



**BACnet[®] TESTING LABORATORIES
ADDENDA**

**Addendum bq to
BTL Test Package 23.3**

**Final
Revised 3/14/2024**

Approved by the BTL Working Group on August 18, 2023;
Approved by the BTL Working Group Voting Members on May 13, 2024;
Published on May 15, 2024.

[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-23.3 bq-1: Access Event Generation [BTLWG-695].....2

In the following document, language to be added to existing clauses within the BTL Test Package 23.3 is indicated through the use of *italics*, while deletions are indicated by ~~striketrough~~. 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-23.3 bq-1: Access Event Generation [BTLWG-695]

Overview:

Introduce a new test to verify that the last event of the Access Point object is always access-denied or access-granted when a single access transaction generates multiple Access Events with access-denied or access-granted events.

Approach:

1. Make(the IUT perform a single access transaction which generates multiple access events with includes access-denied, or access-granted event)
2. repeat x = (1 .. number of expected events)
3. Receive notification, saving type into N[x]
4. Check (N[last entry] is access granted or denied)

Changes:

Checklist Changes

None

Test Plan Changes

[Add new tests to the Base Requirements section in Alarm and Event Management - Access Control - B]

5.32 Alarm and Event Management - Access Control - B

5.32.1 Base Requirements

BTL - 8.4.X19 - Multiple Access Events with either Denied or Granted Access Event Value (ConfirmedEventNotification)		
	Test Conditionality	Shall be skipped if IUT does not perform access transactions which generates multiple access events with one being denied or granted.
	Test Directives	
	Testing Hints	
BTL - 8.5.X19 - Multiple Access Events with either Denied or Granted Access Event Value (UnconfirmedEventNotification)		
	Test Conditionality	Shall be skipped if IUT does not perform access transactions which generates multiple access events with one being denied or granted.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Add new test 8.4.X19]

8.4 ConfirmedEventNotification Service Initiation Tests

8.4.X19 Multiple Access Events with either Denied or Granted Access Event Value (ConfirmedEventNotification)

Reason for Change: No test exists for this functionality.

Purpose: To verify that the last access event generated by the Access Point object is either denied or granted event value.

Test Concept: The test concept corresponds to 8.5.X19.

Configuration Requirements: The configuration requirements are identical to those in 8.5.X19.

Test Steps: The test steps for this test case are identical to the test steps in 8.5.X19, except that the UnconfirmedEventNotification requests are ConfirmedEventNotification requests, and the TD acknowledges receiving the notifications.

Notes to Tester: The passing results for this test case are identical to the ones in 8.5.X19, except that the event notifications shall be conveyed using a ConfirmedEventNotification service request.

[Add new test 8.5.X19]

8.5 UnconfirmedEventNotification Service Initiation Tests

8.5.X19 Multiple Access Events with either Denied or Granted Access Event Value (UnconfirmedEventNotification)

Reason for Change: No test exists for this functionality.

Purpose: To verify that the last access event generated by the Access Point object is either denied or granted event value.

Test Concept: The IUT's Access Point object is configured such that a single access transaction generates multiple Access Events with either one of the event values being denied or granted.

Configuration Requirements:

This test uses the standard configuration for Access Point objects, refer to 7.3.2.X56 for further information. This test also requires the following additional configuration:

a) The IUT shall be configured with at least one instance of Access Point object which generates ConfirmedEventNotification and UnconfirmedEventNotification service requests describing multiple Access Events with denied or granted event values.

Test Steps:

1. READ EventTag = Access_Event_Tag
2. MAKE (the IUT perform a single access transaction, generating multiple access events which includes denied or granted event value)
3. REPEAT X = (1 .. number of expected events) DO {
 BEFORE Notification Fail Time
 RECEIVE UnconfirmedEventNotification-Request,
 'Process Identifier' = (the configured process ID),
 'Initiating Device Identifier' = IUT,
 'Event Object Identifier' = O1,
 'Time Stamp' = (the current local time),
 'Notification Class' = (the configured notification class),
 'Priority' = (the value configured for a TO-NORMAL transition),
 'Event Type' = ACCESS_EVENT,
 'Notify Type' = EVENT | ALARM,
 'AckRequired' = TRUE | FALSE,
 'From State' = NORMAL,
 'To State' = NORMAL,
 'Event Values' = {pAccessEvent stored in N[X]}
 VERIFY pAccessEventTag = EventTag + 1
}
4. CHECK (N[last entry] is GRANTED or any valid denied reason)