BACnet *Errata*
ANSI/ASHRAE STANDARD 135-2020
A Data Communication Protocol for Building Automation and Control Networks

April 23, 2021

This document lists all known *errata* to ANSI/ASHRAE Standard 135-2020 as of the above date. Each entry is cited first by clause, then page number, except where an erratum covers more than one clause. The back page marking identifying the electronic publication of Standard 135-2020 is "Product code: D-86451 9/20".

Changes are indicated by using ~~strikeout~~ for text to be removed and *italics* for text to be added, unless noted otherwise. Grey highlighting is used for marking small corrections.

1) **Annex AB.6.3**, p. 1405: Clarify Heartbeat-Request language.

…

An initiating peer shall send a~~A~~ Heartbeat-Request message ~~shall be initiated~~ to the accepting peer if *the initiating peer has not received a*~~no~~ BVLC message ~~was received~~ over the connection within the heartbeat timeout.

On receipt of Heartbeat-Request, the accepting peer shall respond with a Heartbeat-ACK message to the initiating peer.

Although the accepting peer is not required to keep established BACnet/SC connections alive through periodically sending Heartbeat-Request messages to the initiating peer, the accepting peer may send a Heartbeat-Request message to the initiating peer at any time in order to determine whether the initiating peer and the connection is still alive.

…

2) **Annex AB.7.4 and sub-clauses**, p. 1406 - 1407: Clarify terms used for certificates.

AB.7.4 Connection Security

…

The establishment of a secure WebSocket connection shall be performed as defined in RFC 6455. For establishing a secure WebSocket connection, mutual TLS authentication shall be performed. "Mutual authentication" in this context means that both the initiating peer and the accepting peer shall:

(a) Validate that the peer's operational certificate is well formed.

(b) Validate that the peer's operational certificate is active as of the current date and not expired.

(c) Validate that the peer's operational certificate is not revoked, if such information is available.

(d) Validate that the peer's operational certificate is directly signed by one of the locally configured ~~CA~~*issuer* certificates.

…

AB.7.4.1.1 Operational Credentials

Operational credentials include the certificate of a device, the related private key, and the accepted ~~signing CA~~*issuer* certificates that are used to connect to a BACnet/SC network of an installation. A device may have other certificates *and*~~,~~ private keys ~~and signing CA certificates~~ being used for manufacturer specific communication. These credentials are not considered operational credentials and may be considered to be part of the factory default condition of the device. See Clause AB.7.4.2.

Before deployment to an active network, the connection peers shall be configured with ~~a CA~~*an issuer* certificate store containing one or more ~~CA~~*issuer* certificates of those *signing* CAs that are accepted to have signed the peer's certificate, and a unique operational certificate with matching private key. The operational certificate shall be issued and directly signed by a *signing* CA whose ~~CA~~*issuer* certificate is configured in the ~~CA~~*issuer* certificate store. This allows peer-to-peer mutual authentication so that the accepting peer and the initiating peer can each verify that the certificate presented to it was signed by one of the *signing* CAs in its ~~CA~~*issuer* certificate store.

AB.7.4.1.2 Signing CA

The choice of one or multiple *signing* CAs to sign the operational certificates used in a site shall be dictated by site policy. ~~The~~*Each* signing CA shall be controlled by the site and can be a root CA or an intermediate CA.

The signing CAs shall support processing of certificate signing requests in Privacy Enhanced Mail (PEM) format (RFC 7468) conveying a certificate signing request and return the signed certificates in PEM formatted PKCS7 structure.

AB.7.4.1.3 Configuring Operational Certificates

…
For devices that cannot generate their own public/private key pairs, the key pair needs to be generated by a configuration tool. In this case, the tool shall generate the key pair and create a certificate signing request based on certificate parameters defined by the installation. The tool shall submit the certificate signing request to the signing CA for the installation. The ~~signed~~*operational* certificate returned from the *signing* CA, the private key, and the ~~CA~~*issuer* certificates required for the installation are configured into the device by the tool. The private key shall only be transferred in a secured environment, or over communication secured by TLS.

A device that supports an internal security function that allows it to generate and store its private keys by itself is not allowed to expose the private keys, and may not be allowed to accept a private key from a configuration tool. To create a signed operational certificate, the configuration tool provides certificate parameters of the installation to the device and initiates a private key and ~~certificate signing request~~*certificate signing request* generation by the device. The ~~certificate signing request~~ ~~is~~*certificate signing request is* sent to a signing CA of the installation. The ~~signed~~*operational* certificate returned from the *signing* CA, and the ~~CA~~*issuer* certificates ~~for the accepted CAs as~~ required for the installation are configured into the device by the tool.
…

AB.7.4.2.1 Reset to Factory Defaults

…
Performing a reset to "factory defaults" condition shall erase all operational certificates and respective private keys and all ~~CA~~*issuer* certificates from all BACnet/SC network ports. Any sensitive data the device contains shall also be erased. It is not allowed to simply block access to existing sensitive data while in the factory defaults condition because an attacker with physical access can use this condition to insert new operational credentials and then use that false trust relationship to access sensitive data that was not erased.

3) **Annex AB.7.5.1**, p. 1407: Remove undefined error code.

…

Situation	Error Code
~~The security parameters of the client and server do not match.~~	~~TLS_SECURITY_PARAMETER_MISMATCH~~
…	…

…

4) **Clause 12.X.Y Property_List**, Use standard language for Property_List.

This read-only property, *of type*~~is a~~ BACnetARRAY of *BACnetPropertyIdentifier*~~property identifiers~~, *contains* one *BACnetPropertyIdentifier*~~property identifier~~ for each property that exists within the object. The Object_Name, Object_Type, Object_Identifier, and Property_List properties are not included in the list.

5) **Clause 15.9.1.1.5**, p. 752: Priority parameter defined as Unsigned but described as integer.

This parameter, *of type Unsigned,* shall be ~~an integer~~ in the range 1..16, which indicates the priority assigned to this write operation. …

6) **Clause 15.10.3.2.4**, p. 756: Priority parameter defined as Unsigned but described as integer.

This parameter, *of type Unsigned,* shall be ~~an integer~~ in the range 1..16, which indicates the priority assigned to this write operation. …

7) **Clause 21.6**, p. 886: successful-actions-only parameter type incorrectly defined.

…

BACnetAuditLogQueryParameters ::= CHOICE {

by-target	[0] SEQUENCE {		
		target-device-identifier	[0] BACnetObjectIdentifier,
		target-device-address	[1] BACnetAddress OPTIONAL,
		target-object-identifier	[2] BACnetObjectIdentifier OPTIONAL,
		target-property-identifier	[3] BACnetPropertyIdentifier OPTIONAL,
		target-array-index	[4] Unsigned OPTIONAL,
		target-priority	[5] Unsigned(1..16) OPTIONAL,
		operations	[6] BACnetAuditOperationFlags OPTIONAL,
		successful-actions-only	[7] ~~BOOLEAN~~*BACnetSuccessFilter*
	},		
by-source	[1] SEQUENCE {		
		source-device-identifier	[0] BACnetObjectIdentifier,
		source-device-address	[1] BACnetAddress OPTIONAL,
		source-object-identifier	[2] BACnetObjectIdentifier OPTIONAL,
		operations	[3] BACnetAuditOperationFlags OPTIONAL,
		successful-actions-only	[4] ~~BOOLEAN~~*BACnetSuccessFilter*
	}		
}			

…

8) **Clause 21.2.3**, p. 865: start-at-sequence-number parameter type incorrectly defined.

…

AuditLogQuery-Request ::= SEQUENCE {

audit-log	[0] BACnetObjectIdentifier,
query-parameters	[1] BACnetAuditLogQueryParameters,
start-at-sequence-number	[2] *Unsigned64*~~Unsigned32~~ OPTIONAL,
requested-count	[3] Unsigned16
}	

…

9) **Table 12-47**, p. 435: Value_Source and Value_Source_Array missing footnote references.

Table 12-47. Properties of the BitString Value Object Type

Property Identifier	Property Datatype	Conformance Code
…		
Value_Source	BACnetValueSource	O[8,10,12]
Value_Source_Array	BACnetARRAY[16] of BACnetValueSource	O[9,11]
…		

10) **Table 12-4**, p. 179: Analog Value object Audit_Priority_Filter footnote missing Commandable conditionality.

Property Identifier	Property Datatype	Conformance Code

…		
Audit_Level	BACnetAuditLevel	O[12]
Auditable_Operations	BACnetAuditOperationFlags	O[12]
Audit_Priority_Filter	BACnetOptionalPriorityFilter	O[~~12~~*13*]
…		

12 This property shall be present only if the device supports audit reporting.
13 This property shall be present only if Present_Value is commandable and the device supports audit reporting.

11) **Table 12-10**, p. 205: Binary Value object Audit_Priority_Filter footnote missing Commandable conditionality.

Property Identifier	Property Datatype	Conformance Code
…		
Audit_Level	BACnetAuditLevel	O[16]
Auditable_Operations	BACnetAuditOperationFlags	O[16]
Audit_Priority_Filter	BACnetOptionalPriorityFilter	O[~~16~~*17*]
…		

16 This property shall be present only if the device supports audit reporting.
17 This property shall be present only if Present_Value is commandable and the device supports audit reporting.

12) **Table 12-23**, p. 281: Multi-state Value object Audit_Priority_Filter footnote missing Commandable conditionality.

Property Identifier	Property Datatype	Conformance Code
…		
Audit_Level	BACnetAuditLevel	O[14]
Auditable_Operations	BACnetAuditOperationFlags	O[14]
Audit_Priority_Filter	BACnetOptionalPriorityFilter	O[~~14~~*15*]
…		

14 This property shall be present only if the device supports audit reporting.
15 This property shall be present only if Present_Value is commandable and the device supports audit reporting.

13) **Annex AB.6.2.3**, p. 1405: Disconnecting-ACK message should be from the initiating peer.

…
In state **DISCONNECTING**

Disconnect-ACK received
On receipt of a Disconnect-ACK message from the *initiating*~~accepting~~ peer, close the WebSocket connection, and enter the IDLE state.

…

14) **Clause 12.56**, p. 542: Clarifying the Network Port object language for properties not applicable to the Network_Type.

…
~~As specified in Table 12-71 and the text below, some properties of the Network Port object are required if the object is used to represent a network of a given type. For example, a Network Port object whose Network_Type is MSTP and the node is an MS/TP master node shall include the Max_Master property, and a Network Port object whose Network_Type is IPV4 shall include the IP_Subnet_Mask property. Aside from the properties so required, i~~*I*t is a local matter whether a Network Port object contains properties that do not apply to its Network_Type. For example, a Network Port object whose Network_Type is MSTP may include the IP_Subnet_Mask property, although the value of this property would not be used by the network. Some vendors

may find it convenient to have all of their Network Port objects support the same list of properties regardless of Network_Type. *If a property is present but not applicable to the specified Network_Type, then its content is a local matter.* ~~This is permitted, but not required.~~

…

15) **Clause 12.56.9**, p. 549: Clarify Protocol_Level of PROTOCOL includes BACnet and non-BACnet protocols.

This property, of type BACnetProtocolLevel, indicates whether the object represents a physical network interface (PHYSICAL), a *BACnet or* non-BACnet protocol (PROTOCOL), the BACnet use of the protocol (BACNET_APPLICATION), or a non-BACnet use of the protocol (NON_BACNET_APPLICATION).

16) **Clause 12.56.10**, p. 549: Clarify Protocol_Level property levels in the Reference_Port property.

…
If this property has a value of 4194303, then this object has not been assigned a lower protocol layer. If the object is capable of representing all protocol layers in a single object, then this is a valid configuration and the object shall behave as if this property were absent. If the object is not capable of representing all protocol layers in a single object, *and Protocol_Level is not PHYSICAL*, then this is an indication that the object is not yet configured.

If Protocol_Level has a value of PHYSICAL, then this property shall have a value of 4194303.

Object_Identifier	Network Port, 4
Object_Name	BACnet/MSTP on USB1::COM1
Reference_Port	4194303
Protocol_Level	BACNET_APPLICATION
Network_Type	MSTP
Link_Speed	76800
Link_Speeds	9600,38400,76800
Link_Speed_Autonegotiate	FALSE
Network_Interface_Name	USB1::COM1
MAC_Address	1
Max_Master	12
Max_Info_Frames	3
Slave_Proxy_Enable	FALSE
Manual_Slave_Address_Binding	...
Auto_Slave_Discovery	FALSE
Slave_Address_Binding	...
Network_Number	40
Network_Number_Quality	CONFIGURED
APDU_Length	480
Routing_Table	...

Figure 12-18. Example Network Port With No Hierarchy Chain

A Network Port object is misconfigured if the referenced Network Port object has a Protocol_Level of BACNET_APPLICATION, *a Protocol_Level of NON_BACNET_APPLICATION*, or the referenced Network Port object does not exist.

…

12.56.10.1 Network Port Hierarchies

Support for Network Port object hierarchies is optional.

In the normal case, a single hierarchy chain consists of a Network Port object with a Protocol_Level of PHYSICAL at the bottom; one or more Network Port objects with their Protocol_Level set to PROTOCOL, and a Network Port object with a Protocol_Level of BACNET_APPLICATION *or NON_BACNET_APPLICATION* at the top. Multiple Network Port objects can reference a PROTOCOL or PHYSICAL Network Port object.

A Network Port object with a Protocol_Level of BACNET_APPLICATION, *NON_BACNET_APPLICATION*, or PHYSICAL shall not be in the middle of a hierarchy chain.
…

17) **Clause 9.3.10**, p. 102: The BACnet Extended Data Not Expecting Reply frame type is not limited to master nodes.

This COBS-encoded frame is used ~~by master nodes~~ to convey the data parameter of a DL_UNITDATA.request whose data_expecting_reply parameter is FALSE and whose data parameter length is between 502 and 1497 octets, inclusive.

18) **Clause 12.56.34**, p. 558: Missing BACnet_IPv6_Mode in BBMD_Broadcast_Distribution_Table.

This property, of type BACnetLIST of BACnetBDTEntry, is required to be present and writable if BACnet_IP_Mode *or BACnet_IPv6_Mode* is BBMD.
…

19) **Clause 12.56.35**, p. 559: Missing BACnet_IPv6_Mode in BBMD_Accept_FD_Registrations.

This property, of type BOOLEAN, indicates whether (TRUE) or not (FALSE) this device shall accept foreign device registrations. This property is required to be present and writable if BACnet_IP_Mode *or BACnet_IPv6_Mode* is BBMD.
…

20) **Clause 12.56.37**, p. 559: Missing BACnet_IPv6_Mode in FD_BBMD_Address.

This property, of type BACnetHostNPort, indicates the BBMD with which the local device is to register as a foreign device when BACnet_IP_Mode *or BACnet_IPv6_Mode* is FOREIGN. This property shall be present and writable if BACnet_IP_Mode *or BACnet_IPv6_Mode* is FOREIGN.
…

21) **Clause 12.56.38**, p. 559: Missing BACnet_IPv6_Mode in FD_Subscription_Lifetime.

This property, of type Unsigned16, indicates the Time-To-Live value, in seconds, to be used in the Register-Foreign-Device BVLL message. This property shall be present and writable if BACnet_IP_Mode *or BACnet_IPv6_Mode* is FOREIGN.
…

