[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-TP15.0c-1: Updated FAULT Algorithms (135-2016), this addendum replaces BTL-TP15.0a-1 [wID0166], pg 2.

In the following document, language to be added to existing clauses within the BTL Test Package 15.0 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 addition, 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-15.0c-1: Updated FAULT Algorithms

Overview:

Addendum 135-2010af-21 and af-32 at Protocol_Revision 13 added language and many new FAULT algorithms to all objects that provide fault reporting, and to the Event Enrollment object.

Changes:

[In BTL Specified Tests, add these new tests]
8.5.X1 CHANGE_OF_RELIABILITY Tests

8.5.X1.1 CHANGE_OF_RELIABILITY with No Fault Algorithm

Purpose: To verify the correct operation of an object that supports first stage reliability evaluation and does not apply a standardized fault algorithm.

Test Concept: Select an object, O1 that supports first stage reliability evaluation and does not apply a standardized fault algorithm. Ensure that no other fault conditions exist for the object. Create a fault condition. Verify the transition to fault is generated with Reliability set to R1. Remove the fault condition and verify the object transitions out of fault.

Test Configuration: O1 is configured to detect and report faults using unconfirmed event notifications. O1 is configured to have no fault conditions present and the Event_State is NORMAL.

Test Steps:

```
1.      VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.      VERIFY Event_State = NORMAL
3.      MAKE(O1 enter a fault condition)
4.      BEFORE Notification Fail Time
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         O1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      NORMAL,
                'To State' =                        FAULT,
                'Event Values' =                    ( R1 any valid BACnetReliability,
                                                      (?, T, ?, ?),
                                                      (A list of valid values for properties required to be reported
                                                      for O1, and 0 or more other properties of O1)
                                                    )
5.      VERIFY pCurrentReliability = R1
6.      VERIFY Event_State = FAULT
7.      MAKE(O1clear the fault condition)
8.      BEFORE Notification Fail Time
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         O1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                 ALARM | EVENT,
                'AckRequired' =                 TRUE | FALSE,
                'From State' =                  FAULT,
                'To State' =                    NORMAL,
                'Event Values' =                ( NO_FAULT_DETECTED,
                                                  (?, F, ?, ?),
                                                  (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                )
9.      VERIFY pCurrentReliability = NO_FAULT_DETECTED
10.     VERIFY Event_State = NORMAL
```

Notes to Tester: The mechanism to enter the NONE fault algorithm is a local matter.

8.5.X1.2 CHANGE_OF_RELIABILITY with the FAULT_CHARACTERSTRING Algorithm

Purpose: To verify the correct operation of the FAULT_CHARACTERSTRING fault algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_CHARACTERSTRING algorithm, and no other fault conditions exist for the object. pMonitoredValue is changed to a fault string and back to a non-fault string. It is verified that O1 generates the correct transitions.

Test Configuration: O1 is configured to detect and report faults, to have no fault conditions present, and to be in the NORMAL state. FVSET is the set of character strings defined as fault values for O1. ONVSET is the set of character strings defined as offnormal values for O1. FV1 contain a substring that exists in FVSET. If the empty string is included in the FVSET, then FV1 should be the empty string. NFV1 is a string value that does not contain substrings from FVSET or ONVSET.

Test Steps:

```
1.      VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.      VERIFY Event_State = NORMAL
3.      IF (pMonitoredValue is writable) THEN
            WRITE pMonitoredValue = FV1
        ELSE
            MAKE (pMonitoredValue = FV1)
4.      BEFORE **Notification Fail Time**
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =          (any valid process identifier),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' =     O1,
                'Time Stamp' =                  (the current local time or sequence number),
                'Notification Class' =          (the notification class configured for O1),
                'Priority' =                    (the value configured for the transition),
                'Event Type' =                  CHANGE_OF_RELIABILITY,
                'Message Text' =                (optional, any valid message text),
                'Notify Type' =                 ALARM | EVENT,
                'AckRequired' =                 TRUE | FALSE,
                'From State' =                  NORMAL,
                'To State' =                    FAULT,
                'Event Values' =                ( MULTI_STATE_FAULT,
                                                  (T, T, ?, ?),
                                                  (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                )
5.      VERIFY pCurrentReliability = MULTI_STATE_FAULT
6.      VERIFY Event_State = FAULT
7.      IF (pMonitoredValue is writable) THEN
            WRITE pMonitoredValue = NFV1
        ELSE
```

```
          MAKE (pMonitoredValue = NFV1)
8.     BEFORE **Notification Fail Time**
          RECEIVE UnconfirmedEventNotification-Request
               'Process Identifier' =              (any valid process identifier),
               'Initiating Device Identifier' =    IUT,
               'Event Object Identifier' =         O1,
               'Time Stamp' =                      (the current local time or sequence number),
               'Notification Class' =              (the notification class configured for O1),
               'Priority' =                        (the value configured for the transition),
               'Event Type' =                      CHANGE_OF_RELIABILITY,
               'Message Text' =                    (optional, any valid message text),
               'Notify Type' =                     ALARM | EVENT,
               'AckRequired' =                     TRUE | FALSE,
               'From State' =                      FAULT,
               'To State' =                        NORMAL,
               'Event Values' =                    ( NO_FAULT_DETECTED,
                                                     (F, F, ?, ?),
                                                     (A list of valid values for properties required to be reported
                                                     for O1, and 0 or more other properties of O1)
                                                   )
9.     VERIFY pCurrentReliability = NO_FAULT_DETECTED
10.    VERIFY Event_State = NORMAL
```

Notes to Tester: Note that a string is considered a substring of itself. Values required and allowed for O1 are described in standard 135 as "Properties Reported in CHANGE_OF_RELIABILITY Notifications" (Table 13-5 in 135-2016) along with supporting paragraphs.

8.5.X1.3 CHANGE_OF_RELIABILITY with the FAULT_EXTENDED Algorithm

Purpose: To verify the correct operation of the FAULT_EXTENDED fault algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_EXTENDED algorithm, and either pMonitoredValue is configured. Ensure that no other fault conditions exist for the object. In object O1, a condition is created that is detected as a fault by the FAULT_EXTENDED algorithm configured. The fault condition is then removed. It is verified that O1 generates the correct notifications.

Test Configuration: O1 is configured to detect and report faults. O1 is configured to have no fault conditions present, and has an Event_State of NORMAL.

Test Steps:

```
1.     VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.     VERIFY Event_State = NORMAL
3.     MAKE (a fault condition exist)
4.     BEFORE **Notification Fail Time**
          RECEIVE UnconfirmedEventNotification-Request
               'Process Identifier' =              (any valid process identifier),
               'Initiating Device Identifier' =    IUT,
               'Event Object Identifier' =         O1,
               'Time Stamp' =                      (the current local time or sequence number),
               'Notification Class' =              (the notification class configured for O1),
               'Priority' =                        (the value configured for the transition),
               'Event Type' =                      CHANGE_OF_RELIABILITY,
               'Message Text' =                    (optional, any valid message text),
               'Notify Type' =                     ALARM | EVENT,
               'AckRequired' =                     TRUE | FALSE,
               'From State' =                      NORMAL,
               'To State' =                        FAULT,
               'Event Values' =                    ( (R1: any valid reliability value),
                                                     (T, T, ?, ?),
                                        (a vendor specified set of values)
                                       )
5.    VERIFY pCurrentReliability = R1
6.    VERIFY Event_State = FAULT
7.    MAKE (remove the fault condition)
8.    BEFORE Notification Fail Time
         RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =              (any valid process identifier),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         O1,
            'Time Stamp' =                      (the current local time or sequence number),
            'Notification Class' =              (the notification class configured for O1),
            'Priority' =                        (the value configured for the transition),
            'Event Type' =                      CHANGE_OF_RELIABILITY,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     ALARM | EVENT,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      FAULT,
            'To State' =                        NORMAL,
            'Event Values' =                    ( NO_FAULT_DETECTED,
                                                  (?, F, ?, ?),
                                                  (a vendor specified set of values)
                                                )
9.    VERIFY pCurrentReliability = NO_FAULT_DETECTED
10.   VERIFY Event_State = NORMAL
```

8.5.X1.4 CHANGE_OF_RELIABILITY with the FAULT_LIFE_SAFETY Algorithm

Purpose: To verify the correct operation of the FAULT_LIFE_SAFETY fault algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_LIFE_SAFETY algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredValue to FV1, a value which indicates a FAULT_LIFE_SAFETY fault condition. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredValue to NV1, a value which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Test Configuration: O1 is configured to detect faults and to report those using unconfirmed event notifications. O1 is initially configured to have no fault conditions present, and has an Event_State of NORMAL. FV1 is a value for pMonitoredValue which indicates a fault condition, and NV1 is a value for pMonitoredValue which does not indicate a fault condition.

Test Steps:

```
1.    VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.    VERIFY Event_State = NORMAL
3.    IF (pMonitoredValue is writable) THEN
         WRITE pMonitoredValue = FV1
      ELSE
         MAKE (pMonitoredValue = FV1)
4.    BEFORE Notification Fail Time
         RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =              (any valid process identifier),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         O1,
            'Time Stamp' =                      (the current local time or sequence number),
            'Notification Class' =              (the notification class configured for O1),
            'Priority' =                        (the value configured for the transition),
            'Event Type' =                      CHANGE_OF_RELIABILITY,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     ALARM | EVENT,
            'AckRequired' =                     TRUE | FALSE,
        'From State' =                  NORMAL,
        'To State' =                    FAULT,
        'Event Values' =                ( MULTI_STATE_FAULT,
                                          (T, T, ?, ?),
                                          (A list of valid values for properties required to be reported
                                          for O1, and 0 or more other properties of O1)
                                        )
5.  VERIFY pCurrentReliability = MULTI_STATE_FAULT
6.  VERIFY Event_State = FAULT
7.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = NV1
    ELSE
        MAKE (pMonitoredValue = NV1)
8.  BEFORE **Notification Fail Time**
        RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =              (any valid process identifier),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         O1,
            'Time Stamp' =                      (the current local time or sequence number),
            'Notification Class' =              (the notification class configured for O1),
            'Priority' =                        (the value configured for the transition),
            'Event Type' =                      CHANGE_OF_RELIABILITY,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     ALARM | EVENT,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      FAULT,
            'To State' =                        NORMAL,
            'Event Values' =                    ( NO_FAULT_DETECTED,
                                                  (F, F, ?, ?),
                                                  (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                )
9.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY Event_State = NORMAL
```

8.5.X1.5 CHANGE_OF_RELIABILITY with the FAULT_STATE Algorithm

Purpose: To verify the correct operation of the FAULT_STATE fault algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_STATE algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredValue to FV1, a value which indicates a FAULT_STATE fault condition. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredValue to NV1, a value which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Test Configuration: O1 is configured to detect faults and to report those using unconfirmed event notifications. O1 is initially configured to have no fault conditions present, and an Event_State of NORMAL. FV1 is a value for pMonitoredValue which indicates a fault condition, and NV1 is a value for pMonitoredValue which does not indicate a fault condition.

Test Steps:

```
1.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.  VERIFY Event_State = NORMAL
3.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = FV1
    ELSE
        MAKE (pMonitoredValue = FV1)
4.  BEFORE **Notification Fail Time**
        RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =              (any valid process identifier),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         O1,
            'Time Stamp' =                      (the current local time or sequence number),
            'Notification Class' =              (the notification class configured for O1),
            'Priority' =                        (the value configured for the transition),
            'Event Type' =                      CHANGE_OF_RELIABILITY,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     ALARM | EVENT,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      NORMAL,
            'To State' =                        FAULT,
            'Event Values' =                    ( MULTI_STATE_FAULT,
                                                  (T, T, ?, ?),
                                                  (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                )
5.      VERIFY pCurrentReliability = MULTI_STATE_FAULT
6.      VERIFY Event_State = FAULT
7.      IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = NV1
    ELSE
        MAKE (pMonitoredValue = NV1)
8.      BEFORE **Notification Fail Time**
        RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =              (any valid process identifier),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         O1,
            'Time Stamp' =                      (the current local time or sequence number),
            'Notification Class' =              (the notification class configured for O1),
            'Priority' =                        (the value configured for the transition),
            'Event Type' =                      CHANGE_OF_RELIABILITY,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     ALARM | EVENT,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      FAULT,
            'To State' =                        NORMAL,
            'Event Values' =                    ( NO_FAULT_DETECTED,
                                                  (F, F, ?, ?),
                                                  (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                )
9.      VERIFY pCurrentReliability = NO_FAULT_DETECTED
10.     VERIFY Event_State = NORMAL
```

8.5.X1.6 CHANGE_OF_RELIABILITY with the FAULT_STATUS_FLAGS Algorithm

Purpose: To verify the correct operation of the FAULT_STATUS_FLAGS fault algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_STATUS_FLAGS algorithm. Ensure that no other fault conditions exist for the object. Set pMonitoredValue to FV1, a value which indicates a FAULT_STATUS_FLAGS fault condition. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredValue to NV1, a value which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Test Configuration: O1 is configured to detect faults and to report those using unconfirmed event notifications. O1 is initially configured to have no fault conditions present, and Event_State is NORMAL. FV1 is a value for pMonitoredValue which indicates a fault condition, and NV1 is a value for pMonitoredValue which does not indicate a fault condition.

Test Steps:

```
1.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.  VERIFY Event_State = NORMAL
3.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = FV1
    ELSE
        MAKE (pMonitoredValue = FV1)
4.  BEFORE Notification Fail Time
        RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =            (any valid process identifier),
            'Initiating Device Identifier' =  IUT,
            'Event Object Identifier' =       O1,
            'Time Stamp' =                    (the current local time or sequence number),
            'Notification Class' =            (the notification class configured for O1),
            'Priority' =                      (the value configured for the transition),
            'Event Type' =                    CHANGE_OF_RELIABILITY,
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   ALARM | EVENT,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    NORMAL,
            'To State' =                      FAULT,
            'Event Values' =                  ( MEMBER_FAULT,
                                                (T, T, ?, ?),
                                                (A list of valid values for properties required to be reported
                                                for O1, and 0 or more other properties of O1)
                                              )
5.  VERIFY pCurrentReliability = MEMBER_FAULT
6.  VERIFY Event_State = FAULT
7.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = NV1
    ELSE
        MAKE (pMonitoredValue = NV1)
8.  BEFORE Notification Fail Time
        RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =            (any valid process identifier),
            'Initiating Device Identifier' =  IUT,
            'Event Object Identifier' =       O1,
            'Time Stamp' =                    (the current local time or sequence number),
            'Notification Class' =            (the notification class configured for O1),
            'Priority' =                      (the value configured for the transition),
            'Event Type' =                    CHANGE_OF_RELIABILITY,
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   ALARM | EVENT,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    FAULT,
            'To State' =                      NORMAL,
            'Event Values' =                  ( NO_FAULT_DETECTED,
                                                (F, F, ?, ?),
                                                (A list of valid values for properties required to be reported
                                                for O1, and 0 or more other properties of O1)
                                              )
9.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY Event_State = NORMAL
```

8.5.X1.7 Event Enrollment Fault Condition Precedence Tests

8.5.X1.7.1 Internal Faults Take Precedence Over Monitored Object Faults

Purpose: To verify that the Event Enrollment object's fault detection gives precedence to internal unreliable operational faults over faults in the monitored object.

Test Concept: Select an Event Enrollment object EE1 which can detect internal faults and which monitors an object O1 that can detect faults. Test that an internal unreliable operational fault takes precedence over a monitored object fault.

Test Steps:

1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY Event_State = NORMAL
3. MAKE(a condition exist which will cause O1 to transition into fault)
4. VERIFY pCurrentReliability = MONITORED_OBJECT_FAULT
5. VERIFY Event_State = FAULT
6. MAKE(a condition exist which will cause EE1 to transition into internal fault R1)
7. VERIFY pCurrentReliability = R1
8. MAKE(clear the condition that caused EE1 to enter into an internal fault)
9. VERIFY pCurrentReliability = MONITORED_OBJECT_FAULT
10. MAKE(clear the condition that caused O1 to transition into fault)
11. VERIFY pCurrentReliability = NO_FAULT_DETECTED
12. VERIFY Event_State = NORMAL

8.5.X1.7.2 Monitored Object Faults Take Precedence Over Fault Algorithms

Purpose: To verify that the Event Enrollment object's fault detection gives precedence to faults in the monitored object over faults detected by fault algorithm.

Test Concept: Select an Event Enrollment object EE1 which applies a fault algorithm and which monitors an object O1 that can detect faults. Test that a monitored object fault takes precedence over a standard fault algorithm fault.

Test Steps:

1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY Event_State = NORMAL
3. MAKE(a condition that results in a fault due to a configured fault algorithm with a reliability value, R1)
4. VERIFY pCurrentReliability = R1
5. VERIFY Event_State = FAULT
6. MAKE(a condition exist which will cause O1 to transition into fault)
7. VERIFY pCurrentReliability = MONITORED_OBJECT_FAULT
8. MAKE(clear the condition that caused O1 to transition into fault)
9. VERIFY pCurrentReliability = R1
10. MAKE(clear the condition for the fault algorithm)
11. VERIFY pCurrentReliability = NO_FAULT_DETECTED
12. VERIFY Event_State = NORMAL

8.5.X1.7.3 Internal Faults Take Precedence Over Fault Algorithms

Purpose: To verify that the Event Enrollment object's fault detection gives precedence to internal unreliable operational faults over faults detected by fault algorithm.

Test Concept: Select an Event Enrollment object EE1 which can detect internal faults and which applies a fault algorithm. Test that an internal unreliable operational fault takes precedence over a standard fault algorithm fault.

Test Steps:

1. VERIFY pCurrentReliability = NO_FAULT_DETECTED

```
2.      VERIFY Event_State = NORMAL
3.      MAKE(a condition that results in a fault due to a configured fault algorithm with a reliability value, R1)
4.      VERIFY pCurrentReliability = R1
5.      VERIFY Event_State = FAULT
6.      MAKE(a condition exist which will cause EE1 to transition into internal fault R2, different from R1)
7.      VERIFY pCurrentReliability = R2
8.      MAKE(clear the condition that caused EE1 to enter into an internal fault)
9.      VERIFY pCurrentReliability = R1
10.     MAKE(clear the condition for the fault algorithm)
11.     VERIFY pCurrentReliability = NO_FAULT_DETECTED
12.     VERIFY Event_State = NORMAL
```

8.5.X1.8 CHANGE_OF_RELIABILITY of Event Enrollment Object, Monitored Object Fault

Purpose: To verify the proper operation of the Event Enrollment object's fault detection when the monitored object enters the fault state.

Test Concept: Select an Event Enrollment object EE1 that monitors an object M1 that can transition into FAULT. Starting with both objects in a NORMAL state, cause a condition which results in a fault in M1. Verify EE1 reports the fault. Clear the condition and verify EE1 reports the return to NORMAL.

Test Configuration: EE1 is configured to process faults in M1 and to report those using unconfirmed event notifications. EE1 and M1 are each initially configured to have no fault conditions present, and Event_State is NORMAL.

Test Steps:

```
1.      VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.      VERIFY Event_State = NORMAL
3.      MAKE (M1 enter any fault state)
4.      BEFORE **Notification Fail Time**
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         EE1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for EE1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      NORMAL,
                'To State' =                        FAULT,
                'Event Values' =                    ( MONITORED_OBJECT_FAULT,
                                                      (T, T, ?, ?),
                                                      M1,
                                                      (optional, property value of M1),
                                                      (optional, M1 Status_Flags, (?, T, ?, ?)),
                                                      (0 or more other properties of M1)
                                                    )
5.      VERIFY pCurrentReliability = MONITORED_OBJECT_FAULT
6.      VERIFY Event_State = FAULT
7.      MAKE (M1 clear fault state)
8.      BEFORE **Notification Fail Time**
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         EE1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =            (the notification class configured for EE1),
                'Priority' =                      (the value configured for the transition),
                'Event Type' =                    CHANGE_OF_RELIABILITY,
                'Message Text' =                  (optional, any valid message text),
                'Notify Type' =                   ALARM | EVENT,
                'AckRequired' =                   TRUE | FALSE,
                'From State' =                    FAULT,
                'To State' =                      NORMAL,
                'Event Values' =                  ( NO_FAULT_DETECTED,
                                                      (F, F, ?, ?),
                                                      M1,
                                                      (optional, property value of M1),
                                                      (optional, M1 Status_Flags, (?, F, ?, ?)),
                                                      (0 or more other properties of M1)
                                                    )
9.    VERIFY pCurrentReliability = NO_FAULT_DETECTED
10.   VERIFY Event_State = NORMAL
```

8.5.X1.9 CHANGE_OF_RELIABILITY of Event Enrollment Object Fault

Purpose: To verify the Event Enrollment object generates a fault event when the object enters into fault due to an internal unreliable operation.

Test Concept: Select an Event Enrollment object EE1 that can be made to enter into fault due to an internal unreliable operation. Starting EE1 in a NORMAL state, cause a condition which results in an internal fault. Verify that EE1 reports the fault. Clear the condition and verify that EE1 reports the return to NORMAL.

Test Configuration: EE1 is configured to be able to enter a fault state and to report those using unconfirmed event notifications. EE1 is initially configured to have no fault conditions present, and Event_State is NORMAL.

Test Steps:

```
1.    VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.    VERIFY Event_State = NORMAL
3.    MAKE (EE1 enter any internal fault state)
4.    BEFORE Notification Fail Time
         RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =            (any valid process identifier),
            'Initiating Device Identifier' =  IUT,
            'Event Object Identifier' =       EE1,
            'Time Stamp' =                    (the current local time or sequence number),
            'Notification Class' =            (the notification class configured for EE1),
            'Priority' =                      (the value configured for the transition),
            'Event Type' =                    CHANGE_OF_RELIABILITY,
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   ALARM | EVENT,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    NORMAL,
            'To State' =                      FAULT,
            'Event Values' =                  ( (R1: any value other than
                                                      MONITORED_OBJECT_FAULT
                                                      and NO_FAULT_DETECTED),
                                                (T, T, ?, ?),
                                                (M1, any valid monitored object),
                                                (optional, property value of M1),
                                                (optional, M1 Status_Flags, (?, F, ?, ?)),
                                                (0 or more other properties of M1)
                                              )
5.    VERIFY pCurrentReliability = R1
6.    VERIFY Event_State = FAULT
```

```
7.      MAKE (EE1 clear fault state)
8.      BEFORE **Notification Fail Time**
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         EE1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for EE1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      FAULT,
                'To State' =                        NORMAL,
                'Event Values' =                    ( NO_FAULT_DETECTED,
                                                      (F, F, ?, ?),
                                                      M1,
                                                      (optional, property value of M1),
                                                      (optional, M1 Status_Flags, (?, F, ?, ?)),
                                                      (0 or more other properties of M1)
                                                    )
9.      VERIFY pCurrentReliability = NO_FAULT_DETECTED
10.     VERIFY Event_State = NORMAL
```

8.5.X1.10 After FAULT-to-NORMAL, Re-Notification of OFFNORMAL

Purpose: To verify that objects go to the NORMAL state after leaving the FAULT state, then transition to OFFNORMAL if the condition still exists.

Test Concept: Select a fault detecting object O1 which is able to detect OFFNORMAL conditions. Make O1 transition to an OFFNORMAL state and then transition to FAULT. Remove the condition causing the FAULT and verify O1 transitions from FAULT to NORMAL, then verify that the object transitions from NORMAL to the original OFFNORMAL state.

Test Configuration: O1 is configured to detect and report unconfirmed events and faults. O1 is configured to have no fault conditions present, and Event_State is OFFNORMAL.

Test Steps:

```
1.      VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.      VERIFY Event_State = NORMAL
3.      MAKE(O1transition to an off normal state)
4.      BEFORE **Notification Fail Time**
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         O1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      (ET1, any valid off normal event type),
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      NORMAL,
                'To State' =                        OFFNORMAL,
                'Event Values' =                    (property-values appropriate for O1)
5.      VERIFY Event_State = OFFNORMAL
6.      MAKE(O1 enter a fault state)
```

```
7.      BEFORE Notification Fail Time
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         O1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      OFFNORMAL,
                'To State' =                        FAULT,
                'Event Values' =                    ( (R1 any valid BACnetReliability),
                                                      (?, T, ?, ?),
                                                      (A list of valid values for properties required to be reported
                                                      for O1, and 0 or more other properties of O1)
                                                    )
8.      MAKE(O1 clear the fault condition)
9.      BEFORE Notification Fail Time
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         O1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      FAULT,
                'To State' =                        NORMAL,
                'Event Values' =                    ( NO_FAULT_DETECTED,
                                                      (F, F, ?, ?),
                                                      (A list of valid values for properties required to be reported
                                                      for O1, and 0 or more other properties of O1)
                                                    )
10.     BEFORE Notification Fail Time
            RECEIVE UnconfirmedEventNotification-Request
                'Process Identifier' =              (any valid process identifier),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         O1,
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      ET1,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      NORMAL,
                'To State' =                        OFFNORMAL,
                'Event Values' =                    (property-values appropriate for O1)
11.     VERIFY pCurrentReliability = NO_FAULT_DETECTED
12.     VERIFY Event_State = OFFNORMAL
```

8.5.X1.11 CHANGE_OF_RELIABILITY with First Stage Object Fault

Purpose: To verify that fault conditions due to first stage faults are detected and reported.

Test Concept: An object in the IUT, O1, which can detect at least one first stage fault is selected. One of O1's detectable first stage faults, R1, is selected for the test. O1 begins the test in the NORMAL state with pCurrentReliability equal to NO_FAULT_DETECTED. The first stage fault condition, R1, is made to exist and it is verified that the pCurrentReliability changes to R1. It is verified that O1 generates the appropriate event notification. The fault condition is removed, and it is verified that the pCurrentReliability returns to NO_FAULT_DETECTED and the appropriate event notification message is generated.

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

Test Steps:

```
1.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.  VERIFY Event_State = NORMAL
3.  MAKE (pCurrentReliability = R1)
4.  BEFORE **Notification Fail Time**
        RECEIVE UnconfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         O1,
            'Time Stamp' =                      (the current local time or sequence number),
            'Notification Class' =              (the notification class configured for O1),
            'Priority' =                        (the value configured for the transition),
            'Event Type' =                      CHANGE_OF_RELIABILITY,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      NORMAL,
            'To State' =                        FAULT,
            'Event Values' =                    ( R1,
                                                  (?, T, ?, ?),
                                                  (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                )
5.  VERIFY pCurrentReliability = R1
6.  VERIFY Event_State = FAULT
7.  MAKE (pCurrentReliability = NO_FAULT_DETECTED)
8.  BEFORE **Notification Fail Time**
        RECEIVE UnconfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         O1,
            'Time Stamp' =                      (the current local time or sequence number),
            'Notification Class' =              (the notification class configured for O1),
            'Priority' =                        (the value configured for the transition),
            'Event Type' =                      CHANGE_OF_RELIABILITY,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      FAULT,
            'To State' =                        NORMAL,
            'Event Values' =                    ( NO_FAULT_DETECTED,
                                                 (?, F, ?, ?),
                                                 (A list of valid values for properties required to be reported
                                                 for O1, and 0 or more other properties of O1)
                                                )
9.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY Event_State = NORMAL
```

[In BTL Test Plan, add the new section “Supports Event Reporting” to Global Group Object]

3.36.20 Supports Event Reporting

The IUT supports, or can be configured to support, event reporting in the Global Group Object.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B or AE-N-E-B in the Checklist with option " Implements the CHANGE_OF_RELIABILITY – FAULT_STATUS_FLAGS Algorithm".
	Testing Hints	
	Notes & Results	

BTL - 8.5.X1.11 - CHANGE_OF_RELIABILITY with First Stage Object fault		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test shall be executed if the object's Reliability property can be made to equal COMMUNICATION_FAILURE otherwise this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Test Plan, Append section 5.2.1 Base Requirements, AE-N-I-B]

BTL - 8.5.X1.10 - After FAULT-to-NORMAL, Re-Notification of OFFNORMAL		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT has no object in which CHANGE_OF_RELIABILITY is implemented in an object that can be configured into an offnormal state, this test shall be skipped.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant shall be selected.
	Testing Hints	
	Notes & Results	

[In BTL Test Plan, add tests to Alarm and Event - Notification - Internal - B Base Requirements, with Test Directives to indicate selecting objects to which to apply the tests]

5.2.30 Implements the CHANGE_OF_RELIABILITY – No Fault Algorithm

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

BTL - 8.5.X1.1 - CHANGE_OF_RELIABILITY with No Fault Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***

	Test Conditionality	The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and which does not apply a standardized fault algorithm.
	Test Directives	Apply this test to all object types that support fault detection but do not apply a standardized fault algorithm..
	Testing Hints	
	Notes & Results	

5.2.31 Implements the CHANGE_OF_RELIABILITY – FAULT_CHARACTERSTRING Algorithm

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

BTL - 8.5.X1.2 - CHANGE_OF_RELIABILITY with the FAULT_CHARACTERSTRING Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant shall be selected.
	Testing Hints	
	Notes & Results	

5.2.32 Implements the CHANGE_OF_RELIABILITY – FAULT_EXTENDED Algorithm

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

BTL - 8.5.X1.3 - CHANGE_OF_RELIABILITY with the FAULT_EXTENDED Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant shall be selected.
	Testing Hints	
	Notes & Results	

5.2.33 Implements the CHANGE_OF_RELIABILITY – FAULT_LIFE_SAFETY Algorithm

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

BTL - 8.5.X1.4 - CHANGE_OF_RELIABILITY with the FAULT_LIFE_SAFETY Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of

		those properties. At least one instance of each variant shall be selected.
	Testing Hints	
	Notes & Results	

5.2.34 Implements the CHANGE_OF_RELIABILITY – FAULT_STATE Algorithm

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

BTL - 8.5.X1.5 - CHANGE_OF_RELIABILITY with the FAULT_STATE Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant shall be selected.
	Testing Hints	
	Notes & Results	

5.2.35 Implements the CHANGE_OF_RELIABILITY – FAULT_STATUS_FLAGS Algorithm

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

BTL - 8.5.X1.6 - CHANGE_OF_RELIABILITY with the FAULT_STATUS_FLAGS Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed..
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant shall be selected.
	Testing Hints	
	Notes & Results	

5.2.36 Supports CHANGE_OF_RELIABILITY in the Event Enrollment Object

The IUT contains, or can be made to contain, an Event Enrollment object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY.

BTL - 8.5.X1.7.1 - Internal Faults Take Precedence Over Monitored Object Faults		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT does not support an Event Enrollment object which can detect internal faults and monitor an object which detects faults, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 8.5.X1.7.2 - Monitored Object Faults Take Precedence Over Fault Algorithms		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***

	Test Conditionality	If the IUT does not support an Event Enrollment object which monitors an object which detects faults and which applies a fault algorithm, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 8.5.X1.7.3 - Internal Faults Take Precedence Over Fault Algorithms		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT does not support an Event Enrollment object which can detect internal faults and which applies a fault algorithm, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 8.5.X1.8 - CHANGE_OF_RELIABILITY of Event Enrollment Object, Monitored Object Fault		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT has no Event Enrollment object where the Monitored_Object that can transition to fault, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 8.5.X1.9 - CHANGE_OF_RELIABILITY of Event Enrollment Object Fault		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT has no Event Enrollment object that detects an internal unreliable operational fault, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Test Plan, Append section 5.3.1 Base Requirements, AE-N-E-B]

BTL - 8.5.X1.10 - After FAULT-to-NORMAL, Re-Notification of OFFNORMAL		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT has no object in which CHANGE_OF_RELIABILITY is implemented and which can be configured into an offnormal state, this test shall be skipped.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant shall be selected.
	Testing Hints	
	Notes & Results	

[In BTL Test Plan, add tests to Alarm and Event - Notification - External - B Base Requirements, with Test Directives to indicate selecting objects to which to apply the tests]

5.3.22 Implements the CHANGE_OF_RELIABILITY – No Fault Algorithm

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

BTL - 8.5.X1.1 - CHANGE_OF_RELIABILITY with No Fault Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and which does not apply a standardized fault algorithm.
	Test Directives	Apply this test to all object types that support fault detection but do not apply a standardized fault algorithm..
	Testing Hints	
	Notes & Results	

5.3.23 Implements the CHANGE_OF_RELIABILITY – FAULT_CHARACTERSTRING Algorithm

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

BTL - 8.5.X1.2 - CHANGE_OF_RELIABILITY with the FAULT_CHARACTERSTRING Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for each object type that support this fault algorithm,
	Testing Hints	
	Notes & Results	

5.3.24 Implements the CHANGE_OF_RELIABILITY – FAULT_EXTENDED Algorithm

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

BTL - 8.5.X1.3 - CHANGE_OF_RELIABILITY with the FAULT_EXTENDED Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed..
	Test Directives	Repeat the test for each object type that support this fault algorithm,
	Testing Hints	
	Notes & Results	

5.3.25 Implements the CHANGE_OF_RELIABILITY – FAULT_LIFE_SAFETY Algorithm

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

BTL - 8.5.X1.4 - CHANGE_OF_RELIABILITY with the FAULT_LIFE_SAFETY Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***

	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for each object type that support this fault algorithm,
	Testing Hints	
	Notes & Results	

5.3.26 Implements the CHANGE_OF_RELIABILITY – FAULT_STATE Algorithm

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

BTL - 8.5.X1.5 - CHANGE_OF_RELIABILITY with the FAULT_STATE Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for each object type that support this fault algorithm,
	Testing Hints	
	Notes & Results	

5.3.27 Implements the CHANGE_OF_RELIABILITY – FAULT_STATUS_FLAGS Algorithm

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

BTL - 8.5.X1.6 - CHANGE_OF_RELIABILITY with the FAULT_STATUS_FLAGS Algorithm		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for each object type that support this fault algorithm,
	Testing Hints	
	Notes & Results	

5.3.28 Supports CHANGE_OF_RELIABILITY in the Event Enrollment Object

The IUT contains, or can be made to contain, an Event Enrollment object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY.

BTL - 8.5.X1.7.1 - Internal Faults Take Precedence Over Monitored Object Faults		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT does not support an Event Enrollment object which can detect internal faults and monitor an object which detects faults, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 8.5.X1.7.2 - Monitored Object Faults Take Precedence Over Fault Algorithms		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test	If the IUT does not support an Event Enrollment object which monitors

	Conditionality	an object which detects faults and which applies a fault algorithm, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 8.5.X1.7.3 - Internal Faults Take Precedence Over Fault Algorithms		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT does not support an Event Enrollment object which can detect internal faults and which applies a fault algorithm, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 8.5.X1.8 - CHANGE_OF_RELIABILITY of Event Enrollment Object, Monitored Object Fault		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT has no Event Enrollment object where the Monitored_Object that can transition to fault, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 8.5.X1.9 - CHANGE_OF_RELIABILITY of Event Enrollment Object Fault		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If the IUT has no Event Enrollment object that detects an internal unreliable operational fault, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Checklist, add new sections as shown here in italics]

Support	**Listing**	**Option**
Alarm and Event - Notification - Internal - B		
. . .		
	O	Implements intrinsic alarming in an Integer object
	C[3]	*Implements the CHANGE_OF_RELIABILITY – No Fault Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_CHARACTERSTRING Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_EXTENDED Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_LIFE_SAFETY Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_STATE Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_STATUS_FLAGS Algorithm*
	C[3]	*Supports CHANGE_OF_RELIABILITY in the Event Enrollment Object*
. . .		

Support	**Listing**	**Option**
Alarm and Event - Notification - External - B		
. . .		
	C[1]	Implements the UNSIGNED_RANGE algorithm
	C[3]	*Implements the CHANGE_OF_RELIABILITY – No Fault Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_CHARACTERSTRING Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_EXTENDED Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_LIFE_SAFETY Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_STATE Algorithm*
	C[3]	*Implements the CHANGE_OF_RELIABILITY – FAULT_STATUS_FLAGS Algorithm*
	C[3]	*Supports CHANGE_OF_RELIABILITY in the Event Enrollment Object*
. . .		

Global Group Object		
	…	
	O	Supports Event Reporting