

Clarification Request

References: BTL Specified Tests 9.14.2.3

Date of BTL-WG Response: 10-Nov-2016 and 01-Dec-2016

☒ All actions necessitated have been completed

Background:

The DM-LM-B add list element request INVALID_DATATYPE negative testing is sent for various list type properties with an invalid data type. (Two or more elements to be added to the list with the second element having an inappropriate datatype). Case here described is for Schedule Object's List of Object property reference property.

We tried to distinguish through packet but there is "NO" existence of a tag to determine whether next octet is a next reference or optional property.

There are two cases :

1. Suppose the first reference is correct (DeviceObjectPropertyReference) with all the properties (ObjectId, Property, Array Index and Device ID(optional)) then the packet will be:

0A6CAB77 BAC0810A 001F0104 00035E08 0C046DC6

D619363E 0C00ADC6 D719553C 000007D1 3F

This packet starts with 3E for list and it looks objectId from "0C", Property from "19 55" and deviceId from "3C".

If we add another correct reference as "DeviceObjectPropertyReference" then the packet will start with "0C"

0A6CAB77 BAC0810A 002B0104 00035F08 0C046DC6

D619363E 0C00ADC6 D719553C 000007D1 0C004000

0B19553C 00000070 3F

In this case if we add next reference as invalid (BOOLEAN) then the packet will be able to discriminate the next data type after reading the tag '11' (for Boolean) and failed index will be correct as 3

0A6CAB77 BAC0810A 002C0104 00035F08 0C046DC6

D619363E 0C00ADC6 D719553C 000007D1 0C004000

0B19553C 00000070 113F

Similarly below the packet starts with 3E for list and it looks objectId from "0C", Property from "19 55" and deviceId from "3C" for first correct reference (DeviceObjectPropertyReference) with all the properties (ObjectId, Property, Array Index and Device ID(optional)) and then if we add next reference as invalid (BOOLEAN) then the packet

will be able to discriminate the next data type after reading the tag '11'(for Boolean) and failed index will be correct as 2.

0A6CAB77 BAC0810A 002B0104 00035F08 0C046DC6

D619363E 0C00ADC6 D719553C 000007D1 11 3F

2. But If we do not specify the optional property(DeviceID) then we are unable to determined whether it's a new reference or optional property.

a. Suppose we have not given any optional property in first correct reference and add another incorrect property with invalid data type as "BOOLEAN" then the packet would be:

0A6CAB77 BAC0810A 001B0104 00035F08 0C046DC6

D619363E 0C00ADC6 D7195511 3F // after property tag"1955" it takes next tag as '11' which is next reference.

In this case again, there is no provision to discriminate whether it's a tag for device ID(Optional property) or it's a new data type added in the list of values. So, it acknowledges it as an optional property and returns the error "INVALID_DATA_TYPE" with index = 1.

So according to the tags if we are not giving any optional property in first correct reference and add an another incorrect property with invalid data type then it always returns the First failed index number as "Index = 1".

So, it is impossible to detect certain types of errors when decoding complex data types..

Questions:

1. Should INVALID_DATATYPE negative testing in DM-LM-B be removed from the BTL testplan?
2. Should the existing BTL tests for INVALID_DATATYPE negative testing in DM-LM-B be amended to restrict testing to avoid the above-described edge case?

Response:

1. No.
2. Yes.

While there are certain data values that will result in valid responses that would fail the test, the test in general still has benefit. In order to help avoid such situations, the BTL-WG will draft advisory language to make testers aware of the impact of invalid parameter selection in negative tests.

The specific example outlined above provides an excellent illustration of the impacts one will see when developing malformed packets. In this particular case, the problem is caused by the last field of the BACnetDeviceObjectPropertyReference construct being optional. To avoid the problem in this specific test, the tester can choose a BACnetList of a different datatype (if one is available), or move the malformity to a later field in the second

BACnetDeviceObjectPropertyReference so that implementations will be able to clearly identify the end of the first element.

Note: there may be other places in the BTL Test Package where the tester will need to be aware in their selection of parameters for which a datatype has been mangled with an invalid datatype or invalid encoding, of the consequences of an optional last component of a constructed datatype.