negative Count** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.7 - Reading a Range of Items that do not Exist (by Sequence)** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.8 - Reading a Range of Items that do not Exist (by Time)** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **135.1-2019 - 9.21.1.13 - Reading Items with Negative Count and MOREITEMS** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

| BTL - 7.3.2.X61.7 - Accepts Audit Notifications from an Audit Forwarder Test | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.1.2Supports Matching and Combining of Audit Notifications

The IUT supports matching and combining of audit notifications as performed by full audit loggers.

| BTL - 7.3.2.X61.2 - Audit Notification Basic Combining Test | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X61.3 - Audit Notification Combining Failure Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X61.4 - Audit Notification Non-combining Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X61.5 - Audit Notification Combining Duplicate Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X61.6 - Audit Notification Combining Target Value Preference Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.1.3 Supports Hierarchical Logging

The IUT supports forwarding audit notifications to a parent logger with Delete_On_Forward set to FALSE.

| BTL - 7.3.2.X61.8 - Hierarchical Logging Test | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **VERIFY Checklist** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Verify the IUT claims support for AR-F-B. |
| | **Testing Hints** | |

## 13.1.4 Supports Execution of AuditLogQuery

The IUT supports the reading of its Audit Log object via the AuditLogQuery service.

| BTL - 9.X33.1.1 - AuditLogQuery By Target Test | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 9.X33.1.2 - AuditLogQuery By Source Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 9.X33.2.1 - AuditLogQuery For Non-existent Audit Log** | | |

| | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

# 13.2 Audit Reporting-Reporter-B

## 13.2.1 Base Requirements

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

| **BTL - 7.3.1.X499.1 - Audit_Notification_Recipient Test** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.1.X498.1 - Object Specific Configurable Audit_Level NONE Test** | | |
| | **Test Conditionality** | If the IUT does not contain any non-Audit Reporter objects with configurable Audit_Level properties, this test shall be skipped. |
| | **Test Directives** | Apply to one non-Audit Reporter object which has a configurable Audit_Level property. |
| | **Testing Hints** | |
| **BTL - 7.3.1.X498.2 - Audit Reporter Audit_Level Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.1.X498.3 - Audit_Level Change Notification Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **VERIFY EPICS** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Verify that Audit_Level, Auditable_Operations, Audit_Priority_Filter properties are writable in all Audit Reporter objects. |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.1 - Monitored Objects Test** | | |
| | **Test Conditionality** | If the IUT does not contain any Audit Reporter objects with a Monitored_Objects property, this test shall be skipped.<br>If the IUT contains a single Audit Reporter object and its Monitored_Objects property always references all objects in the IUT, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.2.2 Supports Operation Target Auditing

The IUT supports audit reporting. All devices which support audit reporting must be a capable of reporting operations applied to them.

| **BTL - 7.3.2.X62.4 - Target Audit Reporting - Basic Notification Test** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.5 - Target Audit Reporting - Unconfirmed Service Operation Test** | | |
| | **Test Conditionality** | If the IUT does not support auditable unconfirmed operations, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.6 - Target Audit Reporting - Confirmed Service Operation Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |

| | | |
|---|---|---|
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.7 - Target Audit Reporting - Operations with Priority Test** | | |
| | **Test Conditionality** | If the IUT does not support commandable objects, this test shall be skipped. |
| | **Test Directives** | Apply the test to one commandable object. |
| | **Testing Hints** | |
| **BTL - 7.3.1.X500.1 - Audit_Priority_Filter Target Audit Reporting Test** | | |
| | **Test Conditionality** | If the IUT does not support commandable objects, this test shall be skipped. If the IUT does not support the Audit_Priority_Filter property, this test shall be skipped. |
| | **Test Directives** | Apply the test to one commandable object. |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.8 - Target Audit Reporting - Target_Value and Current_Value Test** | | |
| | **Test Conditionality** | If the IUT does not support operations which contain a value (such as writes), this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.9 - Target Audit Reporting - Error Audit Notification Test** | | |
| | **Test Conditionality** | If the IUT does not report errors via audit notifications, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.10 - Target Audit Reporting - GENERAL Operation Test** | | |
| | **Test Conditionality** | If the IUT does not generate GENERAL operation audit notifications, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.1.X501.2 - Auditable_Operations Target Audit Reporting Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.2.3 Supports Operation Source Auditing

The IUT supports operating as a BACnet client.

| | | |
|---|---|---|
| **BTL - 7.3.2.X62.11 - Source Audit Reporting - Basic Notification Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.13 - Source Audit Reporting - Unconfirmed Service Operation Test** | | |
| | **Test Conditionality** | If the IUT cannot perform auditable unconfirmed service operations, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.14 - Source Audit Reporting - Confirmed Service Operation Audit Notification** | | |
| | **Test Conditionality** | If the IUT cannot perform auditable confirmed service operations, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.15 - Source Audit Reporting - Operations with Priority Test** | | |
| | **Test Conditionality** | If the IUT cannot perform auditable operations which include a priority, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.16 - Source Audit Reporting - Error Audit Notification Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.17 - Source Audit Reporting - Single Source Audit Reporter Object Test** | | |

| | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.1.X501.3 - Auditable_Operations Source Audit Reporting Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

### 13.3.4 Generates UnconfirmedAuditNotifications

The IUT is able to report audit notifications using UnconfirmedAuditNotification requests.

| **Verify Test Selection** | | |
|---|---|---|
| | **Test Conditionality** | |
| | **Test Directives** | Ensure that a test from another section is executed with the IUT configured to send UnconfirmedAuditNotifications. |
| | **Testing Hints** | |

### 13.3.5 Generates ConfirmedAuditNotifications

The IUT is able to report audit notifications using ConfirmedAuditNotification requests.

| **Verify Test Selection** | | |
|---|---|---|
| | **Test Conditionality** | |
| | **Test Directives** | Ensure that a test from another section is executed with the IUT configured to send ConfirmedAuditNotifications. |
| | **Testing Hints** | |

### 13.2.6Supports Delaying of Audit Notifications

The IUT supports delaying of audit notifications to allow multiple notifications to be sent in a single audit notification request.

| **BTL - 7.3.2.X62.2 - Maximum_Send_Delay Test** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.3 - Send_Now Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.3 Audit Reporting-Reporter-Simple-B

### 13.3.1Base Requirements

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

| **BTL - 7.3.1.X499.1 - Audit_Notification_Recipient Test** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.1.X498.1 - Object Specific Configurable Audit_Level NONE Test** | | |
| | **Test Conditionality** | If the IUT does not contain any non Audit Reporter objects with configurable Audit_Level properties, this test shall be skipped. |
| | **Test Directives** | Apply to one non-Audit Reporter object which has a configurable Audit_Level property. |
| | **Testing Hints** | |
| **BTL - 7.3.1.X498.2 - Audit Reporter Audit_Level Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.1.X498.3 - Audit_Level Change Notification Test** | | |

| | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.1.X501.1 - Non-configurable Auditable_Operations Property Test** | | |
| | **Test Conditionality** | If the IUT does not contain any objects with a non-configurable Auditable_Operations property, this test shall be skipped. |
| | **Test Directives** | Repeat once for each object type for which at least 1 instance has a non-configurable Auditable_Operations property. |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.1 - Monitored Objects Test** | | |
| | **Test Conditionality** | If the IUT does not contain any Audit Reporter objects with a Monitored_Objects property, this test shall be skipped.<br>If the IUT contains a single Audit Reporter object and its Monitored_Objects property always references all objects in the IUT, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.3.2 Support Operation Target Auditing

The IUT supports audit reporting. All BACnet devices which support audit reporting are required to generation audit notifications for operations performed on the IUT and / or its objects.

| | | |
|---|---|---|
| **BTL - 7.3.2.X62.4 - Target Audit Reporting - Basic Notification Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.5 - Target Audit Reporting - Unconfirmed Service Operation Test** | | |
| | **Test Conditionality** | If the IUT does not support unconfirmed auditable operations, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.6 - Target Audit Reporting - Confirmed Service Operation Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.7 - Target Audit Reporting - Operations with Priority Test** | | |
| | **Test Conditionality** | If the IUT does not support commandable objects, this test shall be skipped. |
| | **Test Directives** | Apply the test to one commandable object. |
| | **Testing Hints** | |
| **BTL - 7.3.1.X500.1 - Audit_Priority_Filter Target Audit Reporting Test** | | |
| | **Test Conditionality** | If the IUT does not support commandable objects, this test shall be skipped.<br>If the IUT does not support the Audit_Priority_Filter property, this test shall be skipped. |
| | **Test Directives** | Apply the test to one commandable object. |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.8 - Target Audit Reporting - Target_Value and Current_Value Test** | | |
| | **Test Conditionality** | If the IUT does not support operations which contain a value (such as writes), this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.9 - Target Audit Reporting - Error Audit Notification Test** | | |
| | **Test Conditionality** | If the IUT does not report errors via audit notifications, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.10 - Target Audit Reporting - GENERAL Operation Test** | | |
| | **Test Conditionality** | If the IUT does not generate GENERAL operation audit notifications, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |

| BTL - 7.3.1.X501.2 - Auditable_Operations Target Audit Reporting Test | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.3.3 Support Operation Source Auditing

The IUT is a BACnet client and supports audit reporting. A device which supports audit reporting and which initiates requests (aside from notifications or unconfirmed services generated in response to other requests such as I-Am requests), is required to support source audit reporting.

| BTL - 7.3.2.X62.11 - Source Audit Reporting - Basic Notification Test | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.13 - Source Audit Reporting - Unconfirmed Service Operation Test** | | |
| | **Test Conditionality** | If the IUT cannot perform auditable unconfirmed service operations, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.14 - Source Audit Reporting - Confirmed Service Operation Audit Notification** | | |
| | **Test Conditionality** | If the IUT cannot perform auditable confirmed service operations, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.15 - Source Audit Reporting - Operations with Priority Test** | | |
| | **Test Conditionality** | If the IUT cannot perform auditable operations which include a priority, this test shall be skipped. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.16 - Source Audit Reporting - Error Audit Notification Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.17 - Source Audit Reporting - Single Source Audit Reporter Object Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.1.X501.3 - Auditable_Operations Source Audit Reporting Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.3.4 Generates UnconfirmedAuditNotifications

The IUT is able to report audit notifications using UnconfirmedAuditNotification requests.

| Verify Test Selection | | |
|---|---|---|
| | **Test Conditionality** | |
| | **Test Directives** | Ensure that a test from another section is executed with the IUT configured to send UnconfirmedAuditNotifications. |
| | **Testing Hints** | |

## 13.3.5 Generates ConfirmedAuditNotifications

The IUT is able to report audit notifications using ConfirmedAuditNotification requests.

| Verify Test Selection | | |
|---|---|---|
| | **Test Conditionality** | |
| | **Test Directives** | Ensure that a test from another section is executed with the IUT configured to send ConfirmedAuditNotifications. |
| | **Testing Hints** | |

### 13.3.6 Supports Delaying of Audit Notifications

The IUT supports delaying of audit notifications to allow multiple notifications to be sent in a single audit notification request.

| BTL - 7.3.2.X62.2 - Maximum_Send_Delay Test | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |
| **BTL - 7.3.2.X62.3 - Send_Now Test** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.4 Audit Reporting-Forwarder-B

### 13.4.1 Base Requirements

The IUT supports forwarding audit notifications to a parent logger with Delete_On_Forward set to TRUE.

| BTL - 7.3.2.X62.18 - Audit Forwarding Test | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | |
| | **Testing Hints** | |

## 13.5 Audit Reporting-View-A

### 13.5.1 Base Requirements

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

### 13.5.2 Supports Reading Audit Logs using AuditLogQuery

The IUT is able to read Audit Log objects using the AuditLogQuery service.

| BTL - 8.X33.1 - Reading a Range of Items Using Any Valid Query | | |
|---|---|---|
| | **Test Conditionality** | If the device claims Audit Reporting-Advanced View and Modify-A this test shall be skipped. |
| | **Test Directives** | Use an Audit Log object as the log object for this test. |
| | **Testing Hints** | |

### 13.5.3 Supports Reading Audit Logs using ReadRange

The IUT is able to read Audit Log objects using the ReadRange service.

| VERIFY Checklist | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Verify that the IUT claims support for Initiates ReadRange. |
| | **Testing Hints** | |
| **135.1-2019 - 8.21.8 - Reading a Range of Items Using Any Valid Range** | | |
| | **Test Conditionality** | If the IUT claims Audit Reporting-Advanced View and Modify-A this test may be skipped. |
| | **Test Directives** | Use an Audit Log object as the log object for this test. |
| | **Testing Hints** | |

## 13.6 Audit Reporting-Advanced View and Modify-A

### 13.6.1 Base Requirements

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

| 135.1-2019 - 8.18.3 - Reading and Presenting Properties |
|---|

| | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Repeat for each standard audit reporting property, in a standard object type.<br>Repeat for each property in the Audit Reporter and Audit Log objects. |
| | **Testing Hints** | |
| **135.1-2019 - 8.22.4 - Accepting Input and Modifying Properties** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Repeat for each standard audit reporting property, in a standard object type.<br>Repeat for each property in the Audit Reporter and Audit Log objects which is not mandated as read only by the standard, or to which access is otherwise restricted by the standard. |
| | **Testing Hints** | |

## 13.6.2 Supports Reading Audit Logs using AuditLogQuery

The IUT is able to read Audit Log objects using the AuditLogQuery service.

| **BTL - 8.X33.1 - Reading a Range of Items Using Any Valid Query** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Use an Audit Log object as the log object for this test. |
| | **Testing Hints** | |

## 13.6.3 Supports Reading Audit Logs using ReadRange

The IUT is able to read Audit Log objects using the ReadRange service.

| **VERIFY Checklist** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Verify that the IUT claims support for Initiates ReadRange. |
| | **Testing Hints** | |
| **135.1-2019 - 8.21.8 - Reading a Range of Items Using Any Valid Range** | | |
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Use an Audit Log object as the log object for this test. |
| | **Testing Hints** | |

## 13.6.4 Supports DS-RP-A

The IUT shall support DS-RP-A in order to receive alarm parameters for presentation to the user.

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

## 13.6.5 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update alarm parameters 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** | |

## 13.6.6 Supports DS-OCD-A

The IUT shall support DS-OCD-A in order to allow the user to create Audit Reporter and Audit Log objects.

| **Verify Checklist** | | |
|---|---|---|
| | **Test Conditionality** | Must be executed. |
| | **Test Directives** | Verify that the IUT claims support for DM-OCD-A and that it claims the ability to create and delete Audit Reporter, and Audit Log objects. |
| | **Testing Hints** | |

# Test Changes

[Add into Clause 6 in BTL Specified Tests, a new timer clause]

**6.3.X Audit Notification Forwarder Fail Time**

The **Audit Notification Forwarder Fail Time** is the elapsed time, in seconds, between when a forwarding audit log receives an audit notification and when a test is considered to have failed because the expected audit notification message has not been transmitted.

[Add into Clause 6 in BTL Specified Tests, a new TCSL statement]

**6.2.14 Assignment Statement**

The assignment statement is used to set the value of a TCSL variable

<assignment statement> ::= <variable> '=' '(' <value description> ')'

The <value description> is a simple English phrase describing the content that is to be placed into the variable. It may reference other variables already in use by the test. For example:

READ X = O1, Present_Value
READ Y = O2, Present_Value
MAX = (the larger of X and Y)

[Add Audit Log object tests into BTL Specified Tests]

### 7.3.2.X61 Audit Log Object Tests

#### 7.3.2.X61.1 One Audit Log Holds all of an Objects History Test

Purpose: Ensure that, for any arbitrary object, there is at least one Audit Log into which all of the object's audit notifications are placed.

Test Concept: Send a sequence of audit notifications which contain entries for multiple objects to the IUT. At least some of the objects shall have multiple audit records in the sequence. For each object instance represented in the audit notifications sent to the IUT, verify that there is at least one Audit Log which contains the all of the audit notifications for the object.

Configuration Requirements: S is a sequence of audit notifications which contain entries for multiple objects to the IUT where at least some of the objects shall have multiple audit records in the sequence.

Test Steps:

```
1. REPEAT AN = (each notification in S) DO {
      TRANSMIT UnconfirmedAuditNotification-Request,
           'Notifications' =          AN
}
2. REPEAT O = (each object represented in S) DO {
      SO = (the sequence of notifications in S for object O)
      FOUND = (false)
      REPEAT AL = (each Audit Log object) DO {
           IF (AL contains all notifications in SO) THEN
                FOUND = (true)
      }
      IF (FOUND is false) THEN {
           ERROR "no audit log was found which contains all notifications for object"
      }
}
```

#### 7.3.2.X61.2 Audit Notification Basic Combining Test

Purpose: Ensure that Audit Log objects correctly combine related audit notification records.

Test Concept: Send a sequence, SEQ1, of unrelated audit notifications to the IUT and verify that the notifications are not combined. Send a source audit notification, SN1, followed by a sequence, SEQ2, of unrelated audit notifications and verify that the notifications are not combined. Send a target audit notification, TN1, which should be combined with SN1. Verify that SN1 and TN1 are combined in the Audit Log. Repeat the process with new notifications but send the target notification before the source notification.'

Configuration Requirements: An audit log that should receive the combined SN/TN notification is AL. The Target Value and Current Value fields in SN and TN shall not be greater than 500 octets. D1 shall be the device sending the source notification and D2 shall be the device sending the target notification. It is acceptable if D1 is the same as D2.

Test Steps:

```
-- the source notification is sent before the target notification
1. REPEAT AN = (each notification in SEQ1) DO {
      TRANSMIT UnconfirmedAuditNotification-Request,
           SOURCE =          (D1 or D2),
           'Notifications' =          AN
}
2. TRANSMIT UnconfirmedAuditNotification-Request,
      SOURCE =                D1,
      'Notifications' =                SN1
3. REPEAT AN = (each notification in SEQ2) DO {
```

```
    SOURCE =            (D1 or D2),
    TRANSMIT UnconfirmedAuditNotification-Request,
        'Notifications' =       AN
}
4. TRANSMIT UnconfirmedAuditNotification-Request,
    SOURCE =            D2,
    'Notifications' =           TN1
5. CHECK(that no record exists in AL which is just SN1)
6. CHECK(that no record exists in AL which is just TN1)
7. CHECK(that a record exists in AL which is the combination of SN1 and TN1)
8. CHECK(that the combined record has all of the source and target fields provided in SN1 and TN1, and no more.)

-- the target notification is sent before the source notification
9. REPEAT AN = (each notification in SEQ3) DO {
    TRANSMIT UnconfirmedAuditNotification-Request,
        SOURCE =        (D1 or D2),
        'Notifications' =       AN
}
10. TRANSMIT UnconfirmedAuditNotification-Request,
    SOURCE =            D1,
    'Notifications' =           TN2
11. REPEAT AN = (each notification in SEQ4) DO {
    SOURCE =            (D1 or D2),
    TRANSMIT UnconfirmedAuditNotification-Request,
        'Notifications' =       AN
}
12. TRANSMIT UnconfirmedAuditNotification-Request,
    SOURCE =            D2,
    'Notifications' =           SN2
13. CHECK(that no record exists in AL which is just SN2)
14. CHECK(that no record exists in AL which is just TN2)
15. CHECK(that a record exists in AL which is the combination of SN2 and TN2)
16. CHECK(that the combined record has all of the source and target fields provided in SN2 and TN2, and no more.)
```

### 7.3.2.X61.3 Audit Notification Combining Failure Test

Purpose: Ensure that Audit Log objects correctly combine related audit notification records which indicate failed actions.

Test Concept: Send a source audit notification SN. Send a target audit notification, TN, which should be combined with SN and which indicates that the action failed. Verify that SN and TN are combined in the Audit Log.

Configuration Requirements: An audit log that should receive the combined SN/TN notification is AL. The Target Value and Current Value fields in SN and TN shall not be greater than 500 octets.

Test Steps:

```
1. TRANSMIT UnconfirmedAuditNotification-Request,
    'Notifications' =       SN
2. TRANSMIT UnconfirmedAuditNotification-Request,
    'Notifications' =       TN
3. CHECK(that no record exists in AL which is just SN)
4. CHECK(that no record exists in AL which is just TN)
5. CHECK(that a record exists in AL which is the combination of SN and TN)
6. CHECK(that the combined record has all of the source and target fields provided in SN and TN, and no more.)
```

### 7.3.2.X61.4 Audit Notification Non-combining Test

Purpose: Ensure that Audit Log objects correctly do not combine unrelated audit notification records.

Test Concept: Send a sequence of unrelated audit notifications to the IUT each differing by 1 field in the matching criteria. Verify that the notifications are not combined.

Configuration Requirements: SEQ is a sequence of audit notifications where each record differs from the previous record by 1 field. For the sequence, SN is a source notification with user-id, user-role, target-value fields, and without source-comment field, and TN is the matching target notification with user-id, user-role, target-value fields. The sequence is:

{
(SN),
(SN but with source-comment field),
(TN with differing operation-source),
(TN with differing operation),
(TN with differing invoke-id),
(TN with differing target-device),
(TN with differing target-property),
(TN with differing user-id),
(TN with differing user-role),
(TN with differing target-value),
(TN with target-timestamp equal to source-timestamp + APDU_Timeout * 3)
}

Test Steps:

1. REPEAT AN = (each notification in SEQ) DO {
   TRANSMIT UnconfirmedAuditNotification-Request,
      'Notifications' = AN
}
2. REPEAT AN = (each notification in SEQ) DO {
   CHECK(that AN is in an Audit Log and is not combined)
}

### 7.3.2.X61.5 Audit Notification Combining Duplicate Test

Purpose: Ensure that Audit Log objects correctly drop duplicate notifications.

Test Concept: Send a source audit notification SN. Verify it is placed in the log. Send a sequence, SEQ1, of unrelated audit notifications and verify SN is not combined with any. Resend SN and verify that SN was not re-added to the log.
Send a target audit notification, TN, which should be combined with SN. Verify that SN and TN are combined in the Audit Log. Send a sequence, SEQ2, of unrelated audit notifications. Resend TN and verify that TN was not re-added to the log.

Test Steps:

1. TRANSMIT UnconfirmedAuditNotification-Request,
   'Notifications' = SN
2. CHECK(that SN is in the Audit Log)
3. REPEAT AN = (each notification in SEQ1) DO {
   TRANSMIT UnconfirmedAuditNotification-Request,
      'Notifications' = AN
}
4. CHECK(that SN is in the Audit Log and is not combined)
5. TRANSMIT UnconfirmedAuditNotification-Request,
   'Notifications' = SN
6. CHECK(that SN is in the Audit Log only once and is at its original position)

7. TRANSMIT UnconfirmedAuditNotification-Request,
   'Notifications' = TN
8. CHECK(that SN is in the Audit Log only once, is combined with TN, and is at its original position)
9. CHECK(that the combined record has all of the source and target fields provided in SN and TN, and no more.)

10. REPEAT AN = (each notification in SEQ2) DO {
    TRANSMIT UnconfirmedAuditNotification-Request,
        'Notifications' = AN
}
11. TRANSMIT UnconfirmedAuditNotification-Request,
    'Notifications' = TN
12. CHECK(that TN is in the Audit Log only once, is combined with SN and is at SN's original position)

### 7.3.2.X61.6 Audit Notification Combining Target Value Preference Test

Purpose: Ensure that Audit Log objects use the Current Value from a target notification when it is provided in both the source and target notifications.

Test Concept: Send a target audit notification TN1 which includes the Current Value field with a value CV1-T. Send a source audit notification, SN1, which should be combined with TN1, and which contains a Current Value field with a value CV1-S (CV1-S is different than CV1-T). Verify that SN1 and TN1 are combined in the Audit Log and that the Current Value in the log uses CV1-T. Repeat the steps sending a target notification TN2 before the source notification SN2 where CV2-S is different than CV2-T.

Test Steps:

1. TRANSMIT UnconfirmedAuditNotification-Request,
    'Notifications' = TN1
2. TRANSMIT UnconfirmedAuditNotification-Request,
    'Notifications' = SN1
3. CHECK(that TN1 is in the Audit Log only once, is combined with SN1, and that target-value is CV1-T)

4. TRANSMIT UnconfirmedAuditNotification-Request,
    'Notifications' = SN2
5. TRANSMIT UnconfirmedAuditNotification-Request,
    'Notifications' = TN2
6. CHECK(that SN2 is in the Audit Log only once, is combined with TN2, and that target-value is CV2-T)

### 7.3.2.X61.7 Accepts Audit Notifications from an Audit Forwarder Test

Purpose: Ensure that Audit Log accepts forwarded audit notifications.

Test Concept: The notification forwarder, AF1, sends a forwarded source notification, SN1, from the original sending device D1, to the IUT. Verify that the IUT places the notification in the Audit Log. AF1 then sends a forwarded target notification, TN1, from the original target device D2, to the IUT. Verify that the IUT combines the target with the source notification.

Configuration Requirements: The test network consists of a source device D1, a target device D2, and a notification forwarder, AF1.

Test Steps:

1. TRANSMIT UnconfirmedAuditNotification-Request,
    SOURCE = AF1,
    'Notifications' = SN1
2. TRANSMIT UnconfirmedAuditNotification-Request,
    SOURCE = AF1,
    'Notifications' = TN1
3. CHECK(that SN1 and TN1 are combined in the Audit Log)

### 7.3.2.X61.8 Hierarchical Logging Test

Purpose: Ensure that an Audit Log configured with a parent correctly forwards notifications to the parent log.

Test Concept: An Audit Log, O1, configured to reference a parent log located in another device, is sent a sequence of audit notifications. Within the sequence will some notifications which should be combined and some which should not be

combined. Verify that the IUT forwards the notifications to the parent before the vendor specified maximum forwarding delay.

Configuration Requirements: The Audit Log, O1, references a parent audit log located in the TD.

Notes to Tester: The standard does not provide guidance on how long an Audit Log object has before it must forward audit notifications to its parent. As such, the vendor is allowed to specify the maximum time as long as it is not unreasonable (delays on the order of days are clearly unreasonable; delays on the order of minutes are clearly acceptable).

Configuration Requirements: The Audit Log, AL1, is configured with a Member_Of set to AL2, where AL2 is in the TD. AL1's Delete_On_Forward shall be set to TRUE. SEQ is a sequence of audit notifications where at least 2 are related and should be combined.

Test Steps:
1. REPEAT AN = (each notification in SEQ) DO {
    TRANSMIT UnconfirmedAuditNotification-Request,
        SOURCE = (a value appropriate to the notification),
        'Notifications' = AN
}
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE **Audit Notification Forwarder Fail Time**
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = (one or more of the notifications from SEQ)
} ELSE {
    BEFORE **Audit Notification Forwarder Fail Time**
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = (one or more of the notifications from SEQ)
    TRANSMIT BACnet-SimpleACK-PDU
}

3. WHILE (not all notifications in SEQ have been sent by the IUT) {
    IF the IUT is configured to send unconfirmed audit notifications THEN {
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = (one or more of the as yet unreceived notifications from SEQ)
    } ELSE {
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = (one or more of the as yet unreceived notifications from SEQ)
        TRANSMIT BACnet-SimpleACK-PDU
    }
}
4. CHECK(that the notifications in SEQ are still in AL1)
5. CHECK(that the notifications in SEQ which are to be combined are combined in AL1)

Notes to Tester: When receiving notifications from the IUT, those notifications which should be combined, may sent combined or not at the IUT's discretion.

[Add Audit Report object tests into BTL Specified Tests]

**7.3.2.X62 Audit Reporter Object Tests**

**7.3.2.X62.1 Monitored Objects Test**

Purpose: Verify that the correct Audit Reporter is used for an auditable object.

Test Concept: Each Audit Reporter, which contains a Monitored_Objects property, in the device is tested individually. The selected Audit Reporter is enabled, and all others are disabled. An object that the enabled Audit Reporter reports for is selected. An auditable operation is performed on the object and it is verified that an audit notification is generated. An object that the enabled Audit Reporter does not report for is selected. An auditable operation is performed on the object and it is verified that no audit notification is generated.

Configuration Requirements: The IUT is configured to send unconfirmed audit notifications. If the Monitored_Objects property is not supported by any Audit Reporter objects in the IUT, this test shall be skipped. If the IUT only supports a single Audit Reporter object for target object reporting, and its Monitored_Objects property is always set to all objects, this test shall be skipped. Configure all Audit Reporters to report all operations and set Audit_Level to NONE for all Audit Reporter objects.

Test Steps:

```
1. IF the Monitored_Objects property is writable in one or more of the AR objects THEN
      MAKE(reconfigure which AR is used by which objects)
2. REPEAT AR = (each Audit Reporter object) DO {
      O1 = (an object O1 which reports thru AR as determined by Monitored_Objects and AR precedence)
      O2 = (an object O2 which reports thru a different Audit Reporter, AR2, as determined by
               Monitored_Objects and AR precedence)
      WRITE AR.Audit_Level = AUDIT_ALL
      MAKE(perform an auditable operation on O1)
      IF the IUT is configured to send unconfirmed auto notifications THEN {
           BEFORE AR.Maximum_Send_Delay + Notification Fail Time
               RECEIVE UnconfirmedAuditNotification-Request,
                    'Notifications' =   (a notification indicating the operation)
      } ELSE {
           BEFORE AR.Maximum_Send_Delay + Notification Fail Time
               RECEIVE ConfirmedAuditNotification-Request,
                    'Notifications' =   (a notification indicating the operation)
               TRANSMIT BACnet-SimpleACK-PDU
      }
      MAKE(perform an auditable operation on O2)
      WAIT AR2.Maximum_Send_Delay + Notification Fail Time
       CHECK(that the IUT did not report an audit notification for the operation)
      WRITE AR.Audit_Level = NONE
}
```

Notes to Tester: In this test, the audit reporting configuration properties are assumed to be writable. If one is only configurable, then the WRITE operation should be replaced with MAKE.

**7.3.2.X62.2 Maximum_Send_Delay Test**

Purpose: Verify that Audit Reporter objects abide the Maximum_Send_Delay value.

Test Concept: An Audit Reporter object is selected which contains a Maximum_Send_Delay property. A sequence of N auditable operations is performed on the IUT, and if the IUT is a client, the IUT is made to perform a sequence of M auditable operations. The IUT is then monitored to ensure that the audit notifications are sent within the selected Maximum_Send_Delay.

N, the number of operations performed on the IUT shall be 2 or greater. M, the number of operations that the IUT will perform shall be 0 if the IUT does not perform auditable operations, and 2 or greater otherwise.

The value that Maximum_Send_Delay is configured for shall be large enough to allow for all of the operations to be performed.

Test Steps:
1. WRITE AR1, Maximum_Send_Delay = (MSD: a value large enough to accomplish and report N+M auditable operations)
2. WRITE AR2, Maximum_Send_Delay = MSD
3. MAKE(perform N auditable operations on the IUT which would be reported through AR1)
4. MAKE(the IUT perform M auditable operations which would be reported through AR2)
5. WAIT Maximum_Send_Delay + (Notification Fail Time * M+N)
6. CHECK(that all expected operations are reported)

**7.3.2.X62.3 Send_Now Test**

Purpose: Verify that writing True to Send_Now results in the sending of delayed audit notifications.

Test Concept: The IUT is configured to delay sending audit notifications. An Audit Reporter object, AR, is selected which contains a Send_Now property. An auditable operation is performed on the IUT. Send_Now is then written and it is verified that the audit notifications are sent.

Configuration Requirements: If the IUT cannot be made to delay sending notifications without heroic efforts, this test shall be skipped.

Test Steps:

1. WRITE AR, Maximum_Send_Delay = (a value large enough to accomplish the test)
2. MAKE(perform whatever actions are required to make the IUT delay sending notifications)
3. WRITE AR, Send_Now = TRUE
4. IF the IUT is configured to send unconfirmed notifications THEN {
    BEFORE Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = (one or more notifications for operation applied to the IUT)
} ELSE {
    BEFORE Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = (one or more notifications for operation applied to the IUT)
    TRANSMIT BACnet-SimpleACK-PDU
}

**7.3.2.X62.4 Target Audit Reporting - Basic Notification Test**

Purpose: Verify that target audit notifications are properly formed.

Test Concept: The IUT is made to send a target audit notification. It is verified that the target fields are present, no source fields are present, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report all auditable operations. The IUT is configured so that the notification will be reported through Audit Reporter AR.

Test Steps:

1. MAKE(perform an auditable operation, O, on IUT)
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                -- source-timestamp absent
                target-timestamp = (IUT's local time),
                source-device = TD,
                -- source-object absent

```
                operation = O,
                -- source-comment absent
                target-comment = (any valid value, or absent unless O is GENERAL),
                invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = IUT,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                        shall be 16 or absent if no priority supplied and the target is a
                        property),
                target-value = (the target value or absent if no target value for the operation.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                -- source-timestamp absent
                target-timestamp = (IUT's local time),
                source-device = TD,
                -- source-object absent
                operation = O,
                -- source-comment absent
                target-comment = (any valid value, or absent unless O is GENERAL),
                invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = IUT,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                        shall be 16 or absent if no priority supplied and the target is a
                        property),
                target-value = (the target value or absent if no target value for the operation.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
    TRANSMIT BACnet-SimpleACK-PDU
}
```

### 7.3.2.X62.5 Target Audit Reporting - Unconfirmed Service Operation Test

Purpose: Verify that target audit notifications for unconfirmed services do not contain InvokeId information.

Test Concept: An auditable unconfirmed service is performed on the IUT and it is verified that the resulting target audit notification does not contain an InvokeId, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report audit notifications for an unconfirmed service. If the IUT does not support audit reporting for any unconfirmed services, this test shall be skipped. The IUT is configured so that the notification will be reported through Audit Reporter AR.

Test Steps:

1. MAKE(perform an auditable operation, O, which uses an unconfirmed service, on IUT)
2. IF the IUT is configured to send unconfirmed audit notifications for operation O THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                -- source-timestamp absent
                target-timestamp = (IUT's local time),
                source-device = TD,
                -- source-object absent
                operation = O,
                -- source-comment absent
                target-comment = (any valid value, or absent unless O is GENERAL),
                -- invoke-id absent,
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = IUT,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                        shall be 16 or absent if no priority supplied and the target is a
                        property),
                target-value = (the target value or absent if no target value for the operation.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                -- source-timestamp absent
                target-timestamp = (IUT's local time),
                source-device = TD,
                -- source-object absent
                operation = O,
                -- source-comment absent
                target-comment = (any valid value, or absent unless O is GENERAL),
                -- invoke-id absent,
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = IUT,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                        shall be 16 or absent if no priority supplied and the target is a
                        property),
                target-value = (the target value or absent if no target value for the operation.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded

```
                    octets),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
    TRANSMIT BACnet-SimpleACK-PDU
}
```

**7.3.2.X62.6 Target Audit Reporting - Confirmed Service Operation Audit Notification**

Purpose: Verify that target audit notifications for confirmed services contain InvokeId information.

Test Concept: An auditable confirmed service is performed on the IUT and it is verified that the resulting target audit notification contains an InvokeId, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report audit notifications for an unconfirmed service. The IUT is configured so that the notification will be reported through Audit Reporter AR.

Test Steps:

```
1. MAKE(perform an auditable operation, O, which uses an unconfirmed service, on IUT)
2. IF the IUT is configured to send unconfirmed audit notifications for operation O THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( { -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = O,
                    -- source-comment absent
                    target-comment = (any valid value, or absent unless O is GENERAL),
                    -- invoke-id absent,
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = (the target object or absent if the target is not an object),
                    target-property = (the target property or absent if the target is not a
                        property),
                    target-priority = (the priority supplied, or absent if the target is not a
                        property. shall be 16 or absent if no priority supplied and the
                        target is a property),
                    target-value = (the target value or absent if no target value for the
                        operation. may be absent if the value size is larger than 32
                        encoded octets),
                current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( { -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = O,
                    -- source-comment absent
                    target-comment = (any valid value, or absent unless O is GENERAL),
                    -- invoke-id absent,
```

```
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = (the target object or absent if the target is not an object),
                    target-property = (the target property or absent if the target is not a
                        property),
                    target-priority = (the priority supplied, or absent if the target is not a
                        property. shall be 16 or absent if no priority supplied and the
                        target is a property),
                    target-value = (the target value or absent if no target value for the
                        operation. may be absent if the value size is larger than 32
                        encoded octets),
                current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
        TRANSMIT BACnet-SimpleACK-PDU
}
```

**7.3.2.X62.7 Target Audit Reporting - Operations with Priority Test**

Purpose: Verify that target audit notifications which for writes which convey a priority include the priority in the notification.

Test Concept: An auditable write, which includes a priority, is performed on a commandable object in the IUT and it is verified that the resulting target audit notification contains a priority, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report audit notifications for writes on a commandable object, O1. The IUT is configured so that the notification will be reported through Audit Reporter AR using unconfirmed notifications. If the IUT does not support the Priority_Array property in any object for which audit reporting can be configured, this test shall be skipped.

Test Steps:

```
1. TRANSMIT WriteProperty-Request,
     'Invoke Id' =        I,
     'Object Identifier' =    O1,
     'Property Identifier' = Present_Value,
     'Property Value' =       (V: any valid value),
     'Priority' =         (PRIO: a priority in the range 1 - 15)
2. BEFORE Internal Processing Fail Time
     RECEIVE BACnet-SimpleACK-PDU
          |
          (BACnet-Error-PDU,
               'Error Type' = (E : any error)
          )
3. IF the IUT is configured to send unconfirmed audit notifications THEN {
     BEFORE AR.Maximum_Send_Delay + Notification Fail Time
          RECEIVE UnconfirmedAuditNotification-Request,
               'Notifications' = ( {
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = WRITE,
                    -- source-comment absent
                    target-comment = (any valid value, or absent),
                    invoke-id = I,
```

```
                    source-user-id = (the value from the operation if provided, otherwise absent),
                    source-user-role = (the value from the operation if provided, otherwise absent),
                    target-device = IUT,
                    target-object = O1,
                    target-property = Present_Value,
                    target-priority = PRIO,
                    target-value = V,
                    current-value = (the value before the write. may be absent if the value size is
                        larger than 32 encoded octets),
                    result = (E, if the op failed, otherwise absent)
                    } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = WRITE,
                    -- source-comment absent
                    target-comment = (any valid value, or absent),
                    invoke-id = I,
                    source-user-id = (the value from the operation if provided, otherwise absent),
                    source-user-role = (the value from the operation if provided, otherwise absent),
                    target-device = IUT,
                    target-object = O1,
                    target-property = Present_Value,
                    target-priority = PRIO,
                    target-value = V,
                    current-value = (the value before the write. may be absent if the value size is
                        larger than 32 encoded octets),
                    result = (E, if the op failed, otherwise absent)
                    } )
        TRANSMIT BACnet-SimpleACK-PDU

}
```

### 7.3.2.X62.8 Target Audit Reporting - Target_Value and Current_Value Test

Purpose: Verify that the IUT reports target values and current values, when the audited operation contains a value (such as for writes).

Test Concept: An auditable operation, which contains a value, is performed on object O1 and property P1. The resulting audit notification is verified to contain the provided value and the value before the operation.

Configuration Requirements: The IUT is configured to report all audit notifications. If possible, a property which is not changing shall be the target of the operation so that the current value field can be validated. If the IUT does not have any objects which support reporting of operation which contain target value, this test shall be skipped. AR is the Audit Report through which O1 reports audit notifications.

Test Steps:

```
1. IF P1 is not changing outside of the operation THEN {
    READ IV =  O1, P1
2. MAKE(perform an auditable operation, on O1, P1, which provides a target value, V,
    which is less than 32 octets in size)
3. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ({
```

```
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = (the operation performed),
                    -- source-comment absent
                    target-comment = (any valid value, or absent),
                    invoke-id = (the invoke id from the operation, or absent if it was
                        unconfirmed),
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = O1,
                    target-property = P1,
                    target-priority = (the priority from the operation, or absent if 16 or not
                        provided in the operation),
                    target-value = V,
                    current-value = (CV: any valid value),
                    result = (E, if the op failed, otherwise absent)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ({
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = (the operation performed),
                    -- source-comment absent
                    target-comment = (any valid value, or absent),
                    invoke-id = (the invoke id from the operation, or absent if it was
                        unconfirmed),
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = O1,
                    target-property = P1,
                    target-priority = (the priority from the operation, or absent if 16 or not
                        provided in the operation),
                    target-value = V,
                    current-value = (CV: any valid value),
                    result = (E, if the op failed, otherwise absent)
                } )
        TRANSMIT BACnet-SimpleACK-PDU
}
4. IF the P1 is not changing outside of the operation THEN
    CHECK(CV equals IV)
```

### 7.3.2.X62.9 Target Audit Reporting - Error Audit Notification Test

Purpose: Verify that operations that fail are properly reported in audit notifications.

Test Concept: An auditable operation, which will fail with a Result(-) or Result(+) with error information is performed on the IUT. It is verified that an audit notification is sent which contains the error that occurred. The auditable operation performed shall be one for which the IUT will report failures via audit notifications.

Configuration Requirements: The IUT is configured to report all audit notifications.

Test Steps:
1. MAKE(perform an auditable operation, O, on IUT which will fail (via return of a BACnetErrorPDU, or a Result(+) with error information)
2. IF the IUT is configured to send unconfirmed audit notifications for operation O THEN {
BEFORE AR.Maximum_Send_Delay + Notification Fail Time
RECEIVE UnconfirmedAuditNotification-Request,
'Notifications' = ({
-- source-timestamp absent
target-timestamp = (IUT's local time),
source-device = TD,
-- source-object absent
operation = O,
-- source-comment absent
target-comment = (any valid value, or absent),
invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
source-user-id = (the value from the operation if provided, otherwise absent),
source-user-role = (the value from the operation if provided, otherwise absent),
target-device = IUT,
target-object = (the target object or absent if the target is not an object),
target-property = (the target property or absent if the target is not a property),
target-priority = (the priority supplied, or absent if the target is not a property. shall be 16 or absent if no priority supplied and the target is a property),
target-value = (the target value or absent if no target value for the operation. may be absent if the value size is larger than 32 encoded octets),
current-value = (the value before the op or absent if no target value. may be absent if the value size is larger than 32 encoded octets),
result = (the error reported for the operation)
} )
} ELSE {
BEFORE AR.Maximum_Send_Delay + Notification Fail Time
RECEIVE ConfirmedAuditNotification-Request,
'Notifications' = ({
-- source-timestamp absent
target-timestamp = (IUT's local time),
source-device = TD,
-- source-object absent
operation = (the operation performed),
-- source-comment absent
target-comment = (any valid value, or absent),
invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
source-user-id = (the value from the operation if provided, otherwise absent),
source-user-role = (the value from the operation if provided, otherwise absent),
target-device = IUT,
target-object = (the target object or absent if the target is not an object),
target-property = (the target property or absent if the target is not a property),
target-priority = (the priority supplied, or absent if the target is not a

```
                        property. shall be 16 or absent if no priority supplied and the
                        target is a property),
                    target-value = (the target value or absent if no target value for the
        operation. may be absent if the value size is larger than 32
                        encoded octets),
                    current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                    result = (the error reported for the operation)
                } )
        TRANSMIT BACnet-SimpleACK-PDU
}
```

**7.3.2.X62.10 Target Audit Reporting - GENERAL Operation Test**

Purpose: Verify that GENERAL operation audit notifications contain a Target Comment.

Test Concept: An auditable GENERAL operation, is performed on the IUT. It is verified that an audit notification is sent which contains the error that occurred.

Configuration Requirements: The IUT is configured to report all audit notifications. If the IUT does not generate GENERAL audit notifications, this test shall be skipped.

Test Steps:

```
1. MAKE(make the IUT generate a GENERAL audit notification)
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ({
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = GENERAL,
                    -- source-comment absent
                    target-comment = (any valid value),
                    invoke-id = (the invoke id from the operation, or absent if it was
                        unconfirmed),
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = (the target object or absent if the target is not an object),
                    target-property = (the target property or absent if the target is not a
                        property),
                    target-priority = (the priority supplied, or absent if the target is not a
                        property. shall be 16 or absent if no priority supplied and the
                        target is a property),
                    target-value = (the target value or absent if no target value for the
                        operation. may be absent if the value size is larger than 32
                        encoded octets),
                    current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                    result = (the reason for failure if the op failed, otherwise absent)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
```

```
            'Notifications' = ({
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = GENERAL,
                    -- source-comment absent
                    target-comment = (any valid value),
                    invoke-id = (the invoke id from the operation, or absent if it was
                        unconfirmed),
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = (the target object or absent if the target is not an object),
                    target-property = (the target property or absent if the target is not a
                        property),
                    target-priority = (the priority supplied, or absent if the target is not a
                        property. shall be 16 or absent if no priority supplied and the
                        target is a property),
                    target-value = (the target value or absent if no target value for the
                        operation. may be absent if the value size is larger than 32
                        encoded octets),
                    current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                    result = (the reason for failure if the op failed, otherwise absent)
                } )
        TRANSMIT BACnet-SimpleACK-PDU
}
```

### 7.3.2.X62.11 Source Audit Reporting - Basic Notification Test

Purpose: Verify that source audit notifications are properly formed.

Test Concept: The IUT is made to send a source audit notification. It is verified that the source fields are present, no target fields are present, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report all auditable source operations. The IUT is configured with AR as the source Audit Reporter object.

Test Steps:

```
1. MAKE(the IUT perform an auditable operation, O, on the TD)
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
     BEFORE AR.Maximum_Send_Delay + Notification Fail Time
       RECEIVE UnconfirmedAuditNotification-Request,
         'Notifications' = ( {
           source-timestamp = (IUT's local time),
           -- target-timestamp absent
           source-device = IUT,
           source-object = (the object which initiated the op or absent if not initiated by an
                object),
           operation = O,
           source-comment = (any valid value or absent),
           -- target-comment absent
           invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
           source-user-id = (the value from the operation if provided, otherwise absent),
           source-user-role = (the value from the operation if provided, otherwise absent),
```

target-device = TD,
target-object = (the target object or absent if the target is not an object),
target-property = (the target property or absent if the target is not a property),
target-priority = (the priority supplied, or absent if the target is not a property. shall be 16 or absent if no priority supplied and the target is a property),
target-value = (the target value or absent if no target value for the operation. may be absent if the value size is larger than 32 encoded octets),
current-value = (the value before the op if the op targeted a property, or absent. May be absent even if targeting a property),
result = (the reason for failure if the op failed, otherwise absent)
} )
} ELSE {
BEFORE AR.Maximum_Send_Delay + Notification Fail Time
RECEIVE ConfirmedAuditNotification-Request,
'Notifications' = ( {
source-timestamp = (IUT's local time),
-- target-timestamp absent
source-device = IUT,
source-object = (the object which initiated the op or absent if not initiated by an object),
operation = O,
source-comment = (any valid value or absent),
-- target-comment absent
invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
source-user-id = (the value from the operation if provided, otherwise absent),
source-user-role = (the value from the operation if provided, otherwise absent),
target-device = TD,
target-object = (the target object or absent if the target is not an object),
target-property = (the target property or absent if the target is not a property),
target-priority = (the priority supplied, or absent if the target is not a property. shall be 16 or absent if no priority supplied and the target is a property),
target-value = (the target value or absent if no target value for the operation. may be absent if the value size is larger than 32 encoded octets),
current-value = (the value before the op if the op targeted a property, or absent. May be absent even if targeting a property),
result = (the reason for failure if the op failed, otherwise absent)
} )
TRANSMIT BACnet-SimpleACK-PDU
}

**7.3.2.X62.12 Source Audit Reporting - Same Device Notification Test**

Purpose: Verify that source and target fields are in audit notifications are performed by the IUT on the IUT.

Test Concept: The IUT is made to perform an auditable operation on itself. It is verified that the source and target fields are present, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report all auditable operations. The IUT is configured with AR as the source Audit Reporter object. If the IUT is unable to perform an auditable operation on itself, this test shall be skipped.

Test Steps:
1. MAKE(the IUT perform an auditable operation, O, on the itself)
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
BEFORE AR.Maximum_Send_Delay + Notification Fail Time
RECEIVE UnconfirmedAuditNotification-Request,

```
            'Notifications' = ( {
                source-timestamp = (T: IUT's local time),
                target-timestamp = T,
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                        object),
                operation = O,
                source-comment = (any valid value or absent),
                target-comment = (any valid value or absent),
                -- invoke-id absent
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = IUT,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                        shall be 16 or absent if no priority supplied and the target is a
                        property),
                target-value = (the target value or absent if no target value for the operation.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (the value before the op if the op targeted a property, or absent),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (T: IUT's local time),
                target-timestamp = T,
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                        object),
                operation = O,
                source-comment = (any valid value or absent),
                target-comment = (any valid value or absent),
                -- invoke-id absent
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = IUT,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                        shall be 16 or absent if no priority supplied and the target is a
                        property),
                target-value = (the target value or absent if no target value for the operation.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (the value before the op if the op targeted a property, or absent),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
    TRANSMIT BACnet-SimpleACK-PDU
}
```

Notes to Tester: The IUT is allowed to send the notifications as 2 separate notifications in the same audit notification message, or in separate messages. When sending separate notifications, one shall be a correctly formed target notification and the other a correctly formed source notification for the operation performed.

**7.3.2.X62.13 Source Audit Reporting - Unconfirmed Service Operation Test**

Purpose: Verify that source audit notifications for unconfirmed services do not contain InvokeId information.

Test Concept: An auditable unconfirmed service is performed by the IUT and it is verified that the resulting source audit notification does not contain an InvokeId, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report source audit notifications for an unconfirmed service. If the IUT does not support source audit reporting for any unconfirmed services, this test shall be skipped. The IUT is configured with AR as the source Audit Reporter object.

Test Steps:
1. MAKE(the IUT perform an auditable operation, O, on the TD which uses an unconfirmed service)
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                    object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                -- invoke-id absent
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = TD,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                    shall be 16 or absent if no priority supplied and the target is a
                    property),
                target-value = (the target value or absent if no target value for the operation.
                    may be absent if the value size is larger than 32 encoded
                    octets),
                current-value = (the value before the op if the op targeted a property, or absent.
                    May be absent even if targeting a property),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                    object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                -- invoke-id absent
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = TD,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                    shall be 16 or absent if no priority supplied and the target is a
                    property),
                target-value = (the target value or absent if no target value for the operation.

```
                    may be absent if the value size is larger than 32 encoded
                    octets),
                current-value = (the value before the op if the op targeted a property, or absent.
                    May be absent even if targeting a property),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
    TRANSMIT BACnet-SimpleACK-PDU
}
```

**7.3.2.X62.14 Source Audit Reporting - Confirmed Service Operation Audit Notification**

Purpose: Verify that source audit notifications for confirmed services contain InvokeId information.

Test Concept: An auditable confirmed service is performed by the IUT and it is verified that the resulting source audit notification contains an InvokeId, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report source audit notifications for a confirmed service. If the IUT does not support source audit reporting for any confirmed services, this test shall be skipped. The IUT is configured with AR as the source Audit Reporter object.

Test Steps:

```
1. MAKE(the IUT perform an auditable operation, O, on the TD which uses an confirmed service)
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                    object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                invoke-id = (the Invoke Id from the operation),
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = TD,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                    shall be 16 or absent if no priority supplied and the target is a
                    property),
                target-value = (the target value or absent if no target value for the operation.
                    may be absent if the value size is larger than 32 encoded
                    octets),
                current-value = (the value before the op if the op targeted a property, or absent.
                    May be absent even if targeting a property),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                    object),
                operation = O,
```

```
                    source-comment = (any valid value or absent),
                    -- target-comment absent
                    invoke-id = (the Invoke Id from the operation),
                    source-user-id = (the value from the operation if provided, otherwise absent),
                    source-user-role = (the value from the operation if provided, otherwise absent),
                    target-device = TD,
                    target-object = (the target object or absent if the target is not an object),
                    target-property = (the target property or absent if the target is not a property),
                    target-priority = (the priority supplied, or absent if the target is not a property.
                            shall be 16 or absent if no priority supplied and the target is a
                            property),
                    target-value = (the target value or absent if no target value for the operation.
                            may be absent if the value size is larger than 32 encoded
                            octets),
                    current-value = (the value before the op if the op targeted a property, or absent.
                            May be absent even if targeting a property),
                    result = (the reason for failure if the op failed, otherwise absent)
                    } )
    TRANSMIT BACnet-SimpleACK-PDU
}
```

**7.3.2.X62.15 Source Audit Reporting - Operations with Priority Test**

Purpose: Verify that source audit notifications which for writes which convey a priority include the priority in the notification.

Test Concept: An auditable write, which includes a priority, is performed on a commandable object by the IUT and it is verified that the resulting source audit notification contains a priority, and other notification fields represent the auditable operation performed.

Configuration Requirements: The IUT is configured to report audit notifications for writes on a commandable object, O1. The IUT is configured with AR as the source Audit Reporter object. If the IUT does not provide priorities in auditable operations it performed, this test shall be skipped.

Test Steps:

```
1. MAKE(the IUT perform an auditable operation in containing a priority, other than 16, on the TD)
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                        object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                invoke-id = (the Invoke Id from the operation),
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = TD,
                target-object = (the target object),
                target-property = (the target property),
                target-priority = (the priority supplied),
                target-value = (the target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (any valid value, or absent),
                result = (the reason for failure if the op failed, otherwise absent)
```

```
                    } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                        object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                invoke-id = (the Invoke Id from the operation),
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = TD,
                target-object = (the target object),
                target-property = (the target property),
                target-priority = (the priority supplied),
                target-value = (the target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (any valid value, or absent),
                result = (the reason for failure if the op failed, otherwise absent)
                } )
    TRANSMIT BACnet-SimpleACK-PDU

}
```

**7.3.2.X62.16 Source Audit Reporting - Error Audit Notification Test**

Purpose: Verify that operations performed by the IUT which fail are properly reported in audit notifications.

Test Concept: The IUT is made to perform an auditable operation on the TD and the TD returns an Error-PDU. It is verified that a source audit notification is sent which contains the error that occurred. This is repeated twice more with the TD returning a Reject PDU, and then an Abort PDU.

Configuration Requirements: The IUT is configured to report all audit notifications. The IUT is configured with AR as the source Audit Reporter object.

Test Steps:

```
-- Error-PDU
1. MAKE(the IUT perform an auditable operation on TD which will fail (via return of a BACnetErrorPDU,
    or a Result(+) with error information)
2. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not
                        initiated by an object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                invoke-id = (the invoke id from the operation, or absent if it was
                        unconfirmed),
                source-user-id = (the value from the operation if provided, otherwise
                        absent),
                source-user-role = (the value from the operation if provided, otherwise
```

```
                    absent),
                target-device = TD,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a
                    property),
                target-priority = (the priority supplied, or absent if the target is not a
                    property. shall be 16 or absent if no priority supplied and the
                    target is a property),
                target-value = (the target value or absent if no target value for the
                    operation. may be absent if the value size is larger than 32
                    encoded octets),
                current-value = (the value before the op if the op targeted a property, or
                    absent. May be absent even if targeting a property),
                result = (the error reported for the operation)
            } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not
                    initiated by an object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                invoke-id = (the invoke id from the operation, or absent if it was
                    unconfirmed),
                source-user-id = (the value from the operation if provided, otherwise
                    absent),
                source-user-role = (the value from the operation if provided, otherwise
                    absent),
                target-device = TD,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a
                    property),
                target-priority = (the priority supplied, or absent if the target is not a
                    property. shall be 16 or absent if no priority supplied and the
                    target is a property),
                target-value = (the target value or absent if no target value for the
                    operation. may be absent if the value size is larger than 32
                    encoded octets),
                current-value = (the value before the op if the op targeted a property, or
                    absent. May be absent even if targeting a property),
                result = (the error reported for the operation)
            } )
    TRANSMIT BACnet-SimpleACK-PDU
}

-- Reject-PDU
3. MAKE(the IUT perform an auditable operation on TD which will fail (via return of a BACnetRejectPDU))
4. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
```

```
                        object),
                    operation = O,
                    source-comment = (any valid value or absent),
                    -- target-comment absent
                    invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
                    source-user-id = (the value from the operation if provided, otherwise absent),
                    source-user-role = (the value from the operation if provided, otherwise absent),
                    target-device = TD,
                    target-object = (the target object or absent if the target is not an object),
                    target-property = (the target property or absent if the target is not a property),
                    target-priority = (the priority supplied, or absent if the target is not a property.
                        shall be 16 or absent if no priority supplied and the target is a
                        property),
                    target-value = (the target value or absent if no target value for the operation.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                    current-value = (the value before the op if the op targeted a property, or absent.
                        May be absent even if targeting a property),
                    result = (the error reported for the operation)
                } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                        object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = TD,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                        shall be 16 or absent if no priority supplied and the target is a
                        property),
                target-value = (the target value or absent if no target value for the operation.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                current-value = (the value before the op if the op targeted a property, or absent.
                        May be absent even if targeting a property),
                result = (the error reported for the operation)
            } )  TRANSMIT BACnet-SimpleACK-PDU
}

-- Abort-PDU
3. MAKE(the IUT perform an auditable operation on TD which will fail (via return of a BACnetAbortPDU))
4. IF the IUT is configured to send unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
```

```
                    object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = TD,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                    shall be 16 or absent if no priority supplied and the target is a
                    property),
                target-value = (the target value or absent if no target value for the operation.
                    may be absent if the value size is larger than 32 encoded
                    octets),
                current-value = (the value before the op if the op targeted a property, or absent.
                    May be absent even if targeting a property),
                result = (the error reported for the operation)
            } )
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
                source-timestamp = (IUT's local time),
                -- target-timestamp absent
                source-device = IUT,
                source-object = (the object which initiated the op or absent if not initiated by an
                    object),
                operation = O,
                source-comment = (any valid value or absent),
                -- target-comment absent
                invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
                source-user-id = (the value from the operation if provided, otherwise absent),
                source-user-role = (the value from the operation if provided, otherwise absent),
                target-device = TD,
                target-object = (the target object or absent if the target is not an object),
                target-property = (the target property or absent if the target is not a property),
                target-priority = (the priority supplied, or absent if the target is not a property.
                    shall be 16 or absent if no priority supplied and the target is a
                    property),
                target-value = (the target value or absent if no target value for the operation.
                    may be absent if the value size is larger than 32 encoded
                    octets),
                current-value = (the value before the op if the op targeted a property, or absent.
                    May be absent even if targeting a property),
                result = (the error reported for the operation)
            } )
    TRANSMIT BACnet-SimpleACK-PDU
}
```

**7.3.2.X62.17 Source Audit Reporting - Single Source Audit Reporter Object Test**
Purpose: Verify that the IUT contains a single Audit Report for source audit reporting.

Test Concept: Check all Audit Reporter objects in the IUT and verify that only one is for source audit reporting.

Test Steps:
1. SourceAR = NONE
2. REPEAT AR = (each Audit Reporter object) DO {
    IF AR.Audit_Source_Reporter is TRUE THEN

```
        IF SourceAR is not NONE THEN
            ERROR "Multiple Audit Reporter objects setup for source reporting."
}
3. CHECK(SourceAR is not NONE)
```

**7.3.2.X62.18 Audit Forwarding Test**

Purpose: Verify that the IUT forwards received audit notifications.

Test Concept: Send a sequence of confirmed and unconfirmed, unicast and broadcast audit notifications to the IUT and verify they are forwarded.

Configuration Requirements: The IUT is configured with an Audit Log, AL, which is setup to forward and delete audit notifications with no delay. The IUT is configured to send notifications unconfirmed.

Test Steps:

```
-- Unicast confirmed
1. TRANSMIT ConfirmedAuditNotification-Request,
      'Notifications' = (N1: a list of 1 or more notifications)
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE Notification Fail Time
      RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = N1
4. TRANSMIT ReadRange-Request,
      'Object Identifier' = AL,
      'Property Identifier' = Log_Buffer,
      'Reference Index' = 1,
      'Count' = 10
5. RECEIVE ReadRange-Ack,
      'Object Identifier' = AL,
      'Property Identifier' = Log_Buffer,
      'Result Flags' = (False, False, False),
      'Count' = 0

-- Unicast unconfirmed
6. TRANSMIT UnconfirmedAuditNotification-Request,
      'Notifications' = (N2: a list of 1 or more notifications)
7. BEFORE Notification Fail Time
      RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = N2
8. TRANSMIT ReadRange-Request,
      'Object Identifier' = AL,
      'Property Identifier' = Log_Buffer,
      'Reference Index' = 1,
      'Count' = 10
9. RECEIVE ReadRange-Ack,
      'Object Identifier' = AL,
      'Property Identifier' = Log_Buffer,
      'Result Flags' = (False, False, False),
      'Count' = 0

-- Local Broadcast
10. TRANSMIT UnconfirmedAuditNotification-Request,
      DESTINATION = LOCAL BROADCAST,
      'Notifications' = (N3: a list of 1 or more notifications)
11. BEFORE Notification Fail Time
      RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = N3
12. TRANSMIT ReadRange-Request,
      'Object Identifier' = AL,
```

'Property Identifier' = Log_Buffer,
'Reference Index' = 1,
'Count' = 10
13. RECEIVE ReadRange-Ack,
'Object Identifier' = AL,
'Property Identifier' = Log_Buffer,
'Result Flags' = (False, False, False),
'Count' = 0
-- Global Broadcast
10. TRANSMIT UnconfirmedAuditNotification-Request,
DESTINATION = GLOBAL BROADCAST,
'Notifications' = (N4: a list of 1 or more notifications)
11. BEFORE Notification Fail Time
RECEIVE UnconfirmedAuditNotification-Request,
'Notifications' = N4
12. TRANSMIT ReadRange-Request,
'Object Identifier' = AL,
'Property Identifier' = Log_Buffer,
'Reference Index' = 1,
'Count' = 10
13. RECEIVE ReadRange-Ack,
'Object Identifier' = AL,
'Property Identifier' = Log_Buffer,
'Result Flags' = (False, False, False),
'Count' = 0

[Add AuditLogQuery initiation tests into BTL Specified Tests]

**8.X33.1 Reading a Range of Items Using Any Valid Query**

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

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

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

Test Steps:

1. MAKE (L1 contain the set of records S1)
2. MAKE (the IUT request a range of samples from L1) {
3. WHILE (not all records from the tester-selected range have been returned)
    RECEIVE AuditLogQuery-Request,
        'Audit Log' = (L1),
        'Query Parameters' = (any valid query),
        'Start at Sequence Number' = (an appropriate sequence number)  -- or absent
        'Requested Count' = (any valid range)
    TRANSMIT AuditLogQuery-ACK,
        'AuditLog' = (L1),
        'Records' = (a set of records appropriate for this response),
        'NoMoreItems' = (an appropriate value for this response)
}
4. CHECK (the records returned in step 3 include the tester-selected range)
5. MAKE (the IUT display the tester-selected range)
6. CHECK (the records displayed in step 5 are consistent with the records returned in step 3)

[Add AuditLogQuery Execution Tests into BTL Specified Tests]

**9.X33 AuditLogQuery Service Execution Tests**

**9.X33 AuditLogQuery Service Positive Tests**

**9.X33.1.1 - AuditLogQuery By Target Test**

Purpose: Verify that an Audit Log correctly returns notifications filtered by a specific target.

Test Concept: An Audit Log, O1, containing a sequence of notifications related to a set of audit targets is queried about a specific target, T1. The query is repeated for each standard form of target based query and the result is compared against the expected set of returned notifications.

Configuration Requirements: The Audit Log, O1, contains a set of audit notifications which contains 0 or more notifications about audit target T1.

Test Steps:

```
1. REPEAT Q = (each query in (
                By Target Device Identifier,
                By Target Device Identifier & Target Device Address,
                By Target Device Identifier & Target Object Identifier,
                By Target Device Identifier & Target Property Identifier,
                By Target Device Identifier & Target Array Index,
                By Target Device Identifier & Target Priority,
                By Target Device Identifier & Operations,
                By Target Device Identifier & Result Filter (failed),
                By Target Device Identifier & Result Filter (success),
                By Target Device Identifier & Target Object Identifier & Target Property Identifier
                ) {
     TRANSMIT AuditLogQuery-Request,
          'Audit Log' =       O1,
          'Query Parameters' =  Q,
          'Requested Count' =    Total_Record_Count
     RECEIVE AuditLogQuery-ACK,
          'Audit Log' =       O1,
          Records' =          (RSEQ: a set of records),
     WHILE (the length of RSEQ is not the number of expected records) {
          TRANSMIT AuditLogQuery-Request,
               'Audit Log' =       O1,
               'Query Parameters' =  Q,
               'Start At Sequence Number' =   (the 'Sequence Number' from the last entry in RSEQ)
               'Requested Count' =    Total_Record_Count
          RECEIVE AuditLogQuery-ACK,
               'Audit Log' =       O1,
               Records' =          (NXTSEQ: a set of records),
          IF (the length of NXTSEQ is 0) THEN
               ERROR "expected more records from the Audit Log"
          RSEQ = (RSEQ record with NXTSEQ records appended)
     }
     CHECK(that the records in RSEQ is the set expected and are returned in sequence number order)
}
```

**9.X33.1.2 - AuditLogQuery By Source Test**

Purpose: Verify that an Audit Log correctly returns notifications filtered by a specific source.

Test Concept: An Audit Log, O1, containing a sequence of notifications related to a set of audit sources is queried about a specific source, S1. The query is repeated for each standard form of source based query and the result is compared against the expected set of returned notifications.

Configuration Requirements: The Audit Log, O1, contains a set of audit notifications which contains 0 or more notifications about audit source S1.

Test Steps:
1. REPEAT Q = (each query in (
By Source Device Identifier,
By Source Device Identifier & Source Device Address,
By Source Device Identifier & Source Object Identifier,
By Source Device Identifier & Operations,
By Source Device Identifier & Result Filter (failed),
By Source Device Identifier & Result Filter (success),
) {
TRANSMIT AuditLogQuery-Request,
'Audit Log' = O1,
'Query Parameters' = Q,
'Requested Count' = Total_Record_Count
RECEIVE AuditLogQuery-ACK,
'Audit Log' = O1,
'Records' = (RSEQ: a set of records),
WHILE (the length of RSEQ is not the number of expected records) {
TRANSMIT AuditLogQuery-Request,
'Audit Log' = O1,
'Query Parameters' = Q,
'Start At Sequence Number' = (the 'Sequence Number' from the last entry in RSEQ)
'Requested Count' = Total_Record_Count
RECEIVE AuditLogQuery-ACK,
'Audit Log' = O1,
'Records' = (NXTSEQ: a set of records),
IF (the length of NXTSEQ is 0) THEN
ERROR "expected more records from the Audit Log"
RSEQ = (RSEQ record with NXTSEQ records appended)
}
CHECK(that the records in RSEQ is the set expected and are returned in sequence number order)
}

**9.X33.2 - AuditLogQuery Negative Tests**

**9.X33.2.1 - Attempting to Query a Non-existent Audit Log**

Purpose: Verify that the correct error is returned when the queried log does not exist.

Test Concept: Send an AuditLogQuery request to the IUT for an AuditLog object which does not exist. Verify that the IUT returns and error class of OBJECT and an error code of UNKNOWN_OBJECT.

Test Steps:

1. TRANSMIT AuditLogQuery-Request,
'Audit Log' = (an audit log not in the IUT),
'Query Parameters' = (any valid value),
'Requested Count' = (any valid value)
2. RECEIVE BACnet-Error PDU,
'Error Class' = OBJECT,
'Error Code' = UNKNOWN_OBJECT

[Add Audit_Level Property Tests into BTL Specified Tests]

**7.3.1.X498 Audit_Level Property Tests**

**7.3.1.X498.1 Object Specific Configurable Audit_Level NONE Test**

Purpose: Verify that the Audit_Level property, in an auditable object, controls the audit level for the object.

Test Concept: An object, O1, is selected which supports a configurable Audit_Level property. With O1 configured to report notifications, Audit_Level is changed to NONE. An auditable action is performed on O1 and it is verified that no notification is generated.

Audit_Level is then changed to AUDIT_CONFIG, and an auditable config action is performed on the object. It is verified that a notification is sent. An auditable non-config action is performed on the object and it is verified that no notification is sent.

Audit_Level is then changed to AUDIT_ALL and an auditable action is performed on the object. It is verified that a notification is sent.

Configuration Requirements: The IUT is configured to generate audit notifications. The selected object, O1, shall have auditable configuration operations that can be applied to it. AR is the Audit Reporter object configured to report for O1.

Test Steps:

1. WRITE O1, Audit_Level = NONE
2. WRITE AR, Audit_Level = (AUDIT_CONFIG or AUDIT_ALL)
3. MAKE(perform an operation on O1 that would be reported by the AR if O1.Audit_Level were AUDIT_ALL)
4. WAIT(AR.Maximum_Send_Delay + Notification Fail Time)
5. CHECK(that the IUT did not report an audit notification for the operation)

6. IF (O1 has auditable configuration operations, such as writable configuration properties) THEN {
    WRITE O1, Audit_Level = AUDIT_CONFIG
    WRITE AR, Audit_Level = NONE
    MAKE(perform a config operation on O1)
    IF the IUT is configured to generate unconfirmed audit notifications THEN {
        BEFORE AR.Maximum_Send_Delay + Notification Fail Time
            RECEIVE UnconfirmedAuditNotification-Request,
                'Notifications' = (a notification of the operation performed)
    } ELSE {
        BEFORE AR.Maximum_Send_Delay + Notification Fail Time
            RECEIVE ConfirmedAuditNotification-Request,
                'Notifications' = (a notification of the operation performed)
        TRANSMIT BACnet-SimpleACK-PDU
    }
    MAKE(perform an auditable non-config operation on O1)
    WAIT AR.Maximum_Send_Delay + Notification Fail Time
    CHECK(that the IUT did not report an audit notification for the operation)
}

7. WRITE O1, Audit_Level = AUDIT_ALL
8. WRITE AR, Audit_Level = NONE
9. MAKE(perform an auditable operation on O1)
10. IF the IUT is configured to generate unconfirmed audit notifications THEN {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = (a notification of the operation performed)
} ELSE {
    BEFORE AR.Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = (a notification of the operation performed)

```
    TRANSMIT BACnet-SimpleACK-PDU
}
```

Notes to Tester: In this test, the audit reporting configuration properties are assumed to be writable. If one is only configurable, then the WRITE operation should be replaced with MAKE.

**7.3.1.X498.2 Audit Reporter Audit_Level Test**

Purpose: Verify that the Audit_Reporter's Audit_Level property is used in objects without an Audit_Level, or when an object's Audit_Level is DEFAULT.

Test Concept: An object, O1, is selected which supports a configurable Audit_Level property, or which does not have an Audit_Level property. The Audit_Reporter for O1 is referred to as AR1. If O1 has an Audit_Level property, it is set to DEFAULT.

With O1 configured to report notifications, AR1's Audit_Level is changed to NONE. An auditable action is performed on O1 and it is verified that no notification is generated.

AR1's Audit_Level is then changed to AUDIT_CONFIG, and an auditable config action is performed on O1. It is verified that a notification is sent. An auditable non-config action is performed on O1 and it is verified that no notification is sent.

Audit_Level is then changed to AUDIT_ALL and an auditable action is performed on O1. It is verified that a notification is sent.

Configuration Requirements: The IUT is configured to generate audit notifications. The selected object, O1, should have auditable configuration operations and auditable non-configuration operations that can be applied to it. AR is the Audit Reporter object configured to report for O1.

Test Steps:

```
1. IF O1 contains an Audit_Level property THEN
      WRITE O1, Audit_Level = DEFAULT
2. WRITE AR.Audit_Level = NONE
3. MAKE(perform an operation on O1 that would be reported by the AR if O1.Audit_Level were AUDIT_ALL)
4. WAIT AR.Maximum_Send_Delay + Notification Fail Time
5. CHECK(that the IUT did not report an audit notification for the operation)

6. WRITE AR.Audit_Level = AUDIT_CONFIG
7. IF O1 supports auditable config operations THEN {
      MAKE(perform a config operation on O1)
      IF the IUT is configured to generate unconfirmed audit notifications THEN {
            BEFORE AR.Maximum_Send_Delay + Notification Fail Time
                  RECEIVE UnconfirmedAuditNotification-Request,
                        'Notifications' =   (a notification of the operation performed)
      } ELSE {
            BEFORE AR.Maximum_Send_Delay + Notification Fail Time
                  RECEIVE ConfirmedAuditNotification-Request,
                        'Notifications' =   (a notification of the operation performed)
            TRANSMIT BACnet-SimpleACK-PDU
      }
}
8. IF O1 supports auditable config operations THEN {
      MAKE(perform a non-config operation on O1)
      WAIT AR.Maximum_Send_Delay + Notification Fail Time
       CHECK(that the IUT did not report an audit notification for the operation)
}

9. WRITE AR.Audit_Level = AUDIT_ALL
10. IF O1 supports auditable config operations THEN {
      MAKE(perform a config operation on O1)
```

```
    IF the IUT is configured to generate unconfirmed audit notifications THEN {
        BEFORE AR.Maximum_Send_Delay + Notification Fail Time
            RECEIVE UnconfirmedAuditNotification-Request,
                'Notifications' =   (a notification of the operation performed)
    } ELSE {
        BEFORE AR.Maximum_Send_Delay + Notification Fail Time
            RECEIVE ConfirmedAuditNotification-Request,
                'Notifications' =   (a notification of the operation performed)
        TRANSMIT BACnet-SimpleACK-PDU
    }
}
11. IF O1 supports auditable config operations THEN {
    MAKE(perform a non-config operation on O1)
    IF the IUT is configured to generate unconfirmed audit notifications THEN {
        BEFORE AR.Maximum_Send_Delay + Notification Fail Time
            RECEIVE UnconfirmedAuditNotification-Request,
                'Notifications' =   (a notification of the operation performed)
    } ELSE {
        BEFORE AR.Maximum_Send_Delay + Notification Fail Time
            RECEIVE ConfirmedAuditNotification-Request,
                'Notifications' =   (a notification of the operation performed)
        TRANSMIT BACnet-SimpleACK-PDU
    }
}
```

Notes to Tester: In this test, the audit reporting configuration properties are assumed to be writable. If one is only configurable, then the WRITE operation should be replaced with MAKE.

**7.3.1.X498.3 Audit_Level Change Notification Test**

Purpose: Verify that changes to Audit_Level property results in audit notifications.

Test Concept: An object, O1, is selected which supports a writable and/or configurable Audit_Level property. The Audit_Level property is written to each valid audit level and a notification of the change is checked for. The Audit_Level property is then configured (not via BACnet writes) to each valid audit level and a notification of the change is checked for.

Configuration Requirements: The IUT is configured to generate audit notifications and O1's Audit_Level property is set to NONE.

Test Steps:

```
1. IF Audit_Level is writable THEN {
    REPEAT AL = ( AUDIT_CONFIG, AUDIT_ALL, DEFAULT, NONE ) DO {
        IF O1 is an Audit Reporter and AL is DEFAULT THEN {
            -- don't test DEFAULT on Audit Report object
        } ELSE {
            WRITE O1, Audit = AL
            IF the IUT is configured to send unconfirmed audit notifications THEN {
                BEFORE AR.Maximum_Send_Delay + Notification Fail Time
                    RECEIVE UnconfirmedAuditNotification-Request,
                        'Notifications' =   (a notification indicating change of
                            Audit_Level)
            } ELSE {
                BEFORE AR.Maximum_Send_Delay + Notification Fail Time
                    RECEIVE ConfirmedAuditNotification-Request,
                        'Notifications' =   (a notification indicating change of
                            Audit_Level)
                TRANSMIT BACnet-SimpleACK-PDU
            }
        }
```

```
    }
}

2. IF Audit_Level is changeable without writing it via BACnet THEN {
    REPEAT AL = ( AUDIT_CONFIG, AUDIT_ALL, DEFAULT, NONE ) DO {
        IF O1 is an Audit Reporter and AL is DEFAULT THEN {
            -- don't test DEFAULT on Audit Report object
        } ELSE {
            MAKE(O1, Audit = AL without using BACnet services to write the property)
            IF the IUT is configured to send unconfirmed audit notifications THEN {
                BEFORE AR.Maximum_Send_Delay + Notification Fail Time
                    RECEIVE UnconfirmedAuditNotification-Request,
                        'Notifications' =   (a notification indicating change of
                            Audit_Level)
            } ELSE {
                BEFORE AR.Maximum_Send_Delay + Notification Fail Time
                    RECEIVE ConfirmedAuditNotification-Request,
                        'Notifications' =   (a notification indicating change of
                            Audit_Level)
                TRANSMIT BACnet-SimpleACK-PDU
            }
        }
    }
}
```

[Add Audit_Notification_Recipient property tests into BTL Specified Tests]

**7.3.1.X499 Audit_Notification_Recipient Property Tests**

**7.3.1.X499.1 Audit_Notification_Recipient Test**

Purpose: Verify that an Audit_Notification_Recipient accepts and correctly uses all forms of recipient addresses.

Test Concept: With an object configured to report notifications, the Device object's Audit_Notification_Recipient property is set to a local broadcast. An action is performed on the IUT which should result in an audit notification, and it is verified that the notification is locally broadcast. This is repeated for global broadcasts, and unicast recipients. Only the unicast form is tested for devices which do not generate UnconfirmedAuditNotifications.

Configuration Requirements: The IUT is configured to report audit notifications. If the IUT supports sending unconfirmed audit notifications, it shall be configured to do so. Audit_Notification_Recipient is configured to be the TD, and audit reporting is enabled in the IUT.

Test Steps:

-- Local Broadcast Recipient
1. IF the IUT supports sending unconfirmed audit notifications THEN {
    WRITE Audit_Notification_Recipient = (local broadcast recipient)
    BEFORE Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' =  (a notification for the change to Audit_Notification_Recipient)
        |
        (UnconfirmedAuditNotification-Request
            DESTINATION =  GLOBAL BROADCAST,
            'Notifications' =  (a notification for the change to
                    Audit_Notification_Recipient)
        )
    IF the first notification was not globally broadcast) THEN {
        RECEIVE UnconfirmedAuditNotification-Request,
            DESTINATION =  LOCAL BROADCAST,
            'Notifications' =  (a notification for the change to Audit_Notification_Recipient)
    }
    MAKE(perform an operation on the IUT which will result in an audit notification, other than changing the
        Audit_Notification_Recipient property)
    BEFORE Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            DESTINATION =  LOCAL BROADCAST,
            'Notifications' =  (a notification correctly indicating the operation performed)
}

-- Global Broadcast Recipient
2. IF the IUT supports sending unconfirmed audit notifications THEN {
    WRITE Audit_Notification_Recipient = (global broadcast recipient)
    BEFORE Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            DESTINATION =  LOCAL BROADCAST,
            'Notifications' =  (a notification for the change to Audit_Notification_Recipient)
        |
        (UnconfirmedAuditNotification-Request
            DESTINATION =  GLOBAL BROADCAST,
            'Notifications' =  (a notification for the change to
                    Audit_Notification_Recipient)
        )
    IF (the first notification was not globally broadcast) THEN {
        RECEIVE UnconfirmedAuditNotification-Request,
            DESTINATION =  GLOBAL BROADCAST,

```
            'Notifications' =       (a notification for the change to Audit_Notification_Recipient)
    }
    MAKE(perform an operation on the IUT which will result in an audit notification, other than changing the
        Audit_Notification_Recipient property)
    BEFORE Maximum_Send_Delay + Notification Fail Time
        RECEIVE UnconfirmedAuditNotification-Request,
            DESTINATION =   GLOBAL BROADCAST,
            'Notifications' =       (a notification correctly indicating the operation performed)
}

-- Unicast Recipient
3. WRITE Audit_Notification_Recipient = (D1: a device other than the TD)
4. IF the IUT is configured to send unconfirmed notifications  THEN {
    BEFORE (Maximum_Send_Delay plus maximum time to resolve TD)
        RECEIVE UnconfirmedAuditNotification-Request,
            DESTINATION =   TD,
            'Notifications' =       (a notification for the change to Audit_Notification_Recipient)
        | (UnconfirmedAuditNotification-Request
            DESTINATION =   GLOBAL BROADCAST,
            'Notifications' =       (a notification for the change to Audit_Notification_Recipient)
        )
} ELSE {
    BEFORE (Maximum_Send_Delay plus maximum time to resolve TD)
        RECEIVE ConfirmedAuditNotification-Request,
            DESTINATION =   D1,
            'Notifications' =       (a notification for the change to Audit_Notification_Recipient)
    TRANSMIT BACnet-SimpleACK-PDU,
        SOURCE =            D1
}
5. IF (the first notification was not globally broadcast) THEN
    RECEIVE UnconfirmedAuditNotification-Request,
        DESTINATION =   TD,
        'Notifications' =       (a notification for the change to Audit_Notification_Recipient)

6. MAKE(perform an operation on the IUT which will result in an audit notification, other than changing the
        Audit_Notification_Recipient property)
7. BEFORE Maximum_Send_Delay
    RECEIVE UnconfirmedAuditNotification-Request,
        DESTINATION =   TD,
        'Notifications' =       (a notification correctly indicating the operation performed)
```

Notes to Tester: Where the IUT is expected to send multiple audit notifications for an operation, the notifications can be generated in any order.

[Add Audit_Priority_Filter property tests into BTL Specified Tests]

**7.3.1.X500.1 Audit_Priority_Filter Target Audit Reporting Test**
Purpose: Verify that Audit_Priority_Filter correctly filters the generation of audit notifications based on priority.

Test Concept: An auditable commandable object for which Audit_Priority_Filter filtering can be applied is configured to report all write operations. The Audit_Priority_Filter is configured to restrict audit notifications to all but a single priority X. The object is commanded at a priority other than X and it is verified that an audit notification is sent. The object is then commanded at priority X and it is verified that no audit notification is sent.

Configuration Requirements: The IUT is configured to report audit notifications for writes on a commandable object, O1. If the IUT does not support the Priority_Array property in any object for which audit reporting can be configured, or if the IUT does not support a configurable Audit_Priority_Filter property, this test shall be skipped. AR shall be the Audit Reporter object for O1.

Test Steps:
-- Test the object's Audit_Priority_Filter
1. IF O1 supports a configurable Audit_Priority_Filter property THEN {
WRITE O1, Audit_Priority_Filter = { all ones except priority X (bit X-1) }
TRANSMIT WriteProperty-Request,
'Invoke Id' = I,
'Object Identifier' = O1,
'Property Identifier' = Present_Value,
'Property Value' = (V: any valid value),
'Priority' = (PRIO: any valid priority, but not X)
BEFORE Internal Processing Fail Time
RECEIVE BACnet-SimpleACK-PDU
|
(BACnet-Error-PDU,
'Error Type' = (E: any error)
)
IF the IUT is configured to send unconfirmed audit notifications THEN {
BEFORE AR.Maximum_Send_Delay + Notification Fail Time
RECEIVE UnconfirmedAuditNotification-Request,
'Notifications' = ({ -- there may be a second notification included for the write to
-- Audit_Priority_Filter. If there is, the order of the
-- notifications in the list is not relevant
-- source-timestamp absent
target-timestamp = (IUT's local time),
source-device = TD,
-- source-object absent
operation = WRITE,
-- source-comment absent
target-comment = (any valid value, or absent),
invoke-id = (the invoke ID of the write to Present_Value),
source-user-id = (the value from the operation if provided, otherwise absent),
source-user-role = (the value from the operation if provided, otherwise absent),
target-device = IUT,
target-object = O1,
target-property = Present_Value,
target-priority = PRIO,
target-value = V,
current-value = (the value before the write. may be absent if the value size is
larger than 32 encoded octets),
result = (E, if the op failed, otherwise absent)
} )
} ELSE {
BEFORE AR.Maximum_Send_Delay + Notification Fail Time
RECEIVE ConfirmedAuditNotification-Request,
'Notifications' = ({ -- there may be a second notification included for the write to

```
                                -- Audit_Priority_Filter. If there is, the order of the
                        -- notifications in the list is not relevant
                            -- source-timestamp absent
                            target-timestamp = (IUT's local time),
                            source-device = TD,
                            -- source-object absent
                            operation = WRITE,
                            -- source-comment absent
                            target-comment = (any valid value, or absent),
                            invoke-id = (the invoke ID of the write to Present_Value),
                            source-user-id = (the value from the operation if provided, otherwise absent),
                            source-user-role = (the value from the operation if provided, otherwise absent),
                            target-device = IUT,
                            target-object = O1,
                            target-property = Present_Value,
                            target-priority = PRIO,
                            target-value = V,
                            current-value = (the value before the write. may be absent if the value size is
                                    larger than 32 encoded octets),
                            result = (E, if the op failed, otherwise absent)
                        } )
            TRANSMIT BACnet-SimpleACK-PDU
        }
        TRANSMIT WriteProperty-Request,
            'Invoke Id' =       I,
            'Object Identifier' =   O1,
            'Property Identifier' = Present_Value,
            'Property Value' =      (V: any valid value),
            'Priority' =        (PRIO: X)
        BEFORE Internal Processing Fail Time
            RECEIVE BACnet-SimpleACK-PDU
                |
                (BACnet-Error-PDU,
                    'Error Type' = (E: any error)
                )
        WAIT AR.Maximum_Send_Delay + Notification Fail Time
        CHECK(no audit notification sent)
}
-- Test the Audit Reporter object's Audit_Priority_Filter
2. IF AR supports a configurable Audit_Priority_Filter property and O1 has no Audit_Priority_Filter or it can be configured
to None THEN {
        IF O1 has an Audit_Priority_Filter THEN
            WRITE O1, Audit_Priority_Filter = NONE
        WRITE AR, Audit_Priority_Filter = { all ones except priority X (bit X-1) }
        TRANSMIT WriteProperty-Request,
            'Invoke Id' =       I,
            'Object Identifier' =   O1,
            'Property Identifier' = Present_Value,
            'Property Value' =      (V: any valid value),
            'Priority' =        (PRIO: any valid priority, but not X)
        BEFORE Internal Processing Fail Time
            RECEIVE BACnet-SimpleACK-PDU
                |
                (BACnet-Error-PDU,
                    'Error Type' = (E: any error)
                )
        IF the IUT is configured to send unconfirmed audit notifications THEN {
            BEFORE AR.Maximum_Send_Delay + Notification Fail Time
                RECEIVE UnconfirmedAuditNotification-Request,
                    'Notifications' = ({        -- there may be a second notification included for the
```

```
                                -- write to Audit_Priority_Filter. If there is, the order
            -- of the notifications in the list is not relevant
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = WRITE,
                    -- source-comment absent
                    target-comment = (any valid value, or absent),
                    invoke-id = (the invoke ID of the write to Present_Value),
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = O1,
                    target-property = Present_Value,
                    target-priority = PRIO,
                    target-value = V,
                    current-value = (the value before the write. may be absent if the value
            size is larger than 32 encoded octets),
                    result = (E, if the op failed, otherwise absent)
                } )
    } ELSE {
        BEFORE AR.Maximum_Send_Delay + Notification Fail Time
            RECEIVE ConfirmedAuditNotification-Request,
                'Notifications' = ({            -- there may be a second notification included for the

                                -- write to Audit_Priority_Filter. If there is, the order
            -- of the notifications in the list is not relevant
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = WRITE,
                    -- source-comment absent
                    target-comment = (any valid value, or absent),
                    invoke-id = (the invoke ID of the write to Present_Value),
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = O1,
                    target-property = Present_Value,
                    target-priority = PRIO,
                    target-value = V,
                    current-value = (the value before the write. may be absent if the value
            size is larger than 32 encoded octets),
                    result = (E, if the op failed, otherwise absent)
                } )
        TRANSMIT BACnet-SimpleACK-PDU
    }
    TRANSMIT WriteProperty-Request,
        'Invoke Id' =       I,
        'Object Identifier' =   O1,
        'Property Identifier' = Present_Value,
        'Property Value' =      (V: any valid value),
        'Priority' =        (PRIO: X)
```

```
    BEFORE Internal Processing Fail Time
        RECEIVE BACnet-SimpleACK-PDU
            |
            (BACnet-Error-PDU,
                'Error Type' = (E: any error)
            )
    WAIT AR.Maximum_Send_Delay + Notification Fail Time
    CHECK(no audit notification sent)
}
```

Notes to Tester: In this test, the audit reporting configuration properties are assumed to be writable. If one is only configurable, then the WRITE operation should be replaced with MAKE.

[Add Auditable_Operations property tests into BTL Specified Tests]

**7.3.1.X501 Auditable_Operations Property Tests**

**7.3.1.X501.1 Non-configurable Auditable_Operations Property Test**

Purpose: Verify that non-configurable Auditable_Operations properties are set to the required value.

Test Concept: Select an object, O1, which has a non-configurable Auditable_Operations property. Verify that the property is set to the required value.

Test Steps:

1. READ AO = Audit_Operations
2. CHECK(Audit_Operations = (FALSE,TRUE,TRUE,TRUE,?,TRUE,?,?,?,?,?,FALSE,FALSE,?,?,?,...))
   -- flags READ, NOTIFICATION and SUBSCRIPTION are FALSE,
   -- flags WRITE, CREATE, DELETE, ACKNOWLEDGE-ALARM are TRUE, and
   -- all other flags can be any value

**7.3.1.X501.2 Auditable_Operations Target Audit Reporting Test**

Purpose: Verify that Auditable_Operations controls which operations are auditable.

Test Concept: The IUT is configured to report some operations and not others through the Auditable_Operations property. Each of the standard auditable operations are performed against the IUT and the filtering provided by Auditable_Operations is verified to be correct.

Configuration Requirements: The IUT is configured to report all audit notifications with the exception that at least some auditable operations are disabled via Auditable_Operations.

Test Steps:

1. REPEAT OP = (each of the standard auditable operations for which the IUT is configured to report
   except any operation which is unreasonably difficult to perform, such as
   AUDIITING_FAILURE) DO {
   MAKE(perform OP on the IUT)
   IF the IUT is configured to send unconfirmed audit notifications THEN {
   BEFORE (Audit Reporter for OP).Maximum_Send_Delay + Notification Fail Time
   RECEIVE UnconfirmedAuditNotification-Request,
   'Notifications' = ({
   -- source-timestamp absent
   target-timestamp = (IUT's local time),
   source-device = TD,
   -- source-object absent
   operation = OP,
   -- source-comment absent
   target-comment = (any valid value, or absent unless OP is GENERAL),
   invoke-id = (the invoke id from the operation, or absent if it was
   unconfirmed),
   source-user-id = (the value from the operation if provided, otherwise
   absent),
   source-user-role = (the value from the operation if provided, otherwise
   absent),
   target-device = IUT,
   target-object = (the target object or absent if the target is not an object),
   target-property = (the target property or absent if the target is not a
   property),
   target-priority = (the priority supplied, or absent if the target is not a
   property. shall be 16 or absent if no priority supplied and the
   target is a property),

```
                    target-value = (the target value or absent if no target value for the
                        operation. may be absent if the value size is larger than 32
                        encoded octets),
                    current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                    result = (the reason for failure if OP failed, otherwise absent)
                } )
    } ELSE {
        BEFORE (Audit Reporter for OP).Maximum_Send_Delay + Notification Fail Time
            RECEIVE ConfirmedAuditNotification-Request,
                'Notifications' = ({
                    -- source-timestamp absent
                    target-timestamp = (IUT's local time),
                    source-device = TD,
                    -- source-object absent
                    operation = OP,
                    -- source-comment absent
                    target-comment = (any valid value, or absent unless OP is GENERAL),
                    invoke-id = (the invoke id from the operation, or absent if it was
                        unconfirmed),
                    source-user-id = (the value from the operation if provided, otherwise
                        absent),
                    source-user-role = (the value from the operation if provided, otherwise
                        absent),
                    target-device = IUT,
                    target-object = (the target object or absent if the target is not an object),
                    target-property = (the target property or absent if the target is not a
                        property),
                    target-priority = (the priority supplied, or absent if the target is not a
                        property. shall be 16 or absent if no priority supplied and the
                        target is a property),
                    target-value = (the target value or absent if no target value for the
                        operation. may be absent if the value size is larger than 32
                        encoded octets),
                    current-value = (the value before the op or absent if no target value.
                        may be absent if the value size is larger than 32 encoded
                        octets),
                    result = (the reason for failure if OP failed, otherwise absent)
                } )
        TRANSMIT BACnet-SimpleACK-PDU
    }
}
2. REPEAT OP = (each of the standard auditable operations for which the IUT is configured to NOT report
        except any operation which is unreasonably difficult to perform, such as
        AUDIITING_FAILURE) DO {
    MAKE(perform OP on the the IUT)
    WAIT (Audit Reporter for OP).Maximum_Send_Delay + Notification Fail Time
    CHECK(no audit notification was received)
}
```

**7.3.1.X501.3 Auditable_Operations Source Audit Reporting Test**

Purpose: Verify that Auditable_Operations controls which operations performed by the IUT are auditable.

Test Concept: The IUT is configured to report some source operations and not others through the Auditable_Operations property. Each of the standard auditable operations which the IUT can perform, are performed by the IUT and the filtering provided by Auditable_Operations is verified to be correct.

Configuration Requirements: The IUT is configured to report all source audit notifications with the exception that at least some auditable operations are disabled via Auditable_Operations.

Test Steps:

1. REPEAT OP = (each of the standard auditable operations the IUT is able to perform on another device and is configured to report except any operation which is unreasonably difficult to perform) DO {
   MAKE(the IUT perform OP on the TD)
   IF the IUT is configured to send unconfirmed audit notifications THEN {
      BEFORE (Audit Reporter for OP).Maximum_Send_Delay + Notification Fail Time
         RECEIVE UnconfirmedAuditNotification-Request,
            'Notifications' = ( {
               source-timestamp = (IUT's local time),
               -- target-timestamp absent
               source-device = IUT,
               source-object = (the object which initiated the op or absent if not initiated by an object),
               operation = OP,
               source-comment = (any valid value, or absent unless OP is GENERAL),
               -- target-comment absent
               invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
               source-user-id = (the value from the operation if provided, otherwise absent),
               source-user-role = (the value from the operation if provided, otherwise absent),
               target-device = TD,
               target-object = (the target object or absent if the target is not an object),
               target-property = (the target property or absent if the target is not a property),
               target-priority = (the priority supplied, or absent if the target is not a property. shall be 16 or absent if no priority supplied and the target is a property),
               target-value = (the target value or absent if no target value for the operation. may be absent if the value size is larger than 32 encoded octets),
               current-value = (the value before the op if the op targeted a property, or absent. May be absent even if targeting a property),
               result = (the reason for failure if the op failed, otherwise absent)
               } )
   } ELSE {
      BEFORE (Audit Reporter for OP).Maximum_Send_Delay + Notification Fail Time
         RECEIVE ConfirmedAuditNotification-Request,
            'Notifications' = ( {
               source-timestamp = (IUT's local time),
               -- target-timestamp absent
               source-device = IUT,
               source-object = (the object which initiated the op or absent if not initiated by an object),
               operation = OP,
               source-comment = (any valid value, or absent unless OP is GENERAL),
               -- target-comment absent
               invoke-id = (the invoke id from the operation, or absent if it was unconfirmed),
               source-user-id = (the value from the operation if provided, otherwise absent),
               source-user-role = (the value from the operation if provided, otherwise absent),
               target-device = TD,

```
                    target-object = (the target object or absent if the target is not an object),
                    target-property = (the target property or absent if the target is not a
                        property),
                    target-priority = (the priority supplied, or absent if the target is not a
                        property. shall be 16 or absent if no priority supplied and the
                        target is a property),
                    target-value = (the target value or absent if no target value for the
                        operation. may be absent if the value size is larger than 32
                        encoded octets),
                    current-value = (the value before the op if the op targeted a property, or
                        absent. May be absent even if targeting a property),
                    result = (the reason for failure if the op failed, otherwise absent)
                    } )
        TRANSMIT BACnet-SimpleACK-PDU
    }
}
```

2. REPEAT OP = (each of the standard auditable operations the IUT is able to perform on another device and is configured to NOT report except any operation which is unreasonably difficult to perform) DO {

```
    MAKE(the IUT perform OP on the TD)
    WAIT AR.Maximum_Send_Delay + Notification Fail Time
    CHECK(no audit notifications were sent)
}
```