BSR/ASHRAE Addendum bj to
ANSI/ASHRAE Standard 135-2016

Public Review Draft

Proposed Addendum bj to Standard 135-2016, BACnet® - A Data Communication Protocol for Building Automation and Control Networks

Second Public Review (June 2018)
(Draft shows Proposed Changes to Current Standard)

This draft has been recommended for public review by the responsible project committee. To submit a comment on this proposed standard, go to the ASHRAE website at www.ashrae.org/standards-research--technology/public-review-drafts and access the online comment database. The draft is subject to modification until it is approved for publication by the Board of Directors and ANSI. Until this time, the current edition of the standard (as modified by any published addenda on the ASHRAE website) remains in effect. The current edition of any standard may be purchased from the ASHRAE Online Store at www.ashrae.org/bookstore or by calling 404-636-8400 or 1-800-727-4723 (for orders in the U.S. or Canada).

This standard is under continuous maintenance. To propose a change to the current standard, use the change submittal form available on the ASHRAE website, www.ashrae.org.

The appearance of any technical data or editorial material in this public review document does not constitute endorsement, warranty, or guaranty by ASHRAE of any product, service, process, procedure, or design, and ASHRAE expressly disclaims such.

ASHRAE, 1791 Tullie Circle, NE, Atlanta GA 30329-2305

[This foreword, the table of contents, the introduction, and the "rationales" on the following pages are not part of this standard. They are merely informative and do not contain requirements necessary for conformance to the standard.]

FOREWORD

The purpose of this addendum is to present a proposed change for public review. These modifications are the result of change proposals made pursuant to the ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135. The proposed changes are summarized below.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-2016 and Addenda 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. Only this new and deleted text is open to comment at this time. All other material in this document is provided for context only and is not open for public review comment except as it relates to the proposed changes.

The use of placeholders like XX, YY, ZZ, X1, X2, NN, x, n, ? etc. should not be interpreted as literal values of the final published version. These placeholders will be assigned actual numbers/letters only after final publication approval of the addendum.

[This Table of Contents and the listed clause headers in brackets are provided only for convenience in this addendum. They will not be part of the standard]

Table of Contents

[The following introduction is informative and provided as background information and rationale for this addendum. It will not be part of the standard.]

INTRODUCTION

The BACnet protocol stack as outlined in Clause 4 of the standard was defined before 1995, when the TCP/IP protocol suite was expensive and not available for smaller devices common in building automation. With today's availability of IP network infrastructures for building automation that may be shared with other applications, and may be professionally managed by an IT department, there is a need for a more IT-friendly BACnet solution that allows communicating BACnet across such infrastructures. This introduction first outlines current issues with BACnet in such environments and provides an overview of the solution proposed in this addendum.

Problem Statement

The following sections summarize the issues with BACnet in today's IP-based network infrastructures.

Networking Today and Trends

- Computer networking has become a synonym for IP networks.
- IP is the ubiquitous network infrastructure standard.
- Customers are familiar with and have IP networks in place.
- IP networking is emerging into:
    - All kinds of communication domains.
    - All kinds of media, including wireless (e.g., WPAN, WLAN, Cellular).
    - Field devices (Internet of Things, wired & wireless, CoRE).
    - Mobile devices.
- In the controls domain, network standards are rapidly moving to IP-centric solutions.
- IPv6 is emerging.
- The Building Automation (BA) market is demanding simple plug-and-play devices.
- The size of internetworked systems is increasing.

BACnet needs to be enabled for shared and managed IP network infrastructures!

BACnet Issues in Shared IP Infrastructures

Building Automation Systems using BACnet may be required to use an IP infrastructure that is shared with office and other applications. Such infrastructures are typically managed by IT departments, for whom BACnet is an unknown protocol. BACnet/IP and BACnet/IPv6 have features and behaviors that are not well accepted by IT departments. As a result, the current application of BACnet has multiple issues from an IT perspective:

- Does not follow standards and behaviors acceptable by IT departments.
- Data security is not suitable for IT networks because it is not based on widely used standards such as Transport Layer Security (TLS).
- Demand for fixed IP addresses, in particular for BBMDs.
- Broadcasts that may propagate through the entire network are not acceptable.
- The use of BACnet routers is perceived as adding some extra routing to IP networks that is not manageable by the IT department.
- Excessive use of IT administered IP addresses may result in high infrastructure lease costs.

Values of BACnet that Need to be Preserved

- Designed for control, operation, and monitoring of BA domains.
- Powerful data & services model that reaches into semantic definitions.
- Interoperability among versions and vendors.
- Large installed base.
- Scalability (including support of inexpensive single twisted pair wired networks).
- Comprehensiveness of the network security architecture.

Features that Need to be Enabled for BACnet

- Enable the use of IP networks in a way suitable for highly managed IP infrastructures.
- Enable the use of standard IP application protocols, such as HTTP and WebSockets.
- Enable seamless and simple traversal of typical IP network hurdles, such as NATs and firewalls.
- Enable the use of IP infrastructure that is built for and is shared with office and other applications.
- Enable the use of the same networking infrastructure that is used for complementary systems, such as smart grid and enterprise applications.
- Enable the use of standard IP mechanisms for auto-configuration, name resolution, information security, and device discovery.
- Integration into management of IP network infrastructures (IT-managed environments).
- BACnet communication being completely agnostic to underlying IPv4 or IPv6 flavor of transport layer (even in mixed-scenarios).

Proposed Solution

This addendum proposes a new datalink layer option that makes full use of TLS secured WebSocket connections. This new BACnet Secure Connect datalink layer option uses a virtual hub-and-spoke topology where the spokes are WebSocket connections from the nodes to the hub.

Introduction of the BACnet Secure Connect Datalink Layer Option

The new BACnet Secure Connect, or BACnet/SC, datalink layer option enables full compatibility with all other datalink options of BACnet. The regular BACnet Network Layer routing function connects BACnet devices implementing existing datalink layer options with those implementing BACnet Secure Connect. However, since BACnet Secure Connect is based on TLS secured WebSocket connections, and WebSocket URIs are used for network level addressing, the most urgent features for compliance with IT infrastructure requirements are provided through this datalink option.

Among those IT infrastructure requirements are:

- No specific IT configurations for building automation. The use of the WebSockets protocol which is based on HTTP conforms to typical firewall configurations. No extra configuration by IT is required.
- Secured communication. The use of TLS to secure the WebSocket connections supports sufficient privacy and security for building automation communication in shared IT infrastructures.
- DNS name resolution and DHCP supported. The use of WebSocket URIs includes the option for DNS host names and their resolution to IP addresses. Through this, DHCP based IP configuration is enabled.
- Works over IPv4 and over IPv6. The BACnet/SC microprotocol does not depend on the IP version and respective address format.
- Enable use of standard NATs. TCP based protocols such as the WebSocket protocol enable standard NAT procedures for TCP connections both in IPv4 and IPv6. The TCP port is not relevant.

IP/IT management functionality such as SNMP may be needed so as to enable minimal common IT network management functionality even in multi-vendor installations. The definition of IP/IT management functionality is out of scope for this addendum.

The minimum BACnet/SC implementation is expected to work on relatively small device platforms.

The BACnet/SC datalink option supports the existing Clause 24 security architecture. But more important, it supports the use of Transport Layer Security (TLS) for securing the BACnet communication. Therefore, the BACnet/SC virtual link control messages do not require a Clause 24 security wrapper option. However, BACnet/SC messages allow conveying authentication data such as the User ID and Role ID.

The support of PKI and X.509 certificates enable strong security for all nodes and messages within a given BACnet/SC network.

Through these features, BACnet/SC supports a variety of shared IP infrastructures, such as:

- Use of standard IT provided infrastructure.
- Use in managed IT environments. Support standard IT network management workflows and policies.
- Use in non-managed IT environments.
- Use in BAS installed IT environments, by BAS field engineers.
- Use in environments that are shared with other than building automation applications (web servers, enterprise applications, office applications, factory automation, etc.).
- Use in environments with limited availability of IT managed IP addresses.

The security features of BACnet/SC provide sufficient security to BACnet communication for a range of security environments:

- Use in open and unprotected environments.
- Use in environments that provide different levels of protection.
- Use for building automation applications that require different levels of security.
- Use in environments that are required to comply with security policies and security equipment in place.
- Changing security requirements over system life-time (e.g., commissioning under factory credentials, secured operation with operational credentials, increased security, new algorithms, key lengths, etc.).

135-2016*bj*-1. Introduce BACnet Secure Connect Datalink Layer Option

Rationale

The current BACnet protocol architecture uses various BACnet specific and IP based datalinks. The IP based datalinks (BVLL and BVLLv6) have properties and behaviors that do not always fit with the requirements, policies, and constraints of IT departments that administer and manage IP network infrastructures.

The need for using standardized and often already present IP network infrastructures for BACnet communication is increasing. Based on existing and new technologies, these IP infrastructures are currently reaching out both to small and constrained devices such as sensors and actuators (Internet of Things), but also into wide-area connectivity and cloud based applications.

The BACnet set of datalink options is extended to facilitate the use of the WebSocket protocol in IT environments. A new BACnet Secure Connect, or BACnet/SC, datalink layer option is added which is specifically designed to meet the requirements of minimally managed to professionally managed IP infrastructures. This also includes the defined application of standard IP technologies for network security, such as TLS.

The use of VMACs for datalink addressing enables seamless communication with devices implemented on other datalinks.

[Insert new entries to **Clause 3.2**, preserving the alphabetical order, p. 2]

BACnet address: address format used by the BACnet network layer, as defined in Clause 6, consisting of a network number and a MAC address.

BACnet network: a network of BACnet devices that share the MAC or VMAC address space under a particular BACnet network number.

[Change entries in **Clause 3.2**, p. 2]

directly connected network: a *BACnet* network that is accessible from a *BACnet* router without messages being relayed through an intervening *BACnet* router. A PTP connection is to a directly connected network if the PTP connection is currently active and no intervening *BACnet* router is used.

half router: a device or node that can participate as one partner in a PTP connection. The two half-router partners that form an active PTP connection together make up a single *BACnet* router.

internetwork: a set of two or more *BACnet* networks interconnected by *BACnet* routers. In a *BACnet* internetwork *interconnected by BACnet routers*, there exists exactly one message path between any two nodes.

inverted network: a BACnet internetwork where two or more *BACnet* networks are connected by a *BACnet* network with an NPDU size smaller than the networks it joins.

local broadcast: a message addressed to all devices or nodes on the same *BACnet* network as the originator.

network: a set of one or more segments interconnected by bridges that have the same network address.

remote broadcast: a message addressed to all devices or nodes on a different *BACnet* network than the originator.

virtual BACnet network: a *BACnet* network of virtual BACnet devices, usually modeled by a gateway where no physical BACnet network exists.

[Insert new entries to **Clause 25**, preserving the alphabetical order, p. 932]

IETF RFC 5246 (2008), The TLS Protocol Version 1.2, Internet Engineering Task Force

IETF RFC 5289 (2008), TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM), Internet Engineering Task Force

IETF RFC 6455 (2011), The WebSocket Protocol, Internet Engineering Task Force

IETF RFC 6762 (2013), Multicast DNS, Internet Engineering Task Force

IETF RFC 7251 (2014), AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS, Internet Engineering Task Force

[Change **Clause 4.1**, p. 12]

4.1 The BACnet Collapsed Architecture

BACnet is based on a four-layer collapsed architecture that corresponds to the physical, data link, network, and application layers of the OSI model as shown in Figure 4-2. The application layer and a simple network layer are defined in the BACnet standard. BACnet provides the following options that correspond to the OSI data link and physical layers.

Ethernet (ISO 8802-3)	Clause 7
ARCNET (ATA 878.1)	Clause 8
MS/TP	Clause 9
PTP	Clause 10
LonTalk (ISO/IEC 14908.1)	Clause 11
BACnet/IP	Annex J
BACnet/IPv6	Annex U
ZigBee	Annex O
BACnet/SC	*Annex YY*

Collectively these options provide a master/slave MAC, deterministic token-passing MAC, high-speed contention MAC, dial-up access, *Internet access,* star and bus topologies, and a choice of twisted-pair, coax, or fiber optic media, in addition to wireless connectivity.

. . .

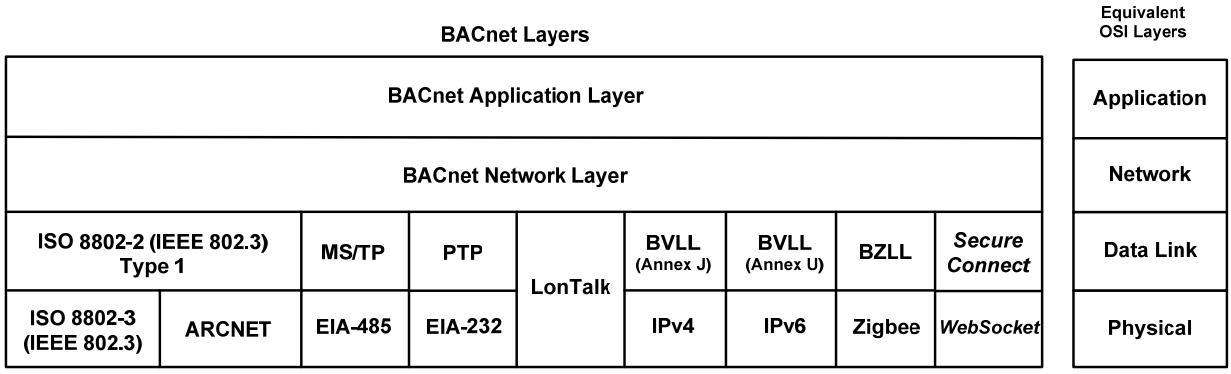
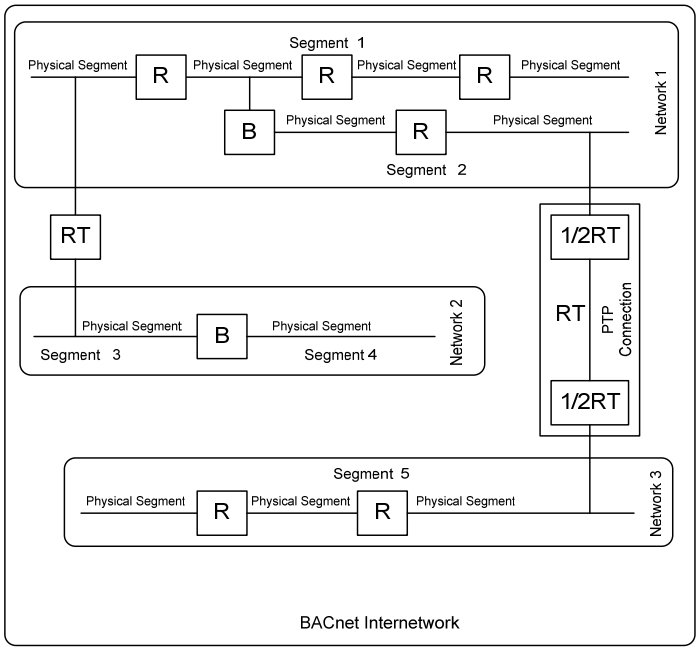


Figure 4-2. BACnet collapsed architecture

The physical layer provides a means of connecting the devices and transmitting the electronic signals that convey the data. Clearly the physical layer is needed in a BAC protocol.

. . .

[Replace **Figure 4-3**, p. 15]
[Current Figure 4-3, to be removed:]



B = Bridge
R = Repeater
RT = Router
1/2RT = Half Router

[New Figure 4-3:]

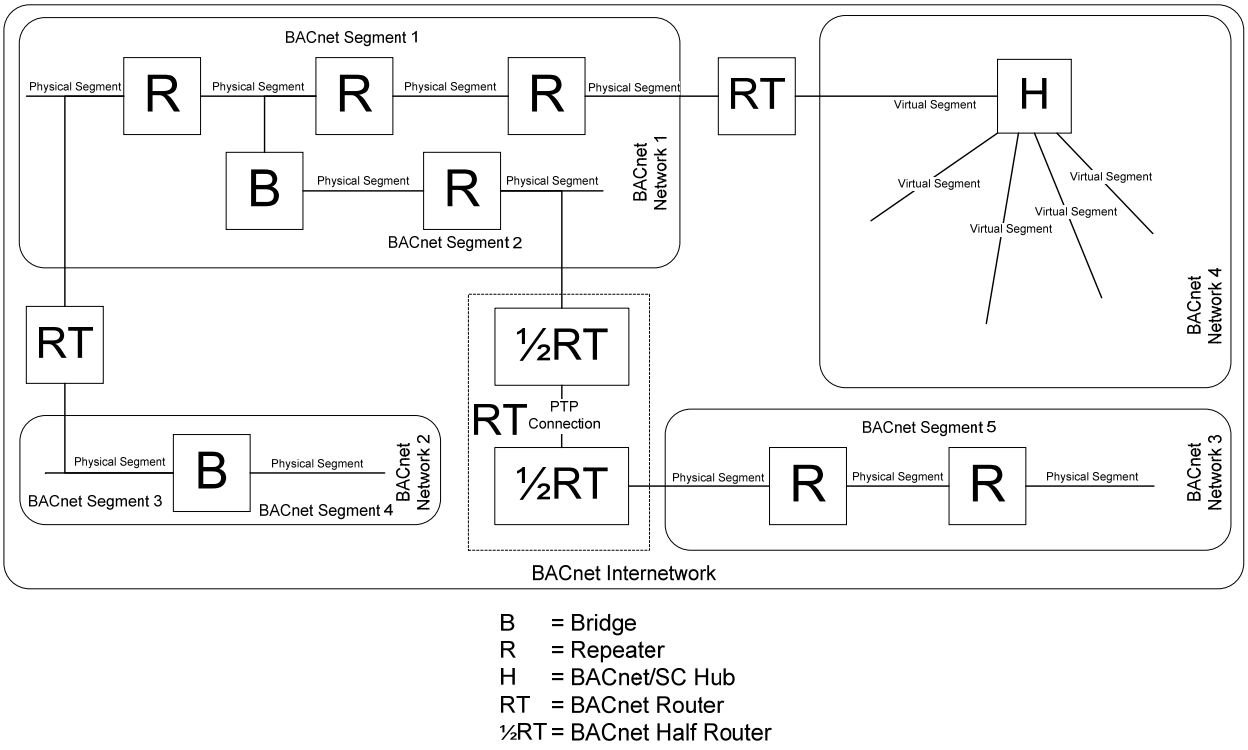


Figure 4-3. A BACnet internetwork, illustrating the concepts of Physical Segments, *Virtual Segments*, Repeaters, *Bridges*, *BACnet/SC Hubs*, *BACnet* Segments, ~~Bridges,~~ *BACnet* Networks, *BACnet* Half Routers, and *BACnet* Routers.

[Change **Clause 4.3**, p. 12]

4.3 Security

The principal security threats to BACnet systems are people who, intentionally or by accident, modify a device's configuration or control parameters. Problems due to ~~an errant~~ *a malfunctioning or misconfigured* computer are outside the realm of security considerations. One important place for security measures is the operator-machine interface. Since the operator-machine interface is not part of the communication protocol, vendors are free to include password protection, audit trails, or other controls to this interface as needed. In addition, write access to any properties that are not explicitly required to be "writable" by this standard may be restricted to modifications made only in virtual terminal mode or be prohibited entirely. This permits vendors to protect key properties with a security mechanism that is as sophisticated as they consider appropriate.

It is recommended that BACnet devices support updating of the device's firmware and software. The procedures for firmware and software upgrades are a local matter.

BACnet also defines services that can be used to provide peer entity, data origin, and operator authentication. See Clause 24.

For the BACnet/SC datalink layer option, standard network security mechanisms based on Transport Layer Security (TLS) are used to provide peer authentication, message integrity, and encryption for communication within a BACnet/SC network. See Annex YY.

135-2016*bj*-2. Introduce BACnet/SC in the Network Layer Specifications

Rationale

The BACnet Secure Connect datalink layer option is introduced in the respective network layer tables.

[Change **Table 6-1**, p. 54]
[Note to reviewer: The APDU header parameter 'max-apdu-length-accepted' is also extended for larger APDUs. See Section 5 of this addendum.]

. . . Another common network layer function is message segmentation and reassembly. To obviate the need for these capabilities at the network layer, BACnet imposes a limitation on the length of the NPDU in messages passed through a BACnet router. The maximum NPDU length shall not exceed the capability of any data link technology encountered along the path from source to destination. A list of the maximum NPDU lengths for BACnet data link technologies is given in Table 6-1.

Table 6-1. Maximum NPDU Lengths When Routing Through Different BACnet Data Link Layers

Data Link Technology	Maximum NPDU Length
ARCNET (ATA 878.1), as defined in Clause 8	501 octets
BACnet/IP, as defined in Annex J	1497 octets
BACnet/IPv6, as defined in Annex U	1497 octets
Ethernet (ISO 8802-3), as defined in Clause 7	1497 octets
LonTalk, as defined in Clause 11	228 octets
MS/TP, as defined in Clause 9	1497 octets
Point-To-Point, as defined in Clause 10	501 octets
ZigBee, as defined in Annex O	501 octets
BACnet/SC, as defined in Annex YY	*65533 octets*[1]

[1] *BACnet/SC network ports of BACnet routers shall support at least 1497 octets.*

. . .

[Change **Table 6-2**, p. 5]

Table 6-2. BACnet DADR and SADR Encoding Rules Based Upon Data Link Layer Technology

BACnet Data Link Layer	DLEN	SLEN	Encoding Rules
ARCNET, as defined in Clause 8	1	1	Encoded as in their MAC layer representations
BACnet/IP, as defined in Annex J	6	6	Encoded as specified in Clause J.1.2
BACnet/IPv6, as defined in Annex U	3	3	Encoded as specified in Clause H.7.2
Ethernet, as defined in Clause 7	6	6	Encoded as in their MAC layer representations
LonTalk domain wide broadcast	2	2	The encoding for the SADR is shown in Figure 6-3 The encoding for the DADR is shown in Figure 6-4
LonTalk multicast	2	2	
LonTalk unicast	2	2	
LonTalk, unique Neuron_ID	7	2	
MS/TP, as defined in Clause 9	1	1	Encoded as in their MAC layer representations
ZigBee, as defined in Annex O	3	3	Encoded as specified in Clause H.7.2
BACnet/SC, as defined in Annex YY	*3*	*3*	*Encoded as specified in Clause H.7.2*

135-2016*bj*-3. Add new Annex YY for the BACnet Secure Connect Datalink Layer Option

Rationale

The BACnet Secure Connect (BACnet/SC) datalink is a new datalink layer option addressing modern IP infrastructure and IT security requirements. The use of the WebSocket protocol (RFC 6455) with Transport Layer Security (TLS) enables the secure exchange of NPDU packets across a wide range of non-managed to tightly managed IT environments. State-of-the-art TLS enables strong information security for BACnet communication.

The BACnet/SC datalink is based on a logical hub-and-spoke model where the spokes are bi-directional, WebSocket protocol-based connections between regular nodes and a hub node. Both regular and hub nodes contain a switch function that handles WebSocket connections. In the hub node, the switch function forwards unicast and broadcast NPDUs between connected regular nodes.

For the sake of efficiency, regular nodes may bypass the hub node for the purpose of sending unicast NPDUs by establishing direct WebSocket connections between each other.

BACnet/SC supports redundancy for the switch function of hub nodes via the concept of a failover hub node that can be deployed alongside the primary hub node.

[Add new **Annex YY, BACnet Secure Connect,** p. 1348]

ANNEX YY - BACnet Secure Connect (NORMATIVE)

(This annex is part of this standard and is required for its use.)

This annex defines a data link protocol by which BACnet devices can transfer messages utilizing the WebSocket protocol as specified in RFC 6455. The Request For Comments (RFC) documents that define the WebSocket protocol are maintained by the Internet Engineering Task Force (IETF).

YY.1 BACnet Secure Connect Datalink

The BACnet Secure Connect, or BACnet/SC, datalink layer specifies a microprotocol enabling the use of WebSocket based connections, specifically the TLS-secured variant, for the exchange of BACnet messages between nodes.

The BACnet/SC microprotocol version defined in this annex is version 1.

The logical topology of a BACnet/SC network generally follows a hub-and-spoke model consisting of multiple BACnet/SC regular nodes and a single BACnet/SC hub node. See Figure YY-1. Both node types manage WebSocket connections via their switch functions. A BACnet/SC regular node wishing to participate in a BACnet/SC network establishes a WebSocket connection to a BACnet/SC hub node. The hub node's switch function accepts WebSocket connections from regular nodes and forwards unicast and broadcast messages to other connected regular nodes.

Optionally, for transmitting unicast BACnet messages, BACnet/SC regular nodes may support direct WebSocket connections with other BACnet/SC regular nodes on the same BACnet network, as an initiator and WebSocket client, or as an acceptor and WebSocket server. These direct WebSocket connections shall be used for unicast messages only; BACnet/SC nodes shall send BACnet broadcast messages to BACnet/SC hub nodes only.

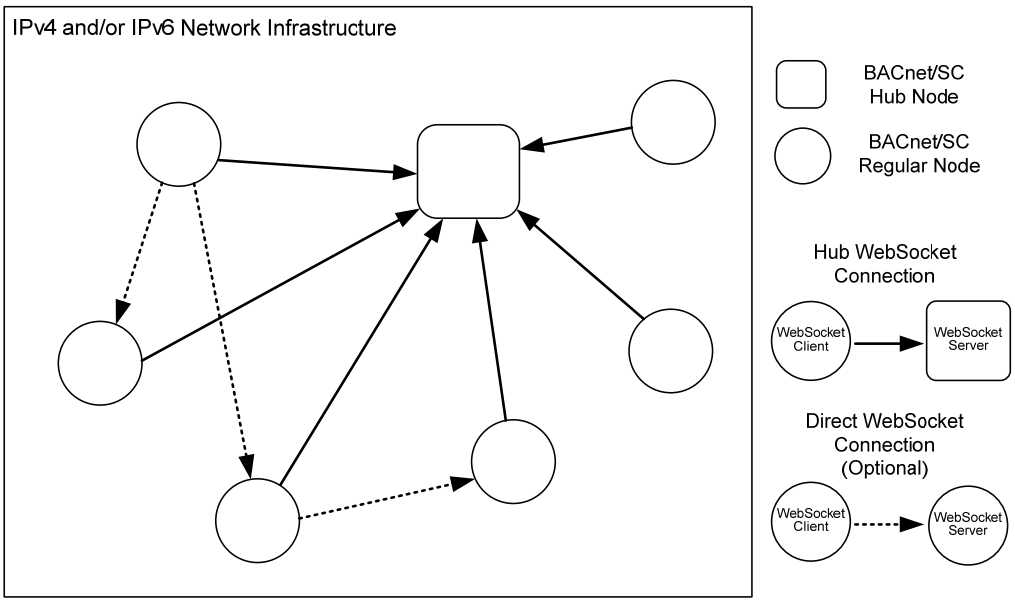


Figure YY-1. BACnet/SC Logical Network Topology

For enhanced availability of the central hub function for a BACnet/SC network, Clause YY.3 specifies a failover hub concept in which the failover hub takes over the hub function in the case that the primary hub node is not available. At any given time, either the primary or the failover hub node performs the hub function. The hub node performing the hub function is referred to as the active hub. Inactive hub nodes perform the basic switch function of regular nodes.

YY.1.1 BACnet/SC Nodes

A BACnet/SC node shall support at least one of the following node types: regular node, primary hub node, or failover hub node. A BACnet/SC node of any type is referred to as 'node' in this annex.

Nodes supporting multiple node types shall support configuration of its node type through the BACnet_SC_Node_Type property of the Network Port object. See Clause 12.56.

Each node implements a BACnet/SC Virtual Link Layer (BVLL) entity for link control and NPDU transport, and the BACnet/SC switch function for dispatching BVLC messages among established WebSocket connections and the local BVLL entity.

Figure YY-2 illustrates two examples of devices implementing BACnet/SC.

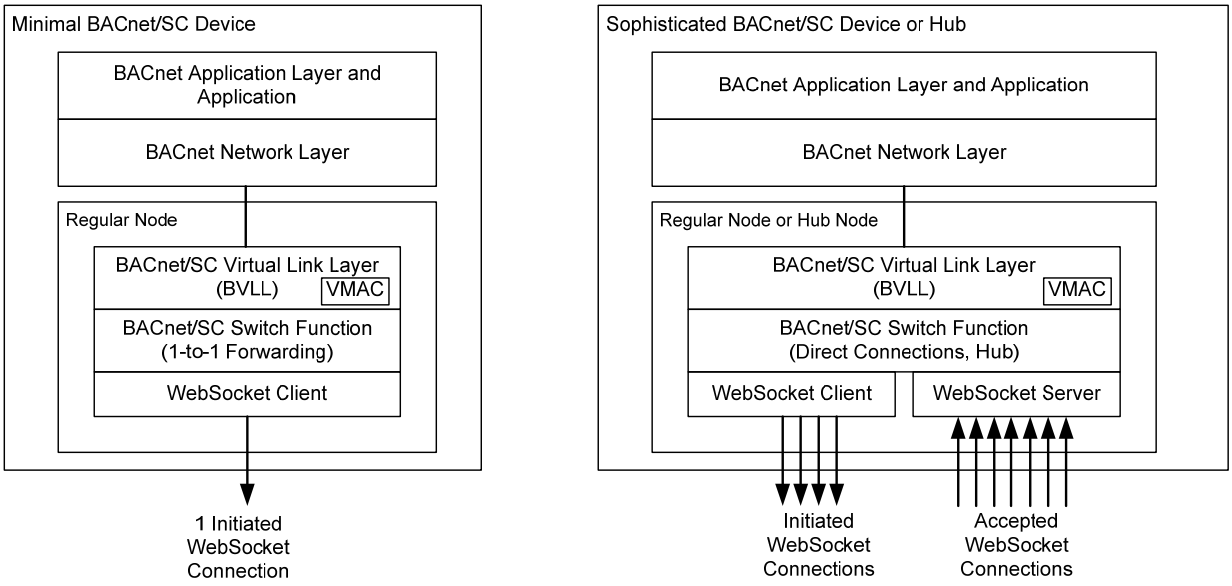


Figure YY-2. Example BACnet/SC Devices

The BVLL for BACnet/SC defines the BACnet/SC Virtual Link Control (BVLC) messages that are used to control the virtual link and to convey BACnet NPDUs. The BVLL entity of a network port implements the BACnet/SC node and is identified on the network by the VMAC of the node. See Clause YY.2.

The BACnet/SC switch function is the logical endpoint of all WebSocket connections initiated or accepted by the BACnet/SC node. Every BACnet/SC node shall support the basic switch function which dispatches messages from the local BVLL entity to a WebSocket connection and from WebSocket connections to the local BVLL entity. For the basic switch function, see Clause YY.5.1. In the minimal case, only one initiated WebSocket connection to the active hub is maintained at a time, and the switch function forwards all messages between the BVLL entity and the single WebSocket connection. See the Minimal BACnet/SC Device example in Figure YY-2.

A BACnet/SC hub supports the hub switch function that forwards unicast messages received from one WebSocket connection to another WebSocket connection and distributes broadcast messages to all WebSocket connections. A hub node performs its hub function only when it is the active hub. See Clause YY.5.2.

For enhanced availability of the hub function for a BACnet/SC network, a BACnet/SC primary hub node, referred to as the primary hub, and a BACnet/SC failover hub node, referred to as the failover hub, are defined. Only one of these hub nodes is the active hub at a time. All BACnet/SC regular nodes shall be capable of initiating a WebSocket connection to the primary hub, support the failover procedures, and be able to initiate a WebSocket connection to the failover hub. See Clause YY.3.

For the use of secure WebSocket connections for BACnet/SC, see Clause YY.7.

YY.1.2 Addressing within BACnet/SC Networks

YY.1.2.1 Network Location of Nodes

The address of hub nodes and regular nodes that accept WebSocket connections for BACnet/SC are specified by WebSocket URIs as defined by the "wss" URI scheme in RFC 6455, Section 3.

For direct WebSocket connections between nodes, the WebSocket URI of a node can be requested by initiating an Address-Resolution BVLC message sent to that node through the active hub. The Address-Resolution-ACK BVLC message in response provides the WebSocket URI of that node.

Optionally, the WebSocket URIs for directly connecting to nodes may be manually configured and take precedence over what is received in Address-Resolution-ACK messages from the nodes. For example, in case the node is in a different IP context and the WebSocket URI returned is not useable. See Clause YY.6.3.4.

YY.1.2.2 VMAC Addressing of Nodes

For the BVLC message exchange, BACnet/SC nodes are identified by their 3-octet virtual MAC address as defined in Clause H.7.2.

For BACnet/SC nodes not yet configured with a Device ID, a random instance VMAC as defined in Clause H.7.2 shall be used.

For directed BVLC messages whose destination is the node at the other end of a WebSocket connection, the WebSocket Peer VMAC address X'3FFFFF' can be used as the destination VMAC address.

For broadcast BVLC messages that need to reach all nodes of the BACnet/SC network, the destination VMAC address shall be the local broadcast virtual MAC address as defined in Clause H.7.X.

YY.1.3 BACnet/SC Network Definition

A BACnet network based on the BACnet/SC datalink option is referred to as a BACnet/SC network. A BACnet/SC network is a set of two or more BACnet/SC nodes in which all regular nodes connect to the same primary and failover hubs. In a BACnet/SC network, at least the primary hub shall be present. The presence of the failover hub is optional.

Direct WebSocket connections between nodes shall only be established between nodes on the same BACnet/SC network. Nodes using direct WebSocket connections shall remain connected to the active hub. Original broadcast BVLC messages shall always and only be sent to the active hub for distribution.

Only one WebSocket connection shall exist at a time between any two BACnet/SC nodes, regardless of which node initiated or accepted the WebSocket connection.

YY.1.4 Remote MAC Addressing of Devices on BACnet/SC Networks

In BACnet network layer services, application layer services, and in data of type BACnetAddress, the MAC address of a BACnet/SC node is the node's virtual MAC address as defined in Clause YY.1.2.2.

YY.1.5 BACnet/SC Network Port Objects

For both regular and hub nodes, participation in a BACnet/SC network represents a single network port regardless of the number of initiated or accepted WebSocket connections in use by the device within that BACnet/SC network. Each BACnet/SC network port is represented by a single Network Port Object with a Protocol_Level of BACNET_APPLICATION and Network_Type of SECURE_CONNECT and optional subordinate Network Port objects.

As with any network port, the lower layers may be represented in subordinate Network Port objects. In this case, the first subordinate Network Port object shall have a Protocol_Level of PROTOCOL and a Network_Type of WEBSOCKETS. This first subordinate object shall be used by one SECURE_CONNECT Network Port object exclusively.

For BACnet/SC, the WebSocket_Protocol property shall have a value of "bsc.bacnet.org". The WebSocket_Payload_Type property shall have a value of BINARY_PAYLOAD.

For Network Port object property requirements, see Clause 12.56.

YY.2 BACnet/SC Virtual Link Layer Messages

The BACnet/SC Virtual Link Layer (BVLL) provides the interface between the BACnet Network Layer (See Clause 6) and the underlying capabilities of the communication subsystem based on WebSockets (RFC 6455). This annex specifies the BACnet Virtual Link Control (BVLC) functions required to transport BACnet unicast and broadcast NPDU messages, and to control the BVLL operation. The purpose and format of each BVLC message is described in the following subclauses. The BVLL behavior is defined in Clause YY.6.

Directed BVLC messages are addressed to a single destination node. Original broadcast BVLC messages are sent by a regular node to the active hub for distribution to all nodes. Forwarded broadcast BVLC messages are used by the active hub to forward the original broadcast BVLC messages to all nodes, excluding the original sender.

The following table lists the BVLC messages defined for BACnet/SC.

Table YY-1 BACnet/SC BVLC Messages

BVLC Function	BVLC Message Length (if no header options)
X'00' BVLC-Result	15 octets for ACK 20 octets + Details for NAK
X'01' Unicast-NPDU	13 octets + BACnet NPDU
X'02' Original-Broadcast-NPDU	13 octets + BACnet NPDU
X'03' Forwarded-Broadcast-NPDU	13 octets + BACnet NPDU
X'04' Address-Resolution	13 octets
X'05' Address-Resolution-ACK	12 octets + WebSocket URI
X'06' Advertisement	21 octets
X'07' Advertisement-Solicitation	13 octets

YY.2.1 General BVLC Message Format

The following table shows the general BVLC message format for BACnet/SC.

Table YY-2 BACnet/SC BVLC Messages Structure

Field	Length	Description
BVLC Type	1-octet	BVLL for BACnet/SC
BVLC Function	1-octet	BVLC function
Source Virtual Address	3-octets	Source node VMAC address
Destination Virtual Address	3-octets	Destination VMAC address
Message ID	2-octets	The message identifier
Header Options	Variable	0 to N header options
Payload Marker	1-octet	Marks the payload
Payload Length	2-octets	The payload length
Payload	Variable	The payload of the BVLC message

The 1-octet 'BVLC Type' field indicates which underlying communication subsystem or microprotocol is in use. A 'BVLC Type' of X'83' indicates a BVLC message for BACnet/SC as defined in this annex.

The 1-octet 'BVLC Function' field identifies the specific function to be carried out in support of the indicated communication subsystem or microprotocol type.

The 3-octet 'Source Virtual Address' field indicates the VMAC address of the original sending node.

The 3-octet 'Destination Virtual Address' field indicates the VMAC address of the destination node for directed BVLC messages, the WebSocket Peer VMAC address for directed BVLC messages to the node at the other end of

the WebSocket connection, or the local broadcast VMAC address for original and forwarded broadcast BVLC messages destined to all nodes of the BACnet network.

The 2-octet 'Message ID' field is a numeric identifier of the message being sent. The original source node of the message shall assign this numeric identifier to BVLC messages. The message ID may be selected to allow for matching BVLC messages sent with BVLC-Result messages received. The BVLC-Result message shall be sent with the 'Message ID' to which it is a response. Forwarded broadcast BVLC messages shall use the message identifier of the original broadcast BVLC message.

The variable size 'Header Options' field contains zero or more header options. See Clause YY.2.2.

The 1-octet 'Payload Marker' field, equal to X'00', indicates the start of the payload, and shall always be present.

The 2-octet 'Payload Length' field is the length, in octets, of the 'Payload Data' field, including the two octets of the 'Payload Length' field itself. If no payload is defined for the BVLC message, the 'Payload Length' shall be present and indicate a value of X'0002'.

The variable sized 'Payload' parameter is the payload being conveyed by the BVLC message. See BVLC message definitions in Clause YY.2.3 and subsequent clauses.

All multi-octet values are encoded with most significant octet first.

YY.2.2 Header Options

BVLC messages allow conveying header options in addition to defined payloads. The presence of one or more header options is indicated by a 'Header Marker' octet greater than X'00'. Multiple header options may be present in a BVLC message, each starting with a 'Header Marker' greater than X'00'.

Each header option includes a 'Header Marker' providing the 'Must Understand' flag and identifying the type of the option, a 'Header Length' field, and the 'Header Data' for the content of the header.

Header Marker	1-octet	'Must Understand' flag and header option type.
Header Length	2-octets	Length of header option, in octets
Header Data	Variable	Octet string as defined for the header option type.

The 1-octet 'Header Marker' field includes the 'Must Understand' flag in the most significant bit, and indicates the header option type in the seven least significant bits.

For the handling of the 'Must Understand' flag and the processing of header options when sending, forwarding, broadcasting, or receiving BVLC messages with header options, see the respective clause for the header option.

The following table lists the header option types defined by this standard, and assigns the numeric header option type used in the 'Header Marker'.

Table YY-3 BVLC Header Options

Header Option Type	Header Marker	Description
Authentication Data Header	X'01'	See Clause YY.2.2.2
Proprietary Header Option	X'7E'	See Clause YY.2.2.1

All other header option types are reserved for ASHRAE and shall not be used.

The 2-octet 'Header Length' field indicates the length in octets of the 'Header Data', including the two octets for the 'Header Length' field.

The variable size 'Header Data' field is an octet string whose content is defined by the respective header option type indicated by the 'Header Marker'. The 'Header Data' field for proprietary header options shall begin with a 2-octet field indicating the vendor identifier of the organization defining the header option and the remaining data octets.

YY.2.2.1 Proprietary Header Options

Vendors may define and use proprietary header options. In order to distinguish vendor specific header options, the first two octets of the header data shall contain the vendor identifier code of the defining organization. See Clause 23.

Any proprietary header option shall consist of the following fields:

Header Marker	1-octet	X'7E'	'Must Understand' = FALSE, Header option type = 126
Header Length	2-octets		Length of header option, in octets
Proprietary Header Data		Variable length	A string of octets. The first two octets, with most significant octet first, shall be the vendor identifier of the organization defining this option.

For BVLC messages whose payload is passed to the local network entity, the processing of proprietary header options is a local matter.

For NPDU messages received from the local network entity, the insertion of any proprietary header option in the BVLC message to be sent is a local matter.

For the active hub, when directed and broadcast BVLC messages are to be forwarded to other WebSocket connections, any proprietary header option, if present, shall be included in the forwarded message as provided in the original BVLC message.

YY.2.2.2 Authentication Data Header Option

The Authentication Data header option conveys authentication data, allowing a peer BACnet/SC device to identify the user. The authentication data may also be derived from or be forwarded in security wrappers. See Clause 24.

The Authentication Data header option consists of the following fields:

Header Marker	1-octet	X'01'	'Must Understand' = FALSE, Header option type = 1
Header Length	2-octets		Length of header option, in octets
Authentication Data		Variable length	A value of type BACnetAuthenticationData, binary encoded per Clause 20.2.

For BVLC messages whose payload is passed to the local network entity, the authentication data contained in this header option when present shall also be forwarded to the local network entity. If the network entity does not support this option, the header option can be ignored when received.

For NPDU messages received from the local network entity, and authentication data is provided by the local network entity, an Authentication Data header shall be inserted in the BVLC message to be sent, conveying the authentication data provided.

For the active hub, when directed and broadcast BVLC messages are to be forwarded to other WebSocket connections, the Authentication Data header, if present, shall be included in the forwarded message as provided in the original BVLC message.

YY.2.3 BVLC-Result

This directed BVLC message provides a mechanism to acknowledge the result of those BVLL service requests that require an acknowledgment, whether successful (ACK) or unsuccessful (NAK). This message is also used to

indicate if a header option was received that cannot be ignored, but is not supported and thus the message cannot be processed.

YY.2.3.1 BVLC-Result Format

The BVLC-Result message consists of the following fields:

BVLC Type	1-octet	X'83'	BVLL for BACnet/SC
BVLC Function	1-octet	X'00'	BVLC-Result
Source Virtual Address	3-octets		
Destination Virtual Address	3-octets		
Message ID	2-octets		The message identifier of the message for which this message is the result.
Header Options		Variable length	0 to N header options
Payload Marker	1-octet	X'00'	Marks Payload
Payload Length	2-octets		
Result For BVLC Function	1-octet	Function	BVLC function for which this is a result
Result Code:	1-octet	X'00'	ACK: Successful completion. The 'Header Option Type' and all subsequent parameters shall be absent.
		X'01'	NAK: The BVLC function failed. The 'Error Header Marker', the 'Error Class', and the 'Error Code' shall be present.
Error Header Marker (Optional)	1-octet		The header marker of the header option that caused the BVLC function to fail. If the NAK is unrelated to a header option, this parameter shall be X'00'.
Error Class (Optional)	2-octets		The 'Error Class' field of the 'Error' constructed datatype defined in Clause 21.
Error Code (Optional)	2-octets		The 'Error Code' field of the 'Error' constructed datatype defined in Clause 21.
Error Details (Optional)		Variable length	Optional UTF-8 encoded reason text, including a terminating zero octet. Shall only be present if 'Result Code' is NAK.

YY.2.4 Unicast-NPDU

This directed BVLC message is used to send unicast NPDUs to another BACnet/SC node.

YY.2.4.1 Unicast-NPDU Format

The Unicast-NPDU message consists of the following fields:

BVLC Type	1-octet	X'83'	BVLL for BACnet/SC
BVLC Function	1-octet	X'01'	Unicast-NPDU
Source Virtual Address:	3-octets		
Destination Virtual Address	3-octets		
Message ID	2-octets		The message identifier
Header Options		Variable length	0 to N header options
Payload Marker	1-octet	X'00'	Marks Payload
Payload Length	2-octets		
BACnet NPDU		Variable length	

YY.2.5 Original-Broadcast-NPDU

This original broadcast BVLC message is used to send a broadcast NPDU to the active hub of the BACnet/SC network, for distribution to all nodes.

YY.2.5.1 Original-Broadcast-NPDU Format

The Original-Broadcast-NPDU message consists of the following fields:

BVLC Type	1-octet	X'83'	BVLL for BACnet/SC
BVLC Function	1-octet	X'02'	Original-Broadcast-NPDU
Source Virtual Address	3-octets		
Destination Virtual Address	3-octets	X'800000'	Local Broadcast VMAC
Message ID	2-octets		The message identifier
Header Options		Variable length	0 to N header options
Payload Marker	1-octet	X'00'	Marks Payload
Payload Length	2-octets		
BACnet NPDU		Variable length	NPDU being broadcast

YY.2.6 Forwarded-Broadcast-NPDU

This forwarded broadcast BVLC message is used by the active hub to forward a broadcast NPDU to nodes.

YY.2.6.1 Forwarded-Broadcast-NPDU Format

The Forwarded-Broadcast-NPDU message consists of the following fields:

BVLC Type	1-octet	X'83'	BVLL for BACnet/SC
BVLC Function	1-octet	X'03'	Forwarded-Broadcast-NPDU
Source Virtual Address	3-octets		
Destination Virtual Address	3-octets	X'800000'	Local Broadcast VMAC
Message ID	2-octets		'Message ID' of Original-Broadcast-NPDU message
Header Options		Variable length	0 to N header options
Payload Marker	1-octet	X'00'	Marks Payload
Payload Length	2-octets		
BACnet NPDU		Variable length	NPDU being broadcast

YY.2.7 Address-Resolution

This directed BVLC message is unicast by BACnet/SC nodes via the hub in order to determine the WebSocket URI of a known virtual MAC address for establishing a direct WebSocket connection to the destination node.

YY.2.7.1 Address-Resolution Format

The Address-Resolution message consists of the following fields:

BVLC Type	1-octet	X'83'	BVLL for BACnet/SC
BVLC Function	1-octet	X'04'	Address-Resolution
Source Virtual Address	3-octets		
Destination Virtual Address	3-octets		
Message ID	2-octets		The message identifier
Header Options		Variable length	0 to N header options
Payload Marker	1-octet	X'00'	Marks Payload
Payload Length	2-octets	X'0002'	There is no payload for this message

YY.2.8 Address-Resolution-ACK

This directed BVLC message is the reply to the Address-Resolution message. The Address-Resolution-ACK message is unicast to the node that originally initiated the Address-Resolution message. This message is only sent by nodes that accept direct WebSocket connections. Other nodes shall return a BVLC-Result NAK message.

YY.2.8.1 Address-Resolution-ACK Format

The Address-Resolution-ACK message consists of the following fields:

BVLC Type	1-octet	X'83'	BVLL for BACnet/SC
BVLC Function	1-octet	X'05'	Address-Resolution-ACK
Source Virtual Address	3-octets		
Destination Virtual Address	3-octets		
Message ID	2-octets		The message identifier
Header Options		Variable length	0 to N header options
Payload Marker	1-octet	X'00'	Marks Payload
Payload Length	2-octets		
WebSocket URI		Variable length	UTF-8 encoded WebSocket URI of the source BACnet/SC node, including a terminating zero octet.

YY.2.9 Advertisement

This directed BVLC message is advertising the configuration and status of the sending node. This message is sent when an Advertisement-Solicitation message is received, when the hub changes its status, and after a WebSocket connection is established.

YY.2.9.1 Advertisement Format

The Advertisement message consists of the following fields:

BVLC Type:	1-octet	X'83'	BVLL for BACnet/SC
BVLC Function:	1-octet	X'06'	Advertisement
Source Virtual Address:	3-octets		
Destination Virtual Address:	3-octets		
Message ID	2-octets		The message identifier
Header Options		Variable length	0 to N header options
Payload Marker	1-octet	X'00'	Marks payload
Payload Length	2-octets	X'0008'	
B-SC BVLL Version	1-octet	X'01'	The BACnet/SC microprotocol version supported by the node.
Node Type	1-octet	X'00'	BACnet/SC regular node
		X'01'	BACnet/SC primary hub
		X'02'	BACnet/SC failover hub
Hub Active	1-octet	X'00'	The node is not the active hub.
		X'01'	The node is the active hub.
Accept Connections	1-octet	X'00'	The node does not accept WebSocket connections.
		X'01'	The node accepts WebSocket connections.
Maximum BVLC Length	2-octet		The maximum BVLC message size that can be received and processed by the node, in number of octets.
Maximum NPDU Length	2-octets		The maximum NPDU message size that can be handled by the node's network entity, in number of octets.

YY.2.10 Advertisement-Solicitation

This directed BVLC message is unicast to a node to solicit an Advertisement message being returned by the destination node.

YY.2.10.1 Advertisement-Solicitation Format

The Hub-Advertisement-Solicitation message consists of the following fields:

BVLC Type:	1-octet	X'83'	BVLL for BACnet/SC
BVLC Function:	1-octet	X'07'	Advertisement-Solicitation
Source Virtual Address:	3-octets		
Destination Virtual Address:	3-octets		
Message ID	2-octets		The message identifier
Header Options		Variable length	0 to N header options
Payload Marker	1-octet	X'00'	Marks Payload
Payload Length	2-octets	X'0002'	There is no payload for this message

YY.3 Redundancy of BACnet/SC Hubs

BACnet/SC supports redundancy of the hub function through the BACnet/SC primary hub and BACnet/SC failover hub. The primary hub is required for a BACnet/SC network and is used under normal conditions. The failover hub, when present in a BACnet/SC network, becomes active and performs the hub function for the BACnet/SC network when the primary hub is not available. See Figure YY-3.

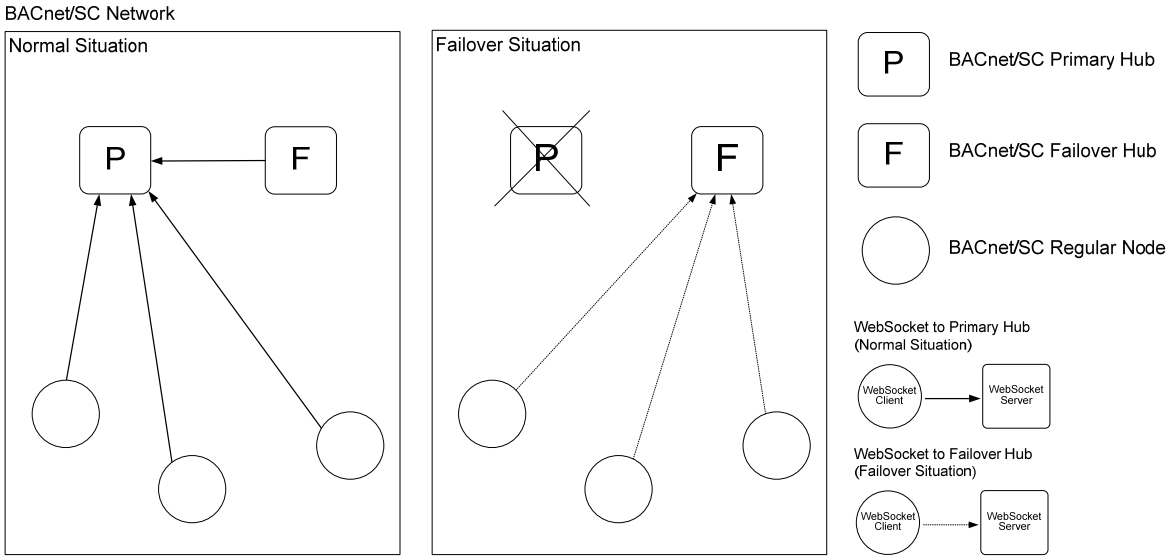


Figure YY-3. BACnet/SC Redundancy of Hub Function

YY.3.1 Configuration for Redundancy

Regular nodes shall support the configuration of the WebSocket URI for the primary hub in the BACnet_SC_Primary_Hub property, and the WebSocket URI for the failover hub in the BACnet_SC_Failover_Hub property of the Network Port object.

Primary hub nodes are not required to support configuration of the BACnet_SC_Primary_Hub property or BACnet_SC_Failover_Hub property.

Failover hub nodes are required to support configuration of the WebSocket URI for the primary hub in the BACnet_SC_Primary_Hub property, and are not required to support configuration of the BACnet_SC_Failover_Hub property.

YY.3.2 Active Hub

When a BACnet/SC hub is active, the hub switch function is enabled and messages are forwarded as a hub. See Clause YY.5.2.

If a BACnet/SC hub indicates it is inactive, it shall behave like a regular node performing the basic switch function only. See Clause YY.5.1.

The primary hub shall be the active hub when ready to accept WebSocket connections and the hub switch function is enabled.

The failover hub shall become the active hub when the WebSocket connection to the primary hub fails, or the primary hub indicates it is inactive.

The failover hub shall stop being the active hub when the WebSocket connection to the primary hub is restored and the primary hub indicates it is active through an Advertisement message sent through the restored WebSocket connection.

YY.3.3 Node Operation for Hub Redundancy

Support for connecting to the primary and failover hub, and to determine the active hub, is required for regular nodes. See Clause YY.6.

YY.4 Direct WebSocket Connections

BACnet/SC supports the use of WebSocket connections between regular nodes of a BACnet/SC network, in addition to the WebSocket connection to the active hub. Such connections are referred to as direct WebSocket connections.

Regular nodes may optionally accept direct WebSocket connections as a WebSocket server, and may optionally initiate direct WebSocket connections to other BACnet/SC nodes as a WebSocket client.

BACnet/SC hubs may optionally accept direct WebSocket connections as a WebSocket server also when inactive, and may optionally initiate direct WebSocket connections to other BACnet/SC nodes as a WebSocket client when inactive.

Direct WebSocket connections are used bi-directionally by the switch function regardless of whether the local node initiated or accepted the direct WebSocket connection.

If a node supports direct WebSocket connections as an initiator and WebSocket client, the method to determine when to initiate a direct WebSocket connection is a local matter.

YY.5 BACnet/SC Switch Function

The BACnet/SC switch function is the logical endpoint for the WebSocket connections of the network port. The switch function forwards BVLC messages between WebSocket connections and the local BVLL entity. All WebSocket connections are bi-directional connections, and shall be used to forward BVLC messages regardless of whether the WebSocket connection was initiated or accepted by the node.

The 'Destination Virtual Address' of BVLC message received from a WebSocket connection, or from the local BVLL entity, is used to select the WebSocket connection or the local BVLL entity to forward a message to.

If multiple WebSocket connections are supported, the switch function shall maintain knowledge of which VMAC is reached through which WebSocket connection, and the basic switch function shall maintain knowledge of which WebSocket connection reaches the active hub.

This information is normally learned from the Advertisement message exchange upon establishment and maintenance of WebSocket connections. An Advertisement can be requested at anytime from any node by sending an Advertisement-Solicitation message. When sending an Advertisement-Solicitation message to the node at the other end of the WebSocket connection without knowing its VMAC address, the WebSocket Peer VMAC address is used as the destination VMAC address.

YY.5.1 Basic Switch Function in Regular Nodes and Inactive Hubs

In all regular nodes and inactive hubs, the switch function dispatches messages between the local BVLL entity and WebSocket connections. See Figure YY-4. The time of establishment of direct WebSocket connections is a local matter. See Clause YY.4.

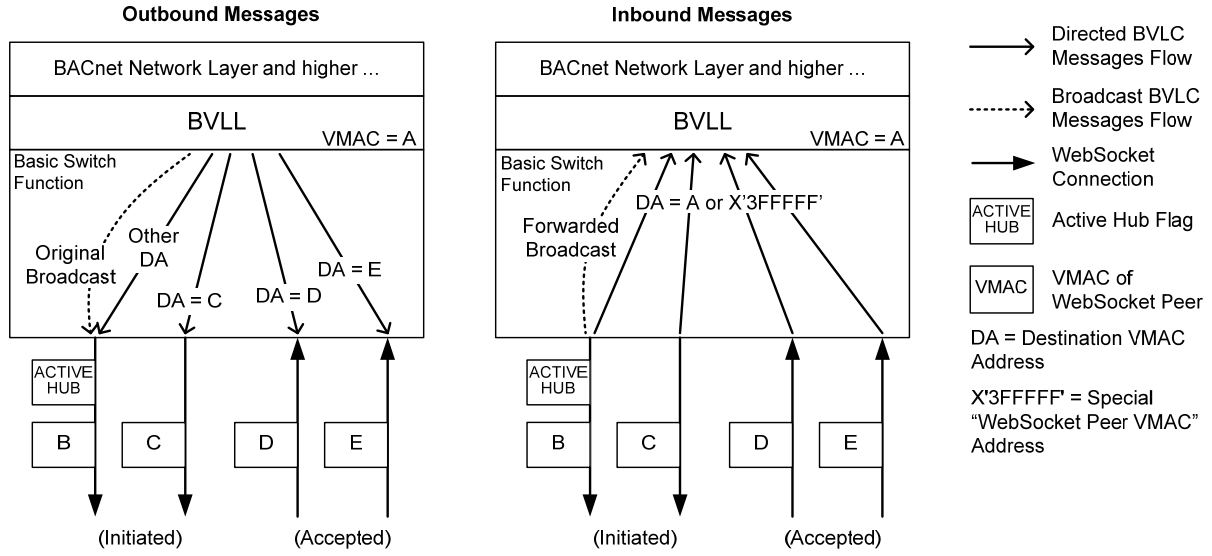


Figure YY-4. Basic Switch Function

On receipt of a directed BVLC message from the local BVLL entity, the directed BVLC message shall be sent through the WebSocket connection to the destination VMAC, if one exists. If no such WebSocket connection exists or is established, the directed BVLC message shall be forwarded to the active hub.

On receipt of a directed BVLC message from the local BVLL entity where the destination VMAC address is the WebSocket Peer VMAC address, the WebSocket connection to use to send the message is not implied. Therefore, the WebSocket connection to use is required to be indicated by the local BVLL entity. The method of indication is a local matter. For the WebSocket Peer VMAC address, see Clause YY.1.2.2.

On receipt of an original broadcast BVLC message from the local BVLL entity, the message is forwarded to the active hub. If no WebSocket connection to an active hub exists, the broadcast BVLC message shall be discarded.

On receipt of a directed BVLC message from any current WebSocket connection whose destination VMAC is the VMAC of the local BVLL entity or is the WebSocket Peer VMAC address, the message shall be forwarded to the local BVLL entity. All other directed BVLC messages shall be discarded.

On receipt of a forwarded broadcast BVLC message from the WebSocket connection to the active hub, the message shall be forwarded to the local BVLL entity.

On receipt of a forwarded broadcast BVLC message from a WebSocket connection to another node than to the active hub, the message shall be discarded.

On receipt of an original broadcast BVLC message from a WebSocket connection, the message shall be discarded. The BVLL shall return a BVLC-Result NAK to the sender of the original broadcast BVLC message indicating an 'Error Code' of COMMUNICATION and an 'Error Class' of NOT_THE_ACTIVE_HUB.

YY.5.2 Hub Switch Function in the Active Hub

In active hubs, in addition to dispatching messages to and from the local BVLL entity, forwarding and distribution of BVLC messages among and between the WebSocket connections is required. This hub switch functionality is only performed while the hub is active. If the hub is inactive, only the basic switch function is performed. See Clause YY.5.1. For active and inactive hubs, see Clause YY.3.2. Figure YY-5 illustrates the hub switch function concept, showing the essential directed and broadcast BVLC message flows.

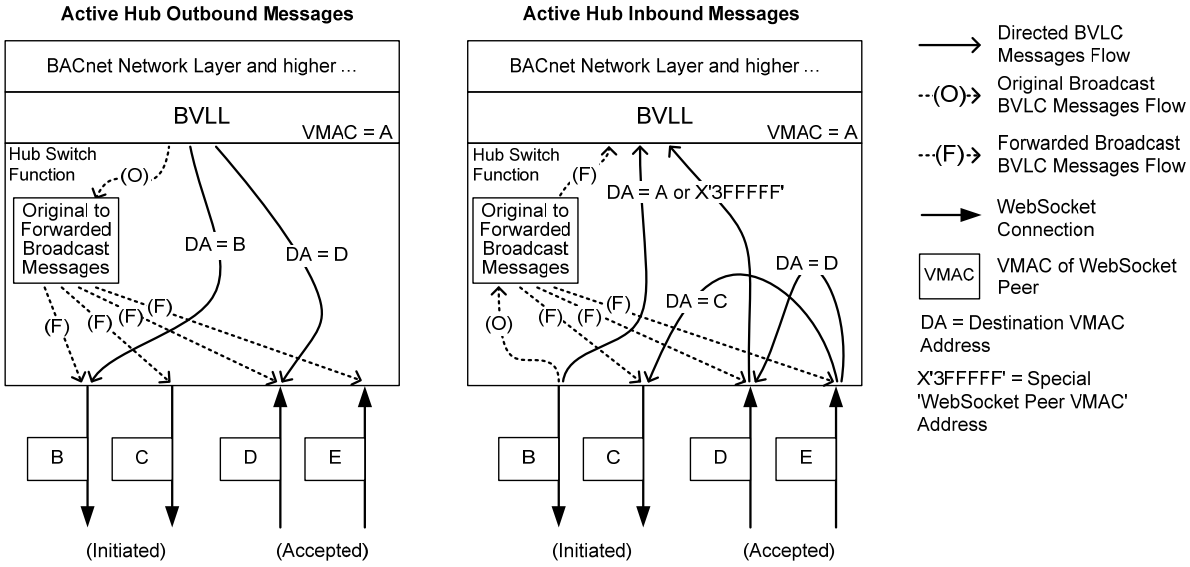


Figure YY-5. Hub Switch Function

On receipt of a directed BVLC message from the local BVLL entity, the directed BVLC message shall be sent through the WebSocket connection to the node identified by the destination VMAC, if a WebSocket connection exists. If no such WebSocket connection exists or is established, the directed BVLC message shall be discarded.

On receipt of a directed BVLC message from the local BVLL entity and the destination VMAC address is the WebSocket Peer VMAC address, the WebSocket connection to use to send the message is not implied. Therefore, the WebSocket connection to use is required to be indicated by the local BVLL entity. The method of indication is a local matter. For the WebSocket Peer VMAC address, see Clause YY.1.2.2.

On receipt of an original broadcast BVLC message from the local BVLL entity, the message shall be converted into the respective forwarded broadcast BVLC message, and a copy shall be forwarded through each current WebSocket connection.

On receipt of a directed BVLC message from any current WebSocket connection whose destination VMAC is the VMAC of the local BVLL entity or is the WebSocket Peer VMAC address, the message shall be forwarded to the local BVLL entity.

On receipt of a directed BVLC message from any current WebSocket connection whose destination is not the local BVLL entity, the message shall be forwarded to the WebSocket connection to the destination VMAC, if one exists. If no such WebSocket connection exists, the directed BVLC message shall be discarded.

On receipt of an original broadcast BVLC message from any current WebSocket connection, the message shall be converted into the respective forwarded broadcast BVLC message, and a copy shall be forwarded through all current WebSocket connections, except to the WebSocket connection the original broadcast BVLC message was received from. A copy of the forwarded BVLC message shall be provided to the local BVLL entity.

On receipt of a forwarded broadcast BVLC message from any current WebSocket connection, the message shall be discarded by the active hub.

YY.6 BACnet/SC Node Operation

YY.6.1 Network Port Disable

If the Network Port object's Out_Of_Service property has a value of TRUE, the BACnet/SC network port shall disable all communication through the port, disconnect all WebSocket connections, and indicate a status of DISABLED in the BACnet_SC_Hub_Status property, and indicate a value of zero in both the BACnet_SC_Primary_Hub_Connection and BACnet_SC_Failover_Hub_Connection property.

When Out_Of_Service becomes FALSE, the node shall reconnect to the network following the procedures in Clause YY.6.3.

YY.6.2 Node Configuration

A BACnet/SC node is configured through the properties of the Network Port object that represents the connection to the BACnet/SC network. See Clause 12.56.

The type of the BACnet/SC node is configured in the BACnet_SC_Node_Type property.

The BACnet_SC_Primary_Hub and the BACnet_SC_Failover_Hub properties are used to configure the WebSocket URI to connect to the primary hub and to the failover hub.

For a node type of REGULAR_NODE, the node is a regular node and the BACnet_SC_Primary_Hub property contains the WebSocket URI for the primary hub, and the BACnet_SC_Failover_Hub property may contain the WebSocket URI for the failover hub.

For a node type of PRIMARY_HUB, the node shall be the primary hub and the value of the BACnet_SC_Primary_Hub property shall be ignored. The value of the BACnet_SC_Failover_Hub property shall be ignored.

For a node type of FAILOVER_HUB, the node shall be the failover hub and the value of the BACnet_SC_Failover_Hub property shall be ignored. The BACnet_SC_Primary_Hub property shall contain the WebSocket URI for the primary hub.

The keep-alive timeout to be applied for all initiated WebSocket connections is configured in the BACnet_SC_Keep_Alive property.

YY.6.3 WebSocket Connection Management

It is the regular node's responsibility to ensure a WebSocket connection between itself and the active hub.

Therefore, regular nodes are required to be capable of establishing a WebSocket connection to the active hub, as a WebSocket client. Optionally, a node may initiate direct WebSocket connections to other nodes as a WebSocket client, or accept such direct WebSocket connections as a WebSocket server. See Clause YY.4.

For the BACnet/SC switch function, a WebSocket connection, once established, is a bi-directional logical connection. Only one WebSocket connection between two nodes shall exist at a time.

For the application of the WebSocket protocol for BACnet/SC, see Clause YY.7.

YY.6.3.1 WebSocket Connection Status Indication

For the indication of the status of a WebSocket connection in the Network Port object's BACnet_SC_Connections property, the BACnetSCConnectionStatus structure is used. This structure has the following fields:

Connection	This field, of type Unsigned, identifies the WebSocket connection status entry in the WebSocket_Connections property. A value of zero shall indicate that no WebSocket connection status entry exists.
VMAC	This field, of type OCTET STRING, is the VMAC address of the node at the other end of the WebSocket connection. If the VMAC is unknown, this field shall contain the WebSocket Peer VMAC address. See Clause YY.1.2.2.
Connection Status	This field, of type BACnetConnectionStatus, indicates the current status of the WebSocket connection. It reflects the WebSocket connection status indicated by the respective WebSocket_Connections property entry, if one exists. If no WebSocket_Connections entry exists, the 'Connection Status' shall be UNKNOWN.
Peer Node Type	This field, of type BACnetSCNodeType, indicates the type of the node at the other end of the WebSocket connection.
Active Hub	This field, of type BOOLEAN, shall be TRUE if the node at the other end of the WebSocket connection is the active hub, otherwise FALSE.
Error	This optional field, of type Error, shall indicate the most recent error for the WebSocket connection. If no error yet occurred, this field shall be absent.
Details	This optional field, of type CharacterString, may be used to indicate error details on the most recent error. This field shall be absent if 'Error' is absent.

YY.6.3.2 WebSocket Connection Keep-Alive

The WebSocket client node shall keep established WebSocket connections alive through periodic initiation of Advertisement-Solicitation messages to the node at the other end of the WebSocket connection, using the WebSocket Peer VMAC address as the destination address. Such messages shall be initiated only if there is no successful BVLC message exchange over the WebSocket connection within the keep-alive timeout. The keep-alive timeout is specified by the BACnet_SC_Keep_Alive property of the Network Port object. See Clause 12.56.Y6.

The WebSocket connections may be kept alive for as long as the WebSocket connection maximum lifetime allows. See Clause YY.7.5.5.

YY.6.3.3 WebSocket Connection to Hub

Regular nodes and the failover hub shall attempt to establish a WebSocket connection to the primary hub. The primary hub is not required to initiate any WebSocket connection.

The minimal timeout between connection attempts is configured in the BACnet_SC_Reconnect_Timeout property. See Clause 12.56.Y5.

The current WebSocket connections to the hubs are indicated in the Network Port object's BACnet_SC_Primary_Hub_Connection and BACnet_SC_Failover_Hub_Connection properties, referring to the respective WebSocket connection status entry in the BACnet_SC_Connections property, by the numeric connection identifier. See Clause 12.56.Y10.

The status of how the node is connected to the active hub shall be indicated in the Network Port object's BACnet_SC_Hub_Status property. See Clause 12.56.Y9. The enumeration BACnetSCHubStatus defines the following status values that may be indicated:

UNKNOWN	The status of the WebSocket connection to the hub is currently unknown.
DISABLED	The network port is disabled and no attempts are initiated to connect to the hub, nor are WebSocket connections accepted if a hub.
SELF_ACTIVE	The node is the active hub and no attempts to connect to a hub are initiated.
NONE	There is no WebSocket connection to the active hub, and there is currently no pending attempt to connect to the active hub.
CONNECTING	There is no WebSocket connection to the active hub and the node has a pending attempt to establish a WebSocket connection to the active hub.
PRIMARY_ACTIVE	The node is connected to the primary hub that is the active hub.
FAILOVER_ACTIVE	The node is connected to the failover hub that is the active hub.
BOTH_ACTIVE	The node is connected to both the primary and failover hub and both indicate being active. In this state, broadcast messages shall only be forwarded to one of the hubs. The selection of the hub is a local matter.
BOTH_INACTIVE	The node is connected to both the primary and failover hub and both indicate being inactive.

YY.6.3.3.1 Regular Node

A regular node shall attempt to establish a WebSocket connection to the primary hub.

If the WebSocket connection to the primary hub cannot be established, or the connection to the primary hub is closed or fails, or the primary hub indicates it is inactive, the BACnet/SC node shall attempt to connect to the failover hub.

If the failover hub becomes inactive or cannot be connected to, the BACnet/SC node shall attempt to connect to the primary hub.

If a WebSocket connection to the primary hub can be re-established, and the primary hub is active, a WebSocket connection to the failover hub is no longer required and may be closed by the regular node or the failover hub.

YY.6.3.3.2 Primary Hub

On startup, the primary hub shall start accepting WebSocket connections from other nodes and perform the hub switch function as the active hub.

If the primary hub is restored to accept WebSocket connections and to perform the hub switch function, the primary hub is the active hub.

YY.6.3.3.3 Failover Hub

On startup, the failover hub shall attempt to establish a WebSocket connection to the primary hub. The failover hub shall start as an inactive hub.

If the Websocket connection to the primary hub fails, or the primary hub indicates being inactive, the failover hub is the active hub and shall accept WebSocket connections and perform the hub function.

While the failover hub is the active hub, it shall continue attempts to connect to the primary hub.

If the WebSocket connection to the primary hub can be established and the primary hub is the active hub, the failover hub shall become the inactive hub and advertise this change to all connected nodes. When inactive, the failover hub may close WebSocket connections accepted as the active hub, after the advertisement of the new inactive state was sent.

YY.6.3.4 Initiating Direct WebSocket Connections

In addition to connecting to the active hub, a regular node may optionally initiate direct WebSocket connections to other nodes, as a WebSocket client. Direct WebSocket connections shall only be established to nodes of the same BACnet network.

The time of initiation and the termination of direct WebSocket connections is a local matter.

The WebSocket URI for directly connecting to a node may be configured in the BACnet_SC_Manual_Node_Bindings property of the Network Port object. See Clause 12.56.Y11. If a WebSocket URI is provided in this property for a node identified by the node's VMAC address, this WebSocket URI shall be used only to connect to the node.

If no WebSocket URI is configured for the node in the BACnet_SC_Manual_Node_Bindings property, the WebSocket URI for directly connecting to a node can be requested from the node by sending an Address-Resolution message to the node through the active hub.

If an Address-Resolution-ACK message is received, the WebSocket URI provided with this message can be used to initiate a direct WebSocket connection to the node.

If a BVLC-Result NAK message with an 'Error Class' of 'COMMUNICATION' and an 'ERROR CODE' of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED is received for the Address-Resolution message, the node is not accepting direct WebSocket connections.

If a WebSocket URI is provided with the Address-Resolution-ACK message, this WebSocket URI is not required to be valid for the network location and context of the requesting node. In this case, a direct WebSocket connection cannot be established.

The status of initiated direct WebSocket connections shall be represented in the BACnet_SC_Connection_Status property of the Network Port object.

YY.6.3.5 Accepting WebSocket Connections From Other Nodes

A regular node may optionally accept direct WebSocket connections from other nodes as a WebSocket server. Hub nodes are required to support accepting WebSocket connections and shall accept WebSocket connections when active. A node may limit the number of simultaneous Websocket connections accepted.

The WebSocket URI for initiating a direct WebSocket connection to the node shall be provided to other nodes with an Address-Resolution-ACK message, in response to an Address-Resolution message. If the node does not support accepting direct WebSocket connections, a BVLC-Result NAK shall be returned with an 'Error Code' of COMMUNICATION and an 'Error Class' of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

If the node does not know the WebSocket URI or knows the WebSocket URI is not suitable for the requestor, a BVLC-Result NAK shall be returned with an 'Error Code' of COMMUNICATION and an 'Error Class' of DATA_NOT_AVAILABLE.

The maintenance of WebSocket connections is in the responsibility of the initiating node. WebSocket connections can be closed by the accepting node at any time.

The status of accepted WebSocket connections shall be represented in the BACnet_SC_Connections property of the Network Port object.

YY.6.3.6 WebSocket Connection Termination

Nodes and hubs may terminate a WebSocket connection at any time, regardless of being a WebSocket client or server. The normal WebSocket connection procedures shall re-establish the minimally required WebSocket connection to the active hub.

YY.6.4 BVLC Message Exchange

YY.6.4.1 General Requirements For All Nodes

On establishment of a WebSocket connection, initiated by the node or accepted from a node, an Advertisement message shall be sent to the node at the other end of the WebSocket connection. The 'Destination Virtual Address' of this message shall be the WebSocket Peer VMAC address, and the message shall be sent through the established WebSocket connection.

On change of any information normally conveyed by the Advertisement message, the node shall initiate Advertisement messages to all nodes at the other ends of all WebSocket connections currently established, using the WebSocket Peer VMAC address as the destination address.

Nodes may initiate Advertisement or Advertisement-Solicitation messages to other nodes at anytime, e.g., for synchronization of status information in or about another node.

On receipt of an Advertisement message from a WebSocket connection, the node shall update its type and status information on the sender node as provided by the Advertisement message. If the destination address in the message is the WebSocket Peer VMAC address, the 'Source Virtual Address' field provides the VMAC of the node at the other end of the WebSocket connection.

On receipt of an Advertisement-Solicitation message from a WebSocket connection, an Advertisement message shall be returned to the sender of the Advertisement-Solicitation message using the sender's VMAC address as the destination address.

On receipt of an Address-Resolution message from a WebSocket connection, an Address-Resolution-ACK message shall only be returned if the node accepts WebSocket connections as a WebSocket server and a WebSocket URI for initiating a WebSocket connection can be provided. If no WebSocket connections are accepted, a BVLC-Result NAK for the Address-Resolution message shall be returned, indicating an 'Error Class' of COMMUNICATION and an 'Error Code' of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

YY.6.4.2 Regular Nodes and Inactive Hubs

The following message exchange procedures are applicable for regular nodes and inactive hubs. For the message exchange procedures of the active hub, see Clause YY.6.4.3.

The basic switch function supports the BVLC message exchange of regular nodes and inactive hubs. See Clause YY.5.1.

On receipt of a Unicast-NPDU or Forwarded-Broadcast-NPDU message from any of the WebSocket connections in place, and the message is addressed to the local node or was a broadcast, the NPDU shall be extracted and forwarded to the local network entity. If an Authentication-Data header is present, the authentication data shall also be forwarded.

On receipt of a Unicast-NPDU message from any of the WebSocket connections in place, and the message is addressed to a node other than the local node, the Unicast-NPDU shall be discarded.

On receipt of an Original-Broadcast-NPDU message from any of the WebSocket connections in place, a BVLC-Result NAK shall be returned to the source node indicating an 'Error Class' of COMMUNICATION, and an 'Error Code' of NOT_THE_ACTIVE_HUB.

On receipt of a unicast NPDU from the local network entity (i.e., a destination VMAC address is provided), the BVLL shall create a Unicast-NPDU BVLC message and send the message to the destination node. See Clause YY.5.1. The 'Destination Virtual Address' shall be the destination VMAC address provided. If authentication data is provided by the local network entity, an Authentication-Data header shall be inserted in the Unicast-NPDU message, conveying the authentication data provided.

On receipt of a broadcast NPDU from the local network entity, i.e., an empty destination MAC address is provided; the BVLL shall create an Original-Broadcast-NPDU BVLC message and send the message to the active hub, if one is reachable. If the active hub is not reachable, the message may be discarded and not sent. If authentication data is provided by the local network entity, an Authentication-Data header shall be inserted in the Original Broadcast-NPDU message, conveying the authentication data provided.

YY.6.4.3 Active Hub

The following message exchange procedures are applicable for a hub when active.

The hub switch function supports the BVLC message exchange of the active hub. See Clause YY.5.2.

On a change of the hub to active status, a BACnet/SC hub shall initiate Advertisement messages to the nodes at the other ends of all current WebSocket connections indicating the new status.

On receipt of a Unicast-NPDU or Forwarded-Broadcast-NPDU message from any of the WebSocket connections in place, and the message is addressed to the local node or was a broadcast, the NPDU shall be extracted and provided to the local network entity.

On receipt of a Unicast-NPDU message from any of the WebSocket connections in place, and the message is addressed to another node than the local node, the Unicast-NPDU shall be forwarded through the WebSocket connection to the destination node, if one exists. If no such connection exists, the message shall be discarded.

On receipt of an Original-Broadcast-NPDU message from any of the WebSocket connections in place, the message shall be converted into a Forwarded-Broadcast-NPDU message, and a copy shall be sent to all current WebSocket connections. In addition, the NPDU shall be provided to the local network entity.

On receipt of a unicast NPDU from the local network entity, i.e., a destination VMAC address is provided; the BVLL shall create a Unicast-NPDU BVLC message and send the message to the destination node, if a WebSocket connection exists. If no WebSocket connection exists, the message shall be discarded. The 'Destination Virtual Address' shall be the destination MAC address provided.

On receipt of a broadcast NPDU from the local network entity, i.e., an empty destination MAC address is provided by the local network entity; the BVLL shall create an Original-Broadcast-NPDU BVLC message and provide it to the local hub switch function. The message shall be converted into a Forwarded-Broadcast-NPDU message, and a copy shall be sent to all current WebSocket connections.

YY.7 Application of WebSockets in BACnet/SC

All WebSocket connections established in a BACnet/SC network shall apply the WebSocket protocol as specified in the following subclauses. WebSocket connections are used in BACnet/SC to bi-directionally exchange binary encoded BACnet/SC BVLC messages. A WebSocket network port may initiate and/or accept one or more WebSocket connections.

YY.7.1 The WebSocket Protocol

The WebSocket binary data packet payload is used for transporting the BACnet/SC BVLC messages between nodes of the BACnet/SC network.

With respect to a specific BACnet/SC WebSocket, the node that initiated the WebSocket is referred to as the WebSocket client, and the node that accepted the WebSocket connection is referred to as the WebSocket server.

The use of the WebSocket protocol for BACnet/SC is indicated by the WebSocket sub-protocol "bsc.bacnet.org". This sub-protocol identifier is registered with IANA as defined in RFC 6455.

The use of WebSocket Ping and Pong message exchange is optional.

YY.7.2 WebSocket URIs

The WebSocket URIs used by BACnet/SC devices are of "wss" URI scheme as defined in RFC 6455.

YY.7.3 WebSocket Binary Data Payload Format

For BACnet/SC, fully encoded BVLC messages are sent as binary data payload over the WebSocket connection.

YY.7.4 Network Security

The information security specified for BACnet/SC provides for confidentiality, integrity, and authenticity of BVLC messages transmitted across a WebSocket connection.

The establishment of a secure WebSocket shall be performed as defined in RFC 6455. For establishing a secure WebSocket when not in factory defaults condition, mutual authentication shall be performed when establishing the TLS connection. In addition to the WebSocket client performing a validation of the server's operational certificate, the WebSocket server shall request the WebSocket client's operational certificate and validate it to be signed by one of the configured trusted CAs.

In BACnet/SC, it is assumed that both the WebSocket client and the WebSocket server of a WebSocket connection are trusted, including all code they execute. Cross-origin validations are not required by this standard. Requirements on such validations are an installation local matter.

Implementations shall support TLS version 1.2 (RFC 5246) and the cipher suites listed in the following table at a minimum:

Table YY-4. Required Cipher Suites

TLS Cipher Suite	RFC
TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8	as defined in RFC 7251
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	as defined in RFC 5289

Support of newer versions of TLS or additional cipher suites is a local matter. Additional supported TLS versions and cipher suites shall be listed in the PICS. See Annex A.

YY.7.4.1 Certificate Management

Secure WebSocket connections require the use of TLS. The creation of TLS certificates and the management of the certificate signing authority, or authorities, are site-specific deployment options beyond the scope of this standard. However, to ensure interoperability, a WebSocket network port shall support the storage and use of certificates as defined in the following clauses.

YY.7.4.1.1 Manufacturer Credentials

For enabling secure BACnet/SC communication with a BACnet/SC device even for its initial configuration when in factory defaults condition, the device shall be equipped with a manufacturer defined certificate and a related private key. For the factory defaults condition, see Clause YY.7.4.2.

For interoperable and secured configuration of operational credentials, the manufacturer credentials shall enable establishing secure WebSocket connections as specified in Clause YY.7.5, with the exception that a WebSocket server endpoint shall accept any WebSocket client endpoint irrespective of the certificate used by the client.

The method of authentication of a WebSocket server endpoint by the WebSocket client endpoint and any configuration required for this are a local matter of the client.

YY.7.4.1.2 Operational Credentials

Before deployment to an active network, the WebSocket network port shall be configured with one or more CA certificates and a unique operational certificate with matching private key. The operational certificate shall be issued and signed by a CA that is verifiable with one of the CA certificates. This allows peer-to-peer mutual authentication so that a WebSocket server endpoint can verify that a client's certificate presented to it was issued by one of the CAs for which there is a CA certificate configured.

YY.7.4.1.3 Signing CA

The choice of a CA to generate the operational certificates shall be dictated by site policy. If a site-local CA is used, it is the responsibility of that CA to properly safeguard the private key used to issue operational certificates.

YY.7.4.1.4 Configuring Certificates and Activating TLS

The WebSocket network port requires the CA certificate(s), the operational certificate, and the private key in order to activate operational TLS communications on the site. The operational credentials are configured in the CA_Certificates, Operational_Certificate, and Private_Key properties of the Network Port object. See Clause 12.56.

A device may support an internal security function that allows it to generate its private and public key material by itself, e.g.., by a hardware security module. When operating in this mode, the device is not allowed to expose the private key, nor is it allowed to accept a private key from another device. In order to create a signed operational certificate, the Network Port object exposes a certificate signing request that includes parameters and data configured for the site and whose public key is related to the private key held internally. The certificate signing request allows a tool or other agent to forward the request to a CA and then write the operational certificate signed by the CA to the Operational_Certificate property.

The parameters to be included in the certificate signing request are configured in the Network Port object's Certificate_Signing_Parameters property. If this property is changed, and the changes are activated, then a new certificate signing request shall be generated that contains the parameters as configured and made available at the Certificate_Signing_Request property.

After the properties have been successfully written, a ReinitializeDevice ACTIVATE_CHANGES can be requested to cause the pending values to become the active set. The currently active set is reflected in the properties CA_Certificates and Operational_Certificate. When the changes to the Network Port object are activated, the WebSocket network port shall validate the pending certificates and private key. If any of the pending information is empty or malformed, or if the pending Operational_Certificate cannot be verified by any of the pending CA certificates in CA_Certificates, or if the pending private key in Private_Key does not match the pending operational certificate, then the Network Port object shall take on a Reliability of CONFIGURATION_ERROR and not activate TLS with the pending values. If the previous set of configuration values is still available and valid, then the WebSocket network port shall keep using the previous values and drop the pending values.

YY.7.4.2 Factory Defaults Condition

In the factory defaults condition, a BACnet/SC network port shall only possess the manufacturer credentials, no operational credentials shall be configured, and the device shall not contain any sensitive data.

In this condition, the manufacturer credentials shall be used for initiating and accepting secure WebSocket connections for BACnet/SC. See Clause YY.7.4.1.1.

YY.7.4.2.1 Reset to Factory Defaults

Devices shall provide a suitably secure out-of-band mechanism to place a BACnet/SC network port into "factory defaults" condition. It is recommended that this requires physical access to the device.

Performing a reset to "factory defaults" condition shall erase all operational certificates, all CA certificates, and all private keys from all Network Port objects that have the respective properties. Any sensitive data the device contains shall 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.

The manufacturer credentials shall not be erased when reset to "factory defaults" condition.

YY.7.5 WebSocket Operation

BACnet/SC uses WebSocket connections as defined in RFC6455 to bi-directionally transfer BACnet/SC BVLC messages as binary data payload.

The status of the WebSocket connections initiated or accepted is indicated by a state defined in the BACnetConnectionStatus enumeration. The status values that may be indicated are:

UNKNOWN	The status of the WebSocket connection is unknown.
DISABLED	The WebSocket connection is disabled and no attempts to reconnect are made and no WebSocket connection is accepted.
DISCONNECTED	The WebSocket connection is disconnected.
CONNECTING	An attempt to establish the WebSocket connection as a client or server is pending.
CONNECTED	The attempt to establish the WebSocket connection as a client succeeded.
CONNECTION_FAILED	The attempt to establish the WebSocket connection as a client failed due to any reason.
ACCEPTING	The node is accepting WebSocket connection attempts.
ACCEPTED	The WebSocket connection was accepted and successfully established.

Figure YY-6 depicts the WebSocket connection state machine for the WebSocket connection client end and the server end.

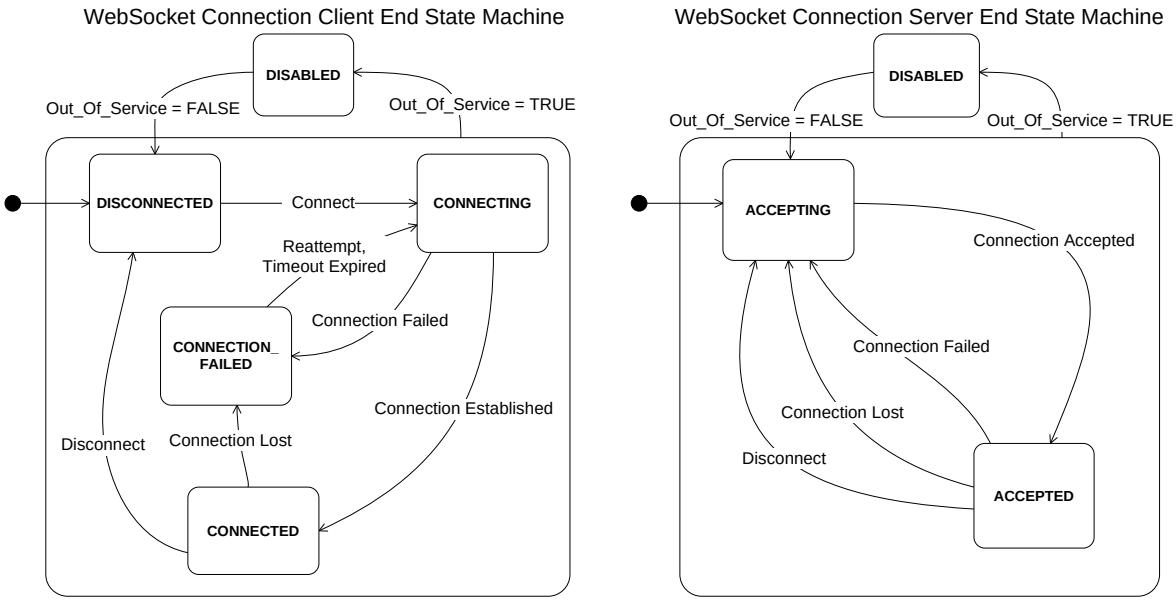


Figure YY-6. WebSocket Connection State Machines

For the representation of the WebSocket connections and their status in the Network Port object's WebSocket_Connections property, the structure BACnetWebSocketStatus is used and contains the following fields:

WebSocket Connection	This field, of type Unsigned; identifies the WebSocket connection. The value of this field is determined by the network port and shall be unique among all WebSocket connections maintained by the WebSocket network port. The value of zero shall not be used.
Connection Status	This field, of type BACnetConnectionStatus, indicates the current status of the WebSocket connection.
URI	This field, of type CharacterString, indicates the WebSocket URI of the server endpoint of this WebSocket connection if initiated. If the WebSocket URI is unknown or the WebSocket connection was accepted, an empty string shall be indicated.
Error	This optional field, of type Error, shall indicate the most recent error for the WebSocket connection. If no error was detected yet, this field shall be absent.
Details	This optional field, of type CharacterString, may be used to indicate error details on the most recent error. If the 'Error' field is absent, than this field shall also be absent.

YY.7.5.1 WebSocket Connection Control

Opening and closing a WebSocket connection shall take place as defined in RFC 6455.

For initiating a WebSocket connection, the WebSocket client shall indicate a sub-protocol of "bsc.bacnet.org" in the 'Sec-WebSocket-Protocol' HTTP header. The WebSocket connection shall only be established if the WebSocket server supports this sub-protocol, i.e., is a BACnet/SC hub or node.

YY.7.5.2 Initiating WebSocket Connections

For BACnet/SC, only secured WebSocket connections over TLS are initiated.

If the WebSocket URI provided indicates a URI scheme other than "wss", no WebSocket connection to that URI shall be initiated. If applicable, an 'Error Code' of WEBSOCKET_SCHEME_NOT_SUPPORTED shall be indicated.

If the DNS resolution of the host name in the WebSocket URI fails, the following error codes shall be indicated for the intended WebSocket connection.

Situation	Error Code
DNS is unknown or not reachable	DNS_UNAVAILABLE
The host name cannot be resolved to its IP or IPv6 address.	DNS_NAME_RESOLUTION_FAILED
There is an error in the local DNS resolver that prevents it from resolving the host name.	DNS_RESOLVER_FAILURE
Any other DNS error situation	DNS_OTHER_ERROR

If the host with the IP address resulting from DNS host name resolution is not reachable, and the respective IP error is available, then the following error codes shall be indicated for the intended WebSocket connection.

Situation	Error Code
IP address not reachable	IP_ADDRESS_NOT_REACHABLE
Any other IP error situation	IP_OTHER_ERROR

If the TCP connection to the IP address and port cannot be established, and the respective TCP error is available, then the following error codes shall be indicated for the intended WebSocket connection.

Situation	Error Code
The connection could not be established due to no response within timeout.	TCP_CONNECT_TIMEOUT
The connection is not accepted by the peer.	TCP_CONNECTION_REFUSED
Any other TCP error situation	TCP_OTHER_ERROR

If the TLS session on the TCP connection cannot be established, and the respective fatal TLS error is available, then the following error codes shall be indicated for the intended WebSocket connection.

Situation	Error Code
The security parameters of the client and server do not match.	TLS_SECURITY_PARAMETER_MISMATCH
The client certificate contains an error that prevents it from being authenticated.	TLS_CLIENT_CERTIFICATE_ERROR
The server certificate contains an error that prevents it from being authenticated.	TLS_SERVER_CERTIFICATE_ERROR
Authentication of the client failed.	TLS_CLIENT_AUTHENTICATION_FAILED
Authentication of the server failed.	TLS_SERVER_AUTHENTICATION_FAILED
Client certificate validity window does not include current time.	TLS_CLIENT_CERTIFICATE_EXPIRED
Server certificate validity window does not include current time.	TLS_SERVER_CERTIFICATE_EXPIRED
Client certificate revoked	TLS_CLIENT_CERTIFICATE_REVOKED
Server certificate is revoked	TLS_SERVER_CERTIFICATE_REVOKED
Any other TLS error situation	TLS_OTHER_ERROR

If the HTTP exchange for upgrade to the WebSocket protocol fails, and the respective HTTP error is available, then the following error codes shall be indicated for the intended WebSocket connection.

Situation	Error Code
Server reports unexpected response code.	HTTP_UNEXPECTED_RESPONSE_CODE
Server does not accept upgrade to the WebSocket protocol.	HTTP_NO_UPGRADE
Redirect to another location of the peer WebSocket port received.	HTTP_RESOURCE_NOT_LOCAL
Proxy Authentication failed	HTTP_PROXY_AUTHENTICATION_FAILED
No response from server within timeout.	HTTP_RESPONSE_TIMEOUT
Syntax error in HTTP response received.	HTTP_RESPONSE_SYNTAX_ERROR
Errors in values of the HTTP response received.	HTTP_RESPONSE_VALUE_ERROR
Missing header fields in response.	HTTP_RESPONSE_MISSING_HEADER
Response contains any other error in HTTP header fields.	HTTP_WEBSOCKET_HEADER_ERROR
No upgrade request was received by the server.	HTTP_UPGRADE_REQUIRED
Upgrading to WebSocket protocol failed.	HTTP_UPGRADE_ERROR
No more HTTP connections are available currently.	HTTP_TEMPORARY_UNAVAILABLE
No inbound requests supported. The host is not an HTTP server.	HTTP_NOT_A_SERVER
Any other error situation	HTTP_OTHER_ERROR

On successful establishment of the WebSocket connection, a connection status of CONNECTED shall be indicated for the intended WebSocket connection.

The status of all initiated WebSocket connections shall be indicated in the WebSocket_Connections property, if present, of the corresponding Network Port object.

YY.7.5.3 Accepting WebSocket Connections

A WebSocket server implements an HTTP server and supports HTTP upgrades to the WebSocket protocol. This WebSocket server shall accept WebSocket connections for the "bsc.bacnet.org" sub-protocol, as defined by this annex.

For BACnet/SC, only secured WebSocket connections over TLS shall be accepted.

The default TCP ports accepting "wss" scheme secured WebSockets over TLS shall be indicated in the WebSocket_Secure_Server_Port property.

The status of all accepted WebSocket connections may be indicated in the WebSocket_Connections property of the Network Port object.

YY.7.5.4 Message Exchange

All BVLC messages are binary encoded as defined in Clause YY.2 and shall be transmitted as binary data frames. BVLC messages can be sent through a WebSocket connection in both directions.

If the length of a BVLC message received through a WebSocket connection exceeds the maximum BVLC length supported by the receiver, the BVLC message shall be discarded and not be processed.

YY.7.5.5 Refreshing WebSocket Connections

WebSocket connections may be required to be refreshed such as when new key material must be generated periodically for TLS. The Network Port object's WebSocket_Maximum_Lifetime property allows configuring the maximum time that any of the WebSocket connections may exist before it shall be re-established.

The WebSocket server shall monitor the lifetime of each accepted WebSocket connection and shall terminate an accepted WebSocket connection when the lifetime exceeds the maximum lifetime.

The WebSocket client shall monitor the lifetime of each initiated WebSocket connection and shall terminate an initiated WebSocket connection when the lifetime exceeds the maximum lifetime. The WebSocket client may re-establish the WebSocket connection afterwards.

YY.7.5.6 Closing WebSocket Connections

WebSocket connections may be closed by either end at any time. See RFC 6455.

The WebSocket close handshake shall be performed when intentionally closing a WebSocket connection. When a WebSocket connection is closed, the resulting close status shall be indicated for the associated WebSocket connection. The close status code received from the WebSocket connection shall map to error codes as follows. For the meaning of WebSocket response codes, see RFC 6455 Section 7.4.1.

WebSocket Close Status Code	Error Code
1000	WEBSOCKET_CLOSED_BY_PEER
1001	WEBSOCKET_ENDPOINT_LEAVES
1002	WEBSOCKET_PROTOCOL_ERROR
1003	WEBSOCKET_DATA_NOT_ACCEPTED
1006	WEBSOCKET_CLOSED_ABNORMALLY
1007	WEBSOCKET_DATA_INCONSISTENT
1008	WEBSOCKET_DATA_AGAINST_POLICY
1009	WEBSOCKET_FRAME_TOO_LONG
1010	WEBSOCKET_EXTENSION_MISSING
1011	WEBSOCKET_REQUEST_UNAVAILABLE
all other codes	WEBSOCKET_OTHER_ERROR

YY.7.5.6.1 Fatal Error Situations in Established WebSocket Connections

When a fatal TLS error occurs in an established TLS session, an error code of TLS_FATAL_ERROR shall be indicated for the associated WebSocket connection.

When the TCP connection fails unexpectedly, e.g., due to a network interruption, and only a TCP error is available, an error code of TCP_FATAL_ERROR shall be indicated for the associated connection.

[Add new Clause **H.7.X**, p. 1020]

H.7.X VMAC Addresses Representing Broadcast and Multicast Addresses

The 3-octet virtual MAC address for local broadcast to all nodes of the BACnet network shall be X'800000'.

135-2016*bj*-4. Extend the Network Port Object for BACnet/SC

Rationale

BACnet/SC requires some configuration items, status indications, and other elements to be network visible and interoperably configurable.

The Network Port object is extended to allow representation and configuration of BACnet/SC.

Respective Clause 21 ASN.1 production modifications and new productions are collected in part 6 of this addendum.

This section also includes an editorial reformatting of the tables indicating presence requirements for properties, to improve readability.

[Network Port Object Property Table Changes]

[Change **Clause 12.56**, p. 517]

12.56 Network Port Object Type

The Network Port object provides access to the configuration and properties of network ports of a device. All BACnet devices shall contain at least one Network Port object per physical port which the device can be configured to communicate BACnet through (unless the port is currently for communications on a network other than the current BACnet internetwork and this use precludes its use for the current BACnet internetwork). It is a local matter whether or not Network Port objects exist for non-configured ports. It is a local matter whether or not the Network Port object is used for non-BACnet ports.

Verification and validation of property values within a Network Port object is a local matter.

Property values which are required to maintain proper operation of the network shall be retained across a device reset.

The Network Port object type can be implemented as a single interface through which all of the settings for a network port are accessed, or the Network Port objects can be organized in a hierarchy which separates the settings for each communication protocol level. See Clause 12.56.10 for more details on hierarchical Network Port objects.

Network Port objects may optionally support intrinsic reporting to facilitate the reporting of fault conditions. Network Port objects that support intrinsic reporting shall apply the NONE event algorithm.

~~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~~It is a local matter whether a Network Port object contains *additional* properties that do not *specifically* 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. ~~This is permitted, but not required.~~

Table 12-71 contains the required and optional base properties included in all Network Port objects independent of the Network_Type or Protocol_Level. Tables 12-71a to 12-71m indicate the additional required and optional properties included in the Network Port object, based on the Network_Type and Protocol_Level of the object. The conformance codes in these tables apply for the Protocol_Level indicated on top of the respective column.

No additional properties are required for Network Port objects with a Protocol_Level of NON_BACNET_APPLICATION, and the support of additional standard Network Port object properties is a local matter.

Table 12-71. *Base* Properties of the Network Port Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Status_Flags	BACnetStatusFlags	R
Reliability	BACnetReliability	R
Out_Of_Service	BOOLEAN	R
Network_Type	BACnetNetworkType	R
Protocol_Level	BACnetProtocolLevel	R
Reference_Port	Unsigned	O
~~Network_Number~~	~~Unsigned16~~	~~O~~[1]
~~Network_Number_Quality~~	~~BACnetNetworkNumberQuality~~	~~O~~
Changes_Pending	BOOLEAN	R
Command	BACnetNetworkPortCommand	O[21]
~~MAC_Address~~	~~OCTET STRING~~	~~O~~[3]
~~APDU_Length~~	~~Unsigned~~	~~O~~[1b]
~~Link_Speed~~	~~REAL~~	~~R~~
~~Link_Speeds~~	~~BACnetARRAY[N] of REAL~~	~~O~~[4]
~~Link_Speed_Autonegotiate~~	~~BOOLEAN~~	~~O~~
Network_Interface_Name	CharacterString	O
~~BACnet_IP_Mode~~	~~BACnetIPMode~~	~~O~~[5]
~~IP_Address~~	~~OCTET STRING~~	~~O~~[6]
~~BACnet_IP_UDP_Port~~	~~Unsigned16~~	~~O~~[5]
~~IP_Subnet_Mask~~	~~OCTET STRING~~	~~O~~[6]
~~IP_Default_Gateway~~	~~OCTET STRING~~	~~O~~[6]
~~BACnet_IP_Multicast_Address~~	~~OCTET STRING~~	~~O~~[7]
~~IP_DNS_Server~~	~~BACnetARRAY[N] of OCTET STRING~~	~~O~~[6]
~~IP_DHCP_Enable~~	~~BOOLEAN~~	~~O~~[8]
~~IP_DHCP_Lease_Time~~	~~Unsigned~~	~~O~~
~~IP_DHCP_Lease_Time_Remaining~~	~~Unsigned~~	~~O~~
~~IP_DHCP_Server~~	~~OCTET STRING~~	~~O~~
~~BACnet_IP_NAT_Traversal~~	~~BOOLEAN~~	~~O~~[9]
~~BACnet_IP_Global_Address~~	~~BACnetHostNPort~~	~~O~~[10]
~~BBMD_Broadcast_Distribution_Table~~	~~BACnetLIST of BACnetBDTEntry~~	~~O~~[11]
~~BBMD_Accept_FD_Registrations~~	~~BOOLEAN~~	~~O~~[11]
~~BBMD_Foreign_Device_Table~~	~~BACnetLIST of BACnetFDTEntry~~	~~O~~[12]
~~FD_BBMD_Address~~	~~BACnetHostNPort~~	~~O~~[13]
~~FD_Subscription_Lifetime~~	~~Unsigned16~~	~~O~~[13]
~~BACnet_IPv6_Mode~~	~~BACnetIPMode~~	~~O~~[14]
~~IPv6_Address~~	~~OCTET STRING~~	~~O~~[15]
~~IPv6_Prefix_Length~~	~~Unsigned8~~	~~O~~[15]
~~BACnet_IPv6_UDP_Port~~	~~Unsigned16~~	~~O~~[14]
~~IPv6_Default_Gateway~~	~~OCTET STRING~~	~~O~~[15]
~~BACnet_IPv6_Multicast_Address~~	~~OCTET STRING~~	~~O~~[14]
~~IPv6_DNS_Server~~	~~BACnetARRAY[N] of OCTET STRING~~	~~O~~[15]
~~IPv6_Auto_Addressing_Enable~~	~~BOOLEAN~~	~~O~~[16]
~~IPv6_DHCP_Lease_Time~~	~~Unsigned~~	~~O~~
~~IPv6_DHCP_Lease_Time_Remaining~~	~~Unsigned~~	~~O~~

~~IPv6_DHCP_Server~~	~~OCTET STRING~~	~~O~~
~~IPv6_Zone_Index~~	~~CharacterString~~	~~O[17]~~
~~Max_Master~~	~~Unsigned8~~	~~O[18]~~
~~Max_Info_Frames~~	~~Unsigned8~~	~~O[18]~~
~~Slave_Proxy_Enable~~	~~BOOLEAN~~	~~O[19]~~
~~Manual_Slave_Address_Binding~~	~~BACnetLIST of BACnetAddressBinding~~	~~O[19]~~
~~Auto_Slave_Discovery~~	~~BOOLEAN~~	~~O[20]~~
~~Slave_Address_Binding~~	~~BACnetLIST of BACnetAddressBinding~~	~~O[21]~~
~~Virtual_MAC_Address_Table~~	~~BACnetLIST of BACnetVMACEntry~~	~~O[22]~~
~~Routing_Table~~	~~BACnetLIST of BACnetRouterEntry~~	~~O~~
Event_Detection_Enable	BOOLEAN	O[~~23,24~~2, 3]
Notification_Class	Unsigned	O[~~23,24~~2, 3]
Event_Enable	BACnetEventTransitionBits	O[~~23,24~~2, 3]
Acked_Transitions	BACnetEventTransitionBits	O[~~23,24~~2, 3]
Notify_Type	BACnetNotifyType	O[~~23,24~~2, 3]
Event_Time_Stamps	BACnetARRAY[3] of BACnetTimeStamp	O[~~23,24~~2, 3]
Event_Message_Texts	BACnetARRAY[3] of CharacterString	O[~~24~~3]
Event_Message_Texts_Config	BACnetARRAY[3] of CharacterString	O[~~24~~3]
Event_State	BACnetEventState	O[~~23~~2]
Reliability_Evaluation_Inhibit	BOOLEAN	O
Property_List	BACnetARRAY[N] of BACnetPropertyIdentifier	R
Tags	BACnetARRAY[N] of BACnetNameValue	O
Profile_Location	CharacterString	O
Profile_Name	CharacterString	O

[1] ~~Required to be writable in routers, *Clause 24* secure devices, and any other device that requires knowledge of the network number for proper operation.~~

[12] Shall be present if, and only if, the object supports execution of any of the values of the Command property. If present, this property shall be writable.

[3] ~~Required if the port is not a PTP link. Read-only if the port is a BACnet/IP port or if the network represented by this object requires VMAC addressing.~~

[4] ~~Required if Link_Speed is writable.~~

[5] ~~Required to be present if the port is a BACnet/IP port.~~

[6] ~~Required if the port is a BACnet/IP port. If the BACnet_IP_DHCP property is TRUE, and this property is configured by DHCP, this property shall be read-only.~~

[7] ~~Required to be present if Network_Type is IPV4, Protocol_Level is BACNET_APPLICATION and the port supports multicast.~~

[8] ~~Shall be present if, and only if, Network_Type is IPV4 and the port can be configured by DHCP.~~

[9] ~~Required to be present if the Network_Type is IPV4, Protocol_Level is BACNET_APPLICATION, and the is capable of communicating through a NAT router as described in Clause J.7.8.~~

[10] ~~Required if Network_Type is IPV4, Protocol_Level is BACNET_APPLICATION, and the device is configured to communicate through a NAT router as described in Clause J.7.8~~

[11] ~~Required to be present if Network_Type is IPV4 or IPV6 and the device is capable of functioning as a BBMD.~~

[12] ~~Required if Network_Type is IPV4 or IPV6 and the device is capable of functioning as a BBMD.~~

[13] ~~Required to be present if Network_Type is IPV4 or IPV6 and BACnet_IP_Mode or BACnet_IPv6_Mode respectively is set to FOREIGN.~~

[14] ~~Required to be present if Network_Type is IPV6 and Protocol_Level is BACNET_APPLICATION.~~

[15] ~~Required to be present if Network_Type is IPV6 . Read-only if the value is configured by automatic address assignment.~~

[16] ~~Required to be present if Network_Type is IPV6 and the port supports automatic IPv6 address assignment.~~

[17] ~~Required to be present if Network_Type is IPV6 and the node supports multiple IPv6 link local addresses.~~

[18] ~~Required to be present and writable if Network_Type is MSTP, the device is an MS/TP master node, and the device supports the WriteProperty service.~~

[19] ~~Required to be present and writable if Network_Type is MSTP, and the device is capable of being a Slave-Proxy device.~~

[20] ~~Required if Network_Type is MSTP, Protocol_Level is BACNET_APPLICATION, and the device is capable~~

~~of being a Slave-Proxy device that implements automatic discovery of slaves.~~

~~[21] Required if Network_Type is MSTP, Protocol_Level is BACNET_APPLICATION, and the device is capable of being a Slave-Proxy device.~~

~~[22] Required if the network represented by this object requires VMAC addressing.~~

[223] These properties are required if the object supports intrinsic reporting.

[324] These properties shall be present only if the object supports intrinsic reporting.

Table 12-71a. *Additional Properties of the Network Port Object Type if Network Type is ETHERNET*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*	*R*[2]		
APDU_Length	*Unsigned*			*R*
Link_Speed	*REAL*	*R*		
Link_Speeds	*BACnetARRAY[N] of REAL*	*O*[3]		
Link_Speed_Autonegotiate	*BOOLEAN*	*O*		
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers and any port that requires knowledge of the BACnet network number for proper operation.*

[2] *Required to be read-only.*

[3] *Required if Link_Speed is writable.*

Table 12-71b. *Additional Properties of the Network Port Object Type if Network Type is ARCNET*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*	*R*[2]		
APDU_Length	*Unsigned*			*R*
Link_Speed	*REAL*	*R*		
Link_Speeds	*BACnetARRAY[N] of REAL*	*O*[3]		
Link_Speed_Autonegotiate	*BOOLEAN*	*O*		
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers and any port that requires knowledge of the BACnet network number for proper operation.*

[2] *Required to be configurable.*

[3] *Required if Link_Speed is writable.*

Table 12-71c. *Additional Properties of the Network Port Object Type if Network Type is MSTP*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*		*R*[2]	
APDU_Length	*Unsigned*			*R*
Max_Master	*Unsigned8*		*O*[3]	
Max_Info_Frames	*Unsigned8*		*O*[3]	
Slave_Proxy_Enable	*BOOLEAN*			*O*[4]
Manual_Slave_Address_Binding	*BACnetLIST of BACnetAddressBinding*			*O*[4]
Auto_Slave_Discovery	*BOOLEAN*			*O*[5]
Slave_Address_Binding	*BACnetLIST of BACnetAddressBinding*			*O*[6]
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers, Clause 24 secure devices, and any port that requires knowledge of the BACnet network number for proper operation.*
[2] *Required to be configurable.*
[3] *Required to be present if the port is an MSTP master. Required to be writable if the port is an MSTP master and the device supports the WriteProperty service.*
[4] *Required to be present and writable if the port is capable of being a Slave-Proxy.*
[5] *Required if the port is capable of being a Slave-Proxy that implements automatic discovery of slaves.*
[6] *Required if the port is capable of being a Slave-Proxy.*

Table 12-71d. *Additional Properties of the Network Port Object Type if Network Type is PTP*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
APDU_Length	*Unsigned*			*R*
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

Table 12-71e. *Additional Properties of the Network Port Object Type when Network Type is LONTALK*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*	*R*[2]		
APDU_Length	*Unsigned*			*R*
Link_Speed	*REAL*	*R*		
Link_Speeds	*BACnetARRAY[N] of REAL*	*O*[3]		
Link_Speed_Autonegotiate	*BOOLEAN*	*O*		

Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers, Clause 24 secure devices, and any port that requires knowledge of the BACnet network number for proper operation.*
[2] *Required to be configurable.*
[3] *Required if Link_Speed is writable.*

Table 12-71f. *Additional Properties of the Network Port Object Type if Network Type is IPV4*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*			*R*[2]
APDU_Length	*Unsigned*			*R*
BACnet_IP_Mode	*BACnetIPMode*			*R*
IP_Address	*OCTET STRING*		*R*[3]	
BACnet_IP_UDP_Port	*Unsigned16*			*R*
IP_Subnet_Mask	*OCTET STRING*		*R*[3]	
IP_Default_Gateway	*OCTET STRING*		*R*[3]	
BACnet_IP_Multicast_Address	*OCTET STRING*			*O*[4]
IP_DNS_Server	*BACnetARRAY[N] of OCTET STRING*		*R*[3]	
IP_DHCP_Enable	*BOOLEAN*		*O*[5]	
IP_DHCP_Lease_Time	*Unsigned*		*O*	
IP_DHCP_Lease_Time_Remaining	*Unsigned*		*O*	
IP_DHCP_Server	*OCTET STRING*		*O*	
BACnet_IP_NAT_Traversal	*BOOLEAN*			*O*[6]
BACnet_IP_Global_Address	*BACnetHostNPort*			*O*[7]
BBMD_Broadcast_Distribution_Table	*BACnetLIST of BACnetBDTEntry*			*O*[8]
BBMD_Accept_FD_Registrations	*BOOLEAN*			*O*[8]
BBMD_Foreign_Device_Table	*BACnetLIST of BACnetFDTEntry*			*O*[9]
FD_BBMD_Address	*BACnetHostNPort*			*O*[10]
FD_Subscription_Lifetime	*Unsigned16*			*O*[10]
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers, Clause 24 secure devices, and any port that requires knowledge of the BACnet network number for proper operation.*
[2] *Required to be read-only and shall contain the six octet B/IP address.*
[3] *If the BACnet_IP_DHCP property is TRUE, and this property is configured by DHCP, this property shall be read-only.*
[4] *Required to be present if the port supports multicast.*
[5] *Shall be present if, and only if, the port can be configured by DHCP.*
[6] *Required to be present if the port is capable of communicating through a NAT router as described in Clause J.7.8.*
[7] *Required if the port is configured to communicate through a NAT router as described in Clause J.7.8*
[8] *Required to be present if the port is capable of functioning as a BBMD.*
[9] *Required if the port is capable of functioning as a BBMD.*
[10] *Required to be present if BACnet_IP_Mode is set to FOREIGN.*

Table 12-71g. *Additional Properties of the Network Port Object Type if Network Type is ZIGBEE*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*	*R*[2]		
APDU_Length	*Unsigned*			*R*
Link_Speed	*REAL*	*R*		
Link_Speeds	*BACnetARRAY[N] of REAL*	*O*[3]		
Link_Speed_Autonegotiate	*BOOLEAN*	*O*		
Virtual_MAC_Address_Table	*BACnetLIST of BACnetVMACEntry*			*R*
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers, Clause 24 secure devices, and any port that requires knowledge of the BACnet network number for proper operation.*
[2] *Required to be read-only.*
[3] *Required if Link_Speed is writable.*

Table 12-71h. *Additional Properties of the Network Port Object Type if Network Type is VIRTUAL*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*	*O*		
APDU_Length	*Unsigned*			*R*
Link_Speed	*REAL*	*R*		
Link_Speeds	*BACnetARRAY[N] of REAL*	*O*[2]		
Link_Speed_Autonegotiate	*BOOLEAN*	*O*		
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers, Clause 24 secure devices, and any port that requires knowledge of the BACnet network number for proper operation.*
[2] *Required if Link_Speed is writable.*

Table 12-71i. *Additional Properties of the Network Port Object Type if Network Type is IPV6*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*			*R*[2]
APDU_Length	*Unsigned*			*R*

BBMD_Broadcast_Distribution_Table	*BACnetLIST of BACnetBDTEntry*			*O*[3]
BBMD_Accept_FD_Registrations	*BOOLEAN*			*O*[3]
BBMD_Foreign_Device_Table	*BACnetLIST of BACnetFDTEntry*			*O*[4]
FD_BBMD_Address	*BACnetHostNPort*			*O*[5]
FD_Subscription_Lifetime	*Unsigned16*			*O*[5]
BACnet_IPv6_Mode	*BACnetIPMode*			*R*
IPv6_Address	*OCTET STRING*		*R*[6]	
IPv6_Prefix_Length	*Unsigned8*		*R*[6]	
BACnet_IPv6_UDP_Port	*Unsigned16*			*R*
IPv6_Default_Gateway	*OCTET STRING*		*R*[6]	
BACnet_IPv6_Multicast_Address	*OCTET STRING*			*R*
IPv6_DNS_Server	*BACnetARRAY[N] of OCTET STRING*		*R*[6]	
IPv6_Auto_Addressing_Enable	*BOOLEAN*		*O*[7]	
IPv6_DHCP_Lease_Time	*Unsigned*		*O*	
IPv6_DHCP_Lease_Time_Remaining	*Unsigned*		*O*	
IPv6_DHCP_Server	*OCTET STRING*		*O*	
IPv6_Zone_Index	*CharacterString*		*O*[8]	
Virtual_MAC_Address_Table	*BACnetLIST of BACnetVMACEntry*			*R*
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers and any port that requires knowledge of the BACnet network number for proper operation.*
[2] *Required to be read-only and shall contain the VMAC address.*
[3] *Required to be present if the port is capable of functioning as a BBMD.*
[4] *Required if the port is capable of functioning as a BBMD.*
[5] *Required to be present if BACnet_IPv6_Mode is set to FOREIGN.*
[6] *Required to be read-only if the value is configured by automatic address assignment.*
[7] *Required to be present if the port supports automatic IPv6 address assignment.*
[8] *Required to be present if the port supports multiple IPv6 link local addresses.*

[Note to reviewer: the network types SECURE_CONNECT and WEBSOCKET are introduced with this addendum]

Table 12-71j. *Additional Properties of the Network Port Object Type if Network Type is SECURE_CONNECT*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*R*[1]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*R*
MAC_Address	*OCTET STRING*			*R*[2]
APDU_Length	*Unsigned*			*R*
Max_BVLC_Length_Accepted	*Unsigned*			*R*
BACnet_SC_Node_Type	*BACnetSCNodeType*			*R*
BACnet_SC_Primary_Hub	*BACnetSCConnectionPeer*			*R*
BACnet_SC_Failover_Hub	*BACnetSCConnectionPeer*			*R*
BACnet_SC_Reconnect_Timeout	*Unsigned*			*R*
BACnet_SC_Keep_Alive	*Unsigned*			*R*
BACnet_SC_Primary_Hub_Connection	*Unsigned*			*O*
BACnet_SC_Failover_Hub_Connection	*Unsigned*			*O*
BACnet_SC_Hub_Status	*BACnetSCHubStatus*			*R*
BACnet_SC_Connections	*BACnetLIST of BACnetSCConnectionStatus*			*O*

BACnet_SC_Manual_Node_Bindings	*BACnetLIST of BACnetSCNodeBinding*			*O*
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*

[1] *Required to be writable in routers and any port that requires knowledge of the BACnet network number for proper operation.*
[2] *Required to be read-only and shall contain the VMAC address.*

Table 12-71k. *Additional Properties of the Network Port Object Type if Network Type is WEBSOCKET*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
WebSocket_Maximum_Lifetime	*Unsigned*		*R*	
WebSocket_Plain_Server_Port	*Unsigned16*		*O*[1]	
WebSocket_Secure_Server_Port	*Unsigned16*		*O*[2]	
WebSocket_Protocol	*CharacterString*		*O*	
WebSocket_Payload_Type	*BACnetWebSocketPayloadType*		*O*	
WebSocket_PingPong	*BOOLEAN*		*O*	
Credentials_Ref	*BACnetObjectIdentifier*		*O*	
Operational_Certificate	*OCTET STRING*		*O*[3,4]	
Private_Key	*OCTET STRING*		*O*[5,6]	
CA_Certificates	*BACnetARRAY[N] of OCTET STRING*		*O*[3,4]	
Certificate_Signing_Request	*OCTET STRING*		*O*[7]	
Certificate_Signing_Parameters	*BACnetCertificateSigningParameters*		*O*[4,7]	
WebSocket_Connections	*BACnetLIST of BACnetWebSocketStatus*		*O*	

[1] *Required to be present and writable if the port is accepting plain WebSocket connections.*
[2] *Required to be present and writable if the port is accepting secured WebSocket connections.*
[3] *Required to be present if the port is accepting or initiating secured WebSocket connections.*
[4] *This property, if present, is required to be writable if the Credentials_Ref property is absent.*
[5] *This property is required to be present if the port is accepting or initiating secured WebSocket connections and the Certificate_Signing_Request property is absent.*
[6] *This property, if present, is required to be writable if the Certificate_Signing_Request and the Credentials_Ref property are absent.*
[7] *If one of these properties is present, both properties shall be present.*

Table 12-71l. *Additional Properties of the Network Port Object Type if Network Type is SERIAL*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Link_Speed	*REAL*	*R*		
Link_Speeds	*BACnetARRAY[N] of REAL*	*O*[1]		
Link_Speed_Autonegotiate	*BOOLEAN*	*O*		

[1] *Required if Link_Speed is writable.*

Table 12-71m. *Additional Properties of the Network Port Object Type if Network_Type is proprietary*

Property Identifier	*Property Datatype*	*PHYSICAL*	*PROTOCOL*	*BACNET_APPLICATION*
Network_Number	*Unsigned16*			*O*[1,2]
Network_Number_Quality	*BACnetNetworkNumberQuality*			*O*[1]
MAC_Address	*OCTET STRING*			*O*[1]
APDU_Length	*Unsigned*			*R*
Routing_Table	*BACnetLIST of BACnetRouterEntry*			*O*
<other standard properties>	*<as defined for the property>*	*O*[3]	*O*[3]	*O*[3]

[1] *Required to be present if the object represents a BACnet network port with a MAC address. Not required for BACnet half-router links.*

[2] *If present, required to be writable in routers and any port that requires knowledge of the BACnet network number for proper operation.*

[3] *Any standard property defined for this object type and not listed in this table may be present. If the presence or capability of one of these properties require other properties to be present, then the other properties shall be present.*

~~For convenience, the following table further illustrates the properties that are required based on the value of the Network_Type property and the various capabilities of the network types. This table is not meant to be an exhaustive list and is shown for informative purposes.~~

~~Table 12-72.~~ ~~Required Properties of the Network Port Object Type Based on Network_Type when Protocol_Level is BACNET_APPLICATION.~~

~~Network_Type~~	~~Additional Required Properties~~
~~ETHERNET~~	~~MAC_Address~~
~~IPV4~~ ~~(BACnet_IP_Mode is NORMAL)~~ ~~-~~ ~~-~~ ~~-~~ ~~-~~ ~~-~~	~~MAC_Address~~ ~~BACnet_IP_Mode~~ ~~IP_Address~~ ~~BACnet_IP_UDP_Port~~ ~~IP_Subnet_Mask~~ ~~IP_Default_Gateway~~ ~~IP_DNS_Server~~
~~IPV4~~ ~~(BACnet_IP_Mode is FOREIGN)~~ ~~-~~ ~~-~~ ~~-~~ ~~-~~ ~~-~~ ~~-~~ ~~-~~	~~MAC_Address~~ ~~BACnet_IP_Mode~~ ~~IP_Address~~ ~~BACnet_IP_UDP_Port~~ ~~IP_Subnet_Mask~~ ~~IP_Default_Gateway~~ ~~IP_DNS_Server~~ ~~FD_BBMD_Address~~ ~~FD_Subscription_Lifetime~~

~~IPV4~~ ~~(BACnet_IP_Mode is BBMD)~~ - - - - - - - -	~~MAC_Address~~ ~~BACnet_IP_Mode~~ ~~IP_Address~~ ~~BACnet_IP_UDP_Port~~ ~~IP_Subnet_Mask~~ ~~IP_Default_Gateway~~ ~~IP_DNS_Server~~ ~~BBMD_Broadcast_Distribution_Table~~ ~~BBMD_Accept_FD_Registrations~~ ~~BBMD_Foreign_Device_Table~~
~~IPV6~~ ~~(BACnet_IPv6_Mode is NORMAL)~~	~~MAC_Address~~ ~~BACnet_IPv6_Mode~~ ~~IPv6_Prefix_Length~~ ~~IPv6_Address~~ ~~BACnet_IPv6_UDP_Port~~ ~~BACnet_IPv6_Multicast_Address~~ ~~IPv6_Default_Gateway~~ ~~IPv6_DNS_Server~~
~~IPV6~~ ~~(BACnet_IPv6_Mode is FOREIGN)~~	~~MAC_Address~~ ~~BACnet_IPv6_Mode~~ ~~IPv6_Prefix_Length~~ ~~IPv6_Address~~ ~~BACnet_IPv6_UDP_Port~~ ~~BACnet_IPv6_Multicast_Address~~ ~~IPv6_Default_Gateway~~ ~~IPv6_DNS_Server~~ ~~FD_BBMD_Address~~ ~~FD_Subscription_Lifetime~~
~~IPV6~~ ~~(BACnet_IPv6_Mode is BBMD)~~	~~MAC_Address~~ ~~BACnet_IPv6_Mode~~ ~~IPv6_Prefix_Length~~ ~~IPv6_Address~~ ~~BACnet_IPv6_UDP_Port~~ ~~BACnet_IPv6_Multicast_Address~~ ~~IPv6_Default_Gateway~~ ~~IPv6_DNS_Server~~ ~~BBMD_Broadcast_Distribution_Table~~ ~~BBMD_Accept_FD_Registrations~~ ~~BBMD_Foreign_Device_Table~~
~~MSTP (Slave node)~~	~~MAC_Address~~
~~MSTP (Master node)~~ - -	~~MAC_Address~~ ~~Max_Master~~ ~~Max_Info_Frames~~

MSTP (capable of Slave Proxy)	MAC_Address
-	Max_Master
-	Max_Info_Frames
-	Slave_Proxy_Enable
-	Manual_Slave_Address_Binding
-	Auto_Slave_Discovery
-	Slave_Address_Binding

Table 12-73. Expected Properties of the Network Port Object Type by Network_Type and Protocol_Level.

Network_Type	Protocol_Level	Properties	Conformance
ARCNET ETHERNET LONTALK VIRTUAL ZIGBEE <proprietary values>	PHYSICAL	MAC_Address Link_Speed Link_Speeds Link_Speed_Autonegotiate Network_Interface_Name	R R O O O
SERIAL	PHYSICAL	Link_Speed Link_Speeds Link_Speed_Autonegotiate Network_Interface_Name	R O O O
IPV4	PROTOCOL	IP_Address IP_Subnet_Mask IP_Default_Gateway IP_DNS_Server IP_DHCP_Enable IP_DHCP_Lease_Time IP_DHCP_Lease_Time_Remaining IP_DHCP_Server	R R R R O O O O
IPV6	PROTOCOL	IPv6_Address IPv6_Prefix_Length IPv6_Default_Gateway IPv6_DNS_Server IPv6_Auto_Addressing_Enable IPv6_DHCP_Lease_Time IPv6_DHCP_Lease_Time_Remaining IPv6_DHCP_Server IPv6_Zone_Index	R R R R O O O O O
MSTP	PROTOCOL	MAC_Address Max_Master Max_Info_Frames	R R R
PTP	PROTOCOL		
<proprietary values>	PROTOCOL		
<proprietary values>	NON_BACNET_APPLICATION		
any (except SERIAL)	BACNET_APPLICATION	all properties	

Table 12-74. Properties of the Network Port Object Type Only Used when Protocol_Level is BACNET_APPLICATION.

Network_Type	Properties

~~ETHERNET~~ ~~ARCNET~~ ~~LONTALK~~ ~~PTP~~ ~~VIRTUAL~~ ~~<proprietary values>~~	
~~ZIGBEE~~	~~Virtual_MAC_Address_Table~~
~~MSTP~~	~~Slave_Proxy_Enable~~ ~~Manual_Slave_Address_Binding~~ ~~Auto_Slave_Discovery~~ ~~Slave_Address_Binding~~
~~IPV4~~	~~BACnet_IP_Mode~~ ~~BACnet_IP_UDP_Port~~ ~~BACnet_IP_Multicast_Address~~ ~~BACnet_IP_NAT_Traversal~~ ~~BACnet_IP_Global_Address~~ ~~BBMD_Broadcast_Distribution_Table~~ ~~BBMD_Accept_FD_Registrations~~ ~~BBMD_Foreign_Device_Table~~ ~~FD_BBMD_Address~~ ~~FD_Subscription_Lifetime~~
~~IPV6~~	~~Virtual_MAC_Address_Table~~ ~~BACnet_IPv6_Mode~~ ~~BACnet_IPv6_UDP_Port~~ ~~BACnet_IPv6_Multicast_Address~~ ~~BBMD_Broadcast_Distribution_Table~~ ~~BBMD_Accept_FD_Registrations~~ ~~BBMD_Foreign_Device_Table~~ ~~FD_BBMD_Address~~ ~~FD_Subscription_Lifetime~~
~~all~~	~~Network_Number~~ ~~Network_Number_Quality~~ ~~APDU_Length~~ ~~Routing_Table~~

[Network Port Property Changes]

[Change **Clause 12.56.8**. p. 523]

12.56.8 Network_Type

This property, of type BACnetNetworkType, represents the type of network this Network Port object is representing.

This property shall have one of the following values:

ARCNET	
IPV4	
IPV6	
ETHERNET	
LONTALK	

MSTP	MS/TP, as defined in Clause 9.
PTP	Point-To-Point, as defined in Clause 10.
SERIAL	A physical serial port.
ZIGBEE	
VIRTUAL	Indicates that this port represents the configuration and properties of a virtual network as described in Clause H.2.
SECURE_CONNECT	*BACnet Secure Connect virtual link layer as defined in Annex YY.*
WEBSOCKET	*WebSocket protocol as defined in RFC 6455.*
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values to indicate that this port represents the use of message structures, procedures, and medium access control techniques other than those contained in this standard. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

When the Protocol_Level is BACNET_APPLICATION, the Network_Type indicates the protocol over which BACnet is operating and implies that the requirements laid out in the appropriate clause are being met. For example, if the Network_Type is IPV4, then the port is operating as a BACnet/IP port as defined in Annex J.

[Change **Clause 12.56.9**. p. 524]

12.56.9 Protocol_Level

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).

[Change **Clause 12.56.10**. p. 524]

12.56.10 Reference_Port

. . .

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 is PHYSICAL, then this property shall have a value of 4194303, if present.

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.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

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.

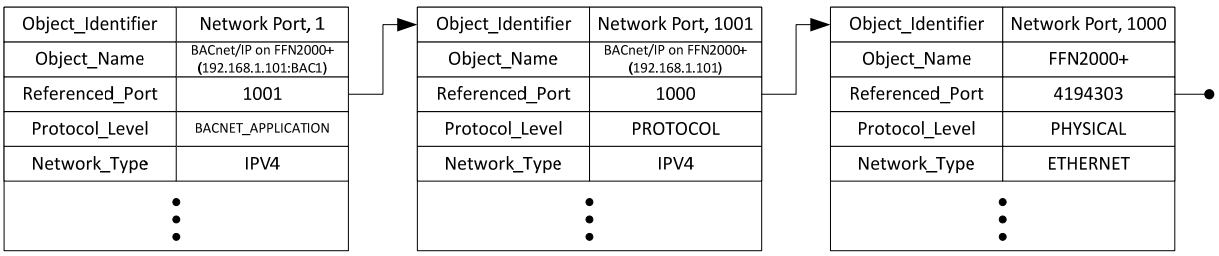


Figure 12-19. Example Network Port Hierarchy Chain

12.56.10.1.1 Property Inheritance

In a hierarchy chain of Network Port objects, a Network Port object with a Protocol_Level of BACNET_APPLICATION shall inherit property values related to configuration of the protocol or physical port from all Network Port objects in the chain.

Where a property is specified in multiple Network Port objects in the hierarchy chain, the property's value in the Network Port object nearest to the top of the chain shall be the value reflected in the topmost Network Port object. For example, in a Network Port object with a Protocol_Level of BACNET_APPLICATION, the Network_Type property shall be the same as the Network_Type of the directly referenced Network Port object.

~~Network Port objects with a Protocol_Level of NON_BACNET_APPLICATION are allowed to inherit property values.~~ *Only* Network Port objects with a Protocol_Level *of* ~~other than~~ BACNET_APPLICATION or NON_BACNET_APPLICATION shall ~~not~~ inherit property values.

Property inheritance allows clients to read and write current network settings by accessing the topmost Network Port object. ~~It is required that inherited properties which are writable in the source (lower) Network Port object be writable in the inheriting Network Port object.~~ *If a property is writable at a lower level, it shall also be writable at the topmost Network Port object.*

When inherited properties are written, in either the source or the inheriting object, the values shall be written through to the other Network Port object. It is acceptable to make inherited properties in the source object read only and the corresponding properties in the inheriting object be writable.

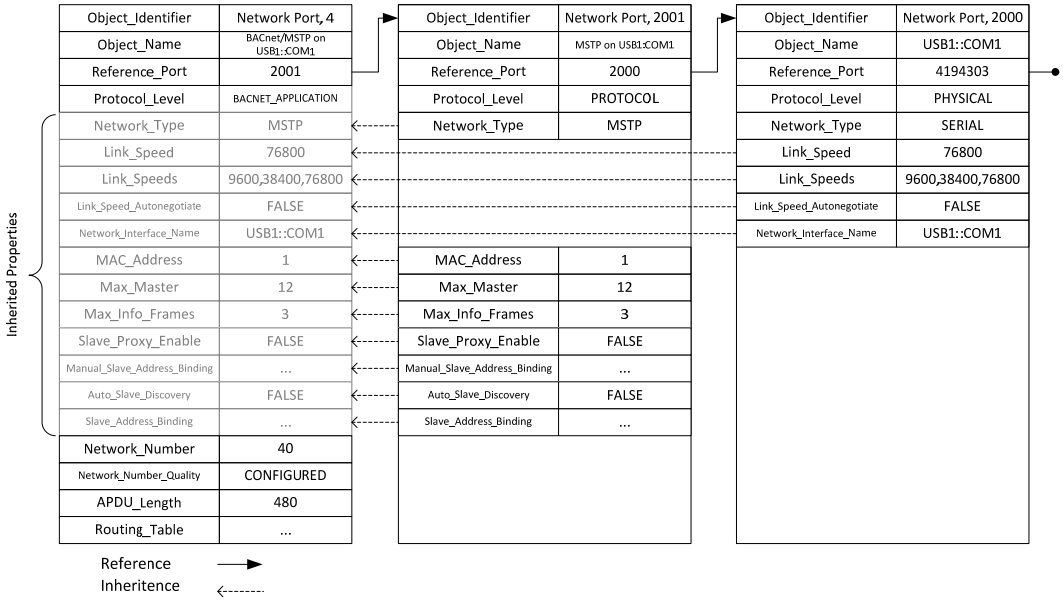


Figure 12-20. Example Network Port Hierarchy Chain Showing Property Inheritance

The below figure demonstrates the use of hierarchies of Network Port objects and how property inheritance is accomplished. This example configuration includes multiple Network Port objects with a Protocol_Level of BACNET_APPLICATION shown on the left, some Network Port objects with a Protocol_Level of PROTOCOL, including a shared Network Port object of Network_Type IPV4, and one Network Port object with a Protocol_Level of PHYSICAL, shown on the top right, which represents the single physical network port used.

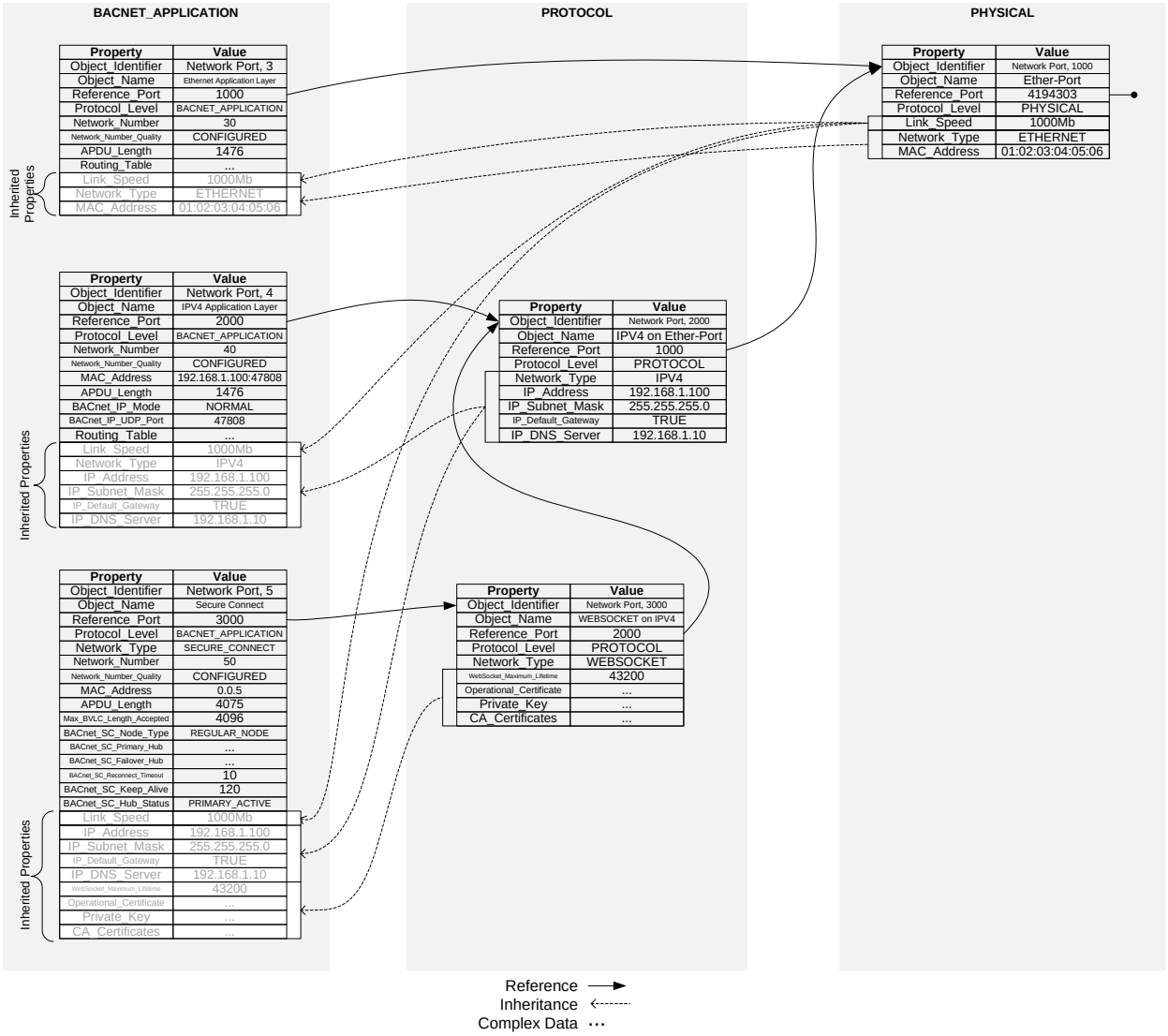


Figure 12-x. *Example Multiple Network Port objects with Protocol_Level of PROTOCOL*

12.56.10.1.2 Pending Changes

[No change in this clause.]

[Change **Clause 12.56.11**, p. 526]

12.56.11 Network_Number

This property, of type Unsigned16, represents the BACnet network number associated with this network.

The range for this property shall be 0 .. 65534. A Network_Number of 0 indicates that the Network_Number is not known or cannot be determined. Writing 0 to the Network_Number property shall force the value of the Network_Number_Quality property to UNKNOWN and allows the device to attempt to learn the network number, if possible. Writing a value other than 0 shall force the Network_Number_Quality property to CONFIGURED.

If *this property is not required based on* the Network_Type *and Protocol_Level* ~~is PTP~~, then this property, *if present,* shall be read-only and contain a value of 0.

This property shall be writable in routers, secure devices, and any other device that requires knowledge of the network number for proper operation. Routers are permitted to refuse a value of 0. In that case, the write request shall result in an error response with 'Error Class' of PROPERTY and an 'Error Code' of VALUE_OUT_OF_RANGE.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

[Change **Clause 12.56.12**, p. 526]

12.56.12 Network_Number_Quality

This read-only property, of type BACnetNetworkNumberQuality, represents the current quality of the Network_Number property. If *this property is not required but present based on* the Network_Type *and Protocol_Level* ~~is PTP~~, *then this property* ~~the Network_Number_Quality~~, *if present,* shall be CONFIGURED.

This property shall have one of the following values:

UNKNOWN	None of the below meanings.
LEARNED	The Network_Number was learned via receipt of a Network-Number-Is message with a flag value of 0 (learned).
LEARNED_CONFIGURED	The Network_Number was learned via receipt of a Network-Number-Is message with a flag value of 1 (configured).
CONFIGURED	The Network_Number is configured for this port.

[Change **Clause 12.56.15**, p. 530]

12.56.15 MAC_Address

This property, of type OCTET STRING, contains the BACnet MAC address used on this network. The value of this property shall be conveyed with the most significant octet first. ~~If Network_Type is IPV4 and the Protocol_Level is BACNET_APPLICATION, then the value of this property shall contain the six octet combination of the IP_Address and BACnet_IP_UDP_Port and shall be read-only~~. ~~If the value of Network_Type is a value that represents a port that requires VMAC addressing, then the value of this property shall be read-only and contain the VMAC address.~~

This property, if present and the Protocol_Level is BACNET_APPLICATION, shall contain the MAC address for this port in the format as defined for the SADR and DADR fields of the NPDU header for the Network_Type. See Clause 6.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. The value of this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

[Network Port Properties for WebSockets]

[Add new properties to the Network Port object in **Clause 12.56**, p. 539]

12.56.X1 WebSocket_Maximum_Lifetime

This property, of type Unsigned, indicates the maximum lifetime in minutes for a WebSocket connection. WebSocket connections shall not be maintained for longer than this time. If the maximum lifetime is reached, the WebSocket connection shall be closed and, if required, re-established. See Clause YY.7.5.5.

If this property is not writable, the value of this property shall be in the range 120..2880. If this property is writable, it shall support a value in the range 10..2880 at a minimum.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X2 WebSocket_Plain_Server_Port

This optional property, of type Unsigned16, indicates the TCP port accepting plain WebSocket connections without TLS. If the network port accepts plain WebSocket connections, this property shall support being configured in the range 0..65535. The default value for this property shall be 80. See RFC 6455, Clause 1.7. If the network port is configured to not accept plain WebSocket connections, this property shall have a value of zero, if present.

A successful change to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X3 WebSocket_Secure_Server_Port

This optional property, of type Unsigned16, indicates the TCP port accepting secured WebSocket connections over TLS. If the network port accepts secure WebSocket connections, this property shall support being configured in the range 0..65535. The default value for this entry shall be 443. See RFC 6455, Clause 1.7. If the network port is configured to not accept secure WebSocket connections, this property shall have a value of zero, if present.

A successful change to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X4 WebSocket_Protocol

This property, of type CharacterString, shall indicate the sub-protocol used on all WebSocket connections of the network port this object represents. The value of this property shall be a character string formatted as defined in RFC 6455 Section 4.1.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X5 WebSocket_Payload_Type

This property, of type BACnetWebSocketPayloadType, indicates the type of the payload data conveyed across the WebSocket connections of the network port this object represents. The values defined for this property are:

UNKNOWN	The type of the payload is unknown.
TEXT_PAYLOAD	The payload is UTF-8 text.
BINARY_PAYLOAD	The payload is binary data.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X6 WebSocket_PingPong

This property, of type BOOLEAN, indicates whether (TRUE) the WebSocket connections are maintained using Ping and Pong WebSocket messages, or no Ping and Pong messages shall be used (FALSE).

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X7 Credentials_Ref

This property, of type BACnetObjectIdentifier, indicates the Network Port object from which the values of the Operational_Certificate, the CA_Certificates, and the Private_Key properties are used by this object for establishing secure WebSocket connections.

The Operational_Certificate, the CA_Certificates, and the Private_Key properties are configured at the Network Port object being referenced by this property. The respective local properties, if present, shall not be writable and if readable, shall mirror the referenced Network Port object properties.

If this property is present and the instance portion of the object identifier is 4194303, then the Network port object from which to use the operational credentials is not configured, or the operational credentials are required to be configured in this object.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X8 Operational_Certificate

This property, of type OCTET STRING, shall represent the X.509 certificate used to identify the IP host of the port.

The value of this property shall be an X.509 certificate in binary DER format. If no operational certificate is configured, this property, if present, shall be a zero length octet string.

This property shall support certificates up to 2048 octets in size at a minimum. Support of larger certificates is optional.

If the Credentials_Ref property is present and has an instance other than 4194303, then this property shall be read-only and mirror the content of the Operational_Certificate property of the object referenced by the Credentials_Ref property.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X9 Private_Key

This write-only property, of type OCTET STRING, shall represent the private key related to the operational certificate present in the Operational_Certificate property.

The value of this property shall be a private key in PKCS #8 binary DER format (RFC 5858).

If this property is writable, writing a zero length octet string shall delete the private key.

Attempts to read this property shall return a Result(-) with an 'Error Class' of PROPERTY and an 'Error Code' of READ_ACCESS_DENIED.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. The value of this property shall become effective when the port receives a ReinitializeDevice service request with a 'Reinitialized State of Port' of ACTIVATE_CHANGES or WARMSTART.

If the port supports only an internally preset or generated private key and supports the Certificate_Signing_Request and Certificate_Signing_Parameters properties, then a write attempt to this property shall return a Result(-) with an 'Error Class' of PROPERTY and an 'Error Code' of WRITE_ACCESS_DENIED.

12.56.X10 CA_Certificates

This property, of type BACnetARRAY[N] of OCTET STRING, shall represent the certificate authorities for X.509 public certificates the port accepts when it is establishing a WebSocket connection.

Each OCTET STRING shall be an X.509 certificate in binary DER format. If no CA certificate is present in an array element, the OCTET STRING shall be of zero length.

This property shall support certificates up to 2048 octets in size in each element, and shall support at least one entry. Support of larger certificates as well as multiple CA certificates is optional.

If the Credentials_Ref property is present and has an instance other than 4194303, then this property shall be read-only and mirror the content of the CA_Certificates property of the object referenced by the Credentials_Ref property.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X11 Certificate_Signing_Request

This read-only property, of type OCTET STRING, shall represent the PKCS #10 certificate signing request as defined by RFC5967 that can be used by a certificate authority to create an operational certificate, matching an internally held private key. See Clause YY.7.4.1.

The certificate presented in this property shall contain the parameters configured in the Certificate_Signing_Parameters property. The public keys and other parameters contained in this certificate shall be related to an internal private key. This private key is not exposed and cannot be configured from any remote entity. This private key is expected to be pre-assigned or generated and stored by some secure internal function of the port, for example, a hardware security module.

The certificate presented in this property shall be updated only when the Certificate_Signing_Parameters property is changed and activated. Whether the port is generating a new internal private key and public key for the updated certificate is a local matter.

The certificate presented in this property allows some external entity to initiate a certificate signing request to a CA to create an operational certificate signed by the CA. The selection of the signing CA is a site local matter.

If no certificate signing request is available, this property, if present, shall be a zero length octet string.

The resulting operational certificate signed by the CA can then be used by the external entity to configure the Operational Certificate property and will match the port's internal private key. The Private_Key property shall not be written in this case.

12.56.X12 Certificate_Signing_Parameters

This property, of type BACnetCertificateSigningParameters, contains additional signing parameters to be incorporated into the certificate presented in the Certificate_Signing_Request property after activating changes to this property. See Clause YY.7.4.1.

The signing parameters are defined as follows:

Common Name	This parameter, of type CharacterString, shall indicate the common name (CN) to be contained in the signed certificate.
Country Name	This parameter, of type CharacterString, shall indicate the name of the country (C) to be contained in the signed certificate.
State or Province Name	This parameter, of type CharacterString, shall indicate the name of state or province (ST) to be contained in the signed certificate.
Locality Name	This parameter, of type CharacterString, shall indicate the name of the locality (L) to be contained in the signed certificate.
Organization Name	This parameter, of type CharacterString, shall indicate the name of the organization (O) to be contained in the signed certificate.
Organizational Unit	This optional parameter, of type CharacterString, shall indicate the name of the organization (OU) to be contained in the signed certificate.
Alternative Names	This optional parameter, of type SEQUENCE OF CharacterString, shall indicate the alternative names to be contained in the 'subjectAltName' field in the signed certificate.

If this property is not configured, then all parameters shall be empty character strings, and optional parameters shall be absent.

A successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.X13 WebSocket_Connections

This property, of type BACnetLIST of BACnetWebSocketStatus, represents the current WebSocket connections maintained by the WebSocket network port.

Each element of this list is a structure as defined in Clause YY.7.5

[Network Port Properties for BACnet/SC]

12.56.Y1 Max_BVLC_Length_Accepted

This property, of type Unsigned, shall indicate the maximum size in octets of the BVLC message the network port can receive and process.

12.56.Y2 BACnet_SC_Node_Type

This property, of type BACnetSCNodeType, indicates whether this network port is a regular node, a primary hub node, or failover hub node. See Annex YY.

REGULAR_NODE	This network port is a BACnet/SC regular node which at least initiates a WebSocket connection to BACnet/SC primary and failover hub nodes.
PRIMARY_HUB	This network port is a BACnet/SC primary hub node.
FAILOVER_HUB	This network port is a BACnet/SC failover hub node.

Attempts to write a value for a node type other than supported by the network port shall cause a Result(-) be returned with an 'Error Class' of OBJECT and an 'Error Code' of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.Y3 BACnet_SC_Primary_Hub

This property, of type BACnetSCConnectionPeer, provides the WebSocket URI for the BACnet/SC primary hub. If no WebSocket URI is configured, this property shall have a value of NULL.

This property shall contain a WebSocket URI if the BACnet/SC node is of type REGULAR_NODE or FAILOVER_HUB. If no WebSocket URI is configured when one of these types, the Reliability property shall be CONFIGURATION_ERROR.

This property may contain NULL if the BACnet/SC node is of type PRIMARY_HUB.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.Y4 BACnet_SC_Failover_Hub

This property, of type BACnetSCConnectionPeer, provides the WebSocket URI for the BACnet/SC failover hub. If no WebSocket URI is configured, this property shall have a value of NULL.

This property may contain a WebSocket URI if the BACnet/SC node is of type REGULAR_NODE.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.Y5 BACnet_SC_Reconnect_Timeout

This property, of type Unsigned, specifies the minimum time in seconds between each attempt to connect to a hub.

Reconnection strategies and timeouts between attempts to connect to other BACnet/SC nodes are generally a local matter. The minimum time between attempts for such WebSocket connections shall be as specified by this property. See Clause YY.6.3.3.

If this property is writable, it shall support a minimum range of 2..300. If this property is not writable, it shall have a value in the range 10..30.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.Y6 BACnet_SC_Keep_Alive

This property, of type Unsigned, identifies the time in seconds of BVLC message inactivity on a WebSocket connection before an Advertisement-Solicitation and Advertisement message exchange is initiated by the BACnet/SC port being the WebSocket client. See Clause YY.6.3.2.

If this property is writable, it shall support a minimum range of 30..300. If this property is not writable, it shall have a value in the range 30..300.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

12.56.Y7 BACnet_SC_Primary_Hub_Connection

This optional property, of type Unsigned, identifies the WebSocket connection to the BACnet/SC primary hub in the BACnet_SC_Connection_Status property. A value of zero indicates that no established WebSocket connection to the primary hub exists.

12.56.Y8 BACnet_SC_Failover_Hub_Connection

This optional property, of type Unsigned, identifies the WebSocket connection to the BACnet/SC failover hub in the BACnet_SC_Connection_Status property. A value of zero indicates that no established WebSocket connection to the failover hub exists.

12.56.Y9 BACnet_SC_Hub_Status

This property, of type BACnetSCHubStatus, shall indicate the status of the connection of the BACnet/SC node to the active hub. For the values this property may take on, see Clause YY.6.

12.56.Y10 BACnet_SC_Connections

This optional property, of type BACnetLIST of BACnetSCConnectionStatus, shall indicate the WebSocket connections initiated and accepted by the BACnet/SC network port.

For the details of the BACnetSCConnectionStatus structure see Clause YY.6.3.1

12.56.Y11 BACnet_SC_Manual_Node_Bindings

This optional property, of type BACnetLIST of BACnetSCNodeBinding, is configured with the WebSocket URIs to be used for directly connecting to other nodes, identified by their VMAC address. See Clause YY.6.3.4.

If this property is writable, then a successful write to this property shall set the Changes_Pending property to TRUE. A value written to this property shall become effective when the device receives a ReinitializeDevice service request with a 'Reinitialized State of Device' of ACTIVATE_CHANGES or WARMSTART.

135-2016*bj*-5. Add and Extend ASN.1 Types for BACnet/SC

Rationale

This addendum section collects the APDU header parameter extensions and the definitions of the new data types and other changes to Clause 21 required for this addendum.

[Clause 20 APDU Header Parameter Extension]

[Change Clause **20.1.2.5 max-apdu-length-accepted**, p. 759]

20.1.2.5 max-apdu-length-accepted

This parameter specifies the maximum size of a single APDU that the issuing device will accept. This parameter is included in the confirmed request so that the responding device may determine how to convey its response. The parameter shall be encoded as follows:

B'0000' Up to MinimumMessageSize (50 octets)
B'0001' Up to 128 octets
B'0010' Up to 206 octets (fits in a LonTalk frame)
B'0011' Up to 480 octets (fits in an ARCNET frame)
B'0100' Up to 1024 octets
B'0101' Up to 1476 octets (fits in *a 1497 octet NPDU in* one Ethernet frame)
B'0110' ~~reserved by ASHRAE~~ *Up to 4075 octets (fits in a 4096 octet NPDU)*
B'0111' ~~reserved by ASHRAE~~ *Up to 8171 octets (fits in an 8192 octet NPDU)*
B'1000' ~~reserved by ASHRAE~~ *Up to 16363 octets (fits in a 16384 octet NPDU)*
B'1001' ~~reserved by ASHRAE~~ *Up to 32747 octets (fits in a 32768 octet NPDU)*
B'1010' ~~reserved by ASHRAE~~ *Up to 65512 octets (fits in a 65533 octet NPDU)*
B'1011' reserved by ASHRAE
B'1100' reserved by ASHRAE
B'1101' reserved by ASHRAE
B'1110' reserved by ASHRAE
B'1111' reserved by ASHRAE

[Clause 21 New Productions]

[Add new ASN.1 productions to **Clause 21** section "**Base Types**" maintaining the alphabetical order, p. 805]

```
BACnetAuthenticationData ::= CHOICE {
    user                [0] SEQUENCE {
                            user-identifier     [0] Unsigned16,
                            user-role           [1] Unsigned8
                            },
    vendor-specific     [200] SEQUENCE {
                            vendor-identifier                   [0] Unsigned16,
                            authentication-mechanism            [1] Unsigned8,
                            authentication-data                 [2] OCTET STRING
                            }
    }
    -- Context tag numbers lower than 200 are equal to the standard user authentication
    -- mechanism value. See Clause 24.2.10.
```



```
BACnetCertificateSigningParameters ::= SEQUENCE {
        common-name                 [0] CharacterString,
        country-name                [1] CharacterString,
        state-or-province-name      [2] CharacterString,
        locality-name               [3] CharacterString,
        organization-name           [4] CharacterString,
        organizational-unit         [5] CharacterString OPTIONAL,
        alternative-names           [6] SEQUENCE OF CharacterString OPTIONAL
}

BACnetSCConnectionPeer ::= CHOICE {
        none            [0] NULL,
        uri             [1] CharacterString
}

BACnetConnectionStatus ::= ENUMERATED {
        unknown             (0),
        disabled            (1),
        disconnected        (2),
        connecting          (3),
        connected           (4),
        connection-failed   (5),
        accepting           (6),
        accepted            (7)
}

BACnetSCNodeBinding ::= SEQUENCE {
        vmac                        [0] OCTET STRING SIZE(3),
        uri                         [1] CharacterString
}

BACnetSCNodeType ::= ENUMERATED {
        regular-node        (0),
        primary-hub         (1),
        failover-hub        (2)
}

BACnetSCConnectionStatus ::= SEQUENCE {
        connection                  [0] Unsigned,
        connection-status           [1] BACnetConnectionStatus,
        vmac                        [2] OCTET STRING SIZE(3),
        peer-node-type              [3] BACnetSCNodeType,
        active-hub                  [4] BOOLEAN;
        error                       [5] Error OPTIONAL,
        details                     [6] CharacterString OPTIONAL
}

BACnetWebSocketPayloadType ::= ENUMERATED {
        unknown             (0),
        text-payload        (1),
        binary-payload      (2)
}
```

BACnetWebSocketStatus ::= SEQUENCE {
    websocket-connection   [0] Unsigned,
    connection-status   [1] BACnetConnectionStatus,
    uri   [2] CharacterString OPTIONAL,
    error   [3] Error,
    details   [4] CharacterString OPTIONAL
}

BACnetSCHubStatus ::= ENUMERATED {
    unknown   (0),
    disabled   (1),
    self-active   (2),
    none   (3),
    connecting   (4),
    primary-active (5),
    failover-active (6),
    both-active   (7),
    both-inactive   (8)
}

[Clause 21 Changes to Existing Productions]

[Add new error codes to the **Error** production in **Clause 21**, p. 798]

Error ::= SEQUENCE {
-- NOTE: The valid combinations of error-class and error-code are defined in Clause 18.
error-class ENUMERATED {
...
},
-- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
-- 64-65535 may be used by others subject to the procedures and constraints described
-- in Clause 23.

error-code ENUMERATED { -- see below for numerical order
...

[Insert the new error codes as added in the numerical order list below into the alphabetical order list, maintaining the alphabetical order]

...
-- numerical order reference
...
-- see invalid-value-in-this-state (138),
-- *see not-the-active-hub* *(?),*
-- *see data-not-available* *(?),*
-- *see http-unexpected-response-code* *(?),*
-- *see http-no-upgrade* *(?),*
-- *see http-resource-not-local* *(?),*
-- *see http-proxy-authentication-failed* *(?),*
-- *see http-response-timeout* *(?),*
-- *see http-response-syntax-error* *(?),*
-- *see http-response-value-error* *(?),*
-- *see http-response-missing-header* *(?),*
-- *see http-websocket-header-error* *(?),*
-- *see http-upgrade-required* *(?),*
-- *see http-upgrade-error* *(?),*
-- *see http-temporary-unavailable* *(?),*
-- *see http-not-a-server* *(?),*
-- *see http-other-error* *(?),*
-- *see websocket-scheme-not-supported* *(?),*
-- *see websocket-unknown-control-message* *(?),*
-- *see websocket-close-error* *(?),*
-- *see websocket-closed-by-peer* *(?),*
-- *see websocket-endpoint-leaves* *(?),*
-- *see websocket-protocol-error* *(?),*
-- *see websocket-data-not-accepted* *(?),*
-- *see websocket-closed-abnormally* *(?),*
-- *see websocket-data-inconsistent* *(?),*
-- *see websocket-data-against-policy* *(?),*
-- *see websocket-frame-too-long* *(?),*
-- *see websocket-extension-missing* *(?),*
-- *see websocket-request-unavailable* *(?),*
-- *see websocket-other-error* *(?),*
-- *see tls-client-certificate-error* *(?),*
-- *see tls-server-certificate-error* *(?),*

```
                -- see tls-client-authentication-failed        (?),
                -- see tls-server-authentication-failed        (?),
                -- see tls-client-certificate-expired          (?),
                -- see tls-server-certificate-expired          (?),
                -- see tls-client-certificate-revoked          (?),
                -- see tls-server-certificate-revoked          (?),
                -- see tls-other-error                         (?),
                -- see tcp-connect-timeout                     (?),
                -- see tcp-connection-refused                  (?),
                -- see tcp-closed-by-local                     (?),
                -- see tcp-closed-other                        (?),
                -- see tcp-other-error                         (?),
                -- see ip-address-not-reachable                (?),
                -- see ip-other-error                          (?),
                ...
                }
                -- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
                -- 256-65535 may be used by others subject to the procedures and constraints described
                -- in Clause 23.
        }
```

[Change **BACnetNetworkType** in **Clause 21**, p. 836]

```
BACnetNetworkType ::= ENUMERATED {
        ethernet        (0),
        arcnet          (1),
        mstp            (2),
        ptp             (3),
        lontalk         (4),
        ipv4            (5),
        zigbee          (6),
        virtual         (7),
        -- formerly: non-bacnet (8), removed in version 1 revision 18
        ipv6            (9),
        serial          (10),
        secure-connect  (?),
        websocket       (?),
        ...
        }
-- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
-- 64-255 may be used by others subject to the procedures and constraints described
-- in Clause 23.
```

[Insert into production **BACnetPropertyIdentifier** in **Clause 21**, preserving the alphabetical and numerical order, p. 845]

```
        ...
        bacnet-sc-connections                (?),
        bacnet-sc-failover-hub               (?),
        bacnet-sc-failover-hub-connection    (?),
        bacnet-sc-hub-status                 (?),
        bacnet-sc-keep-alive                 (?),
        bacnet-sc-manual-node-bindings       (?),
        bacnet-sc-node-type                  (?),
```



```
        bacnet-sc-primary-hub                   (?),
        bacnet-sc-primary-hub-connection        (?),
        bacnet-sc-reconnect-timeout             (?),
        ca-certificates                         (?),
        certificate-signing-parameters          (?),
        certificate-signing-request             (?),
        credentials-ref                         (?),
        max-bvlc-length-accepted                (?),
        operational-certificate                 (?),
        private-key                             (?),
        websocket-connections                   (?),
        websocket-maximum-lifetime              (?),
        websocket-payload-type                  (?),
        websocket-pingpong                      (?),
        websocket-plain-server-ports            (?),
        websocket-protocol                      (?),
        websocket-secure-server-ports           (?),
        ...
-- numerical order reference
        ...
        -- see bacnet-sc-connections            (?),
        -- see bacnet-sc-failover-hub           (?),
        -- see bacnet-sc-failover-hub-connection (?),
        -- see bacnet-sc-hub-status             (?),
        -- see bacnet-sc-keep-alive             (?),
        -- see bacnet-sc-manual-node-bindings   (?),
        -- see bacnet-sc-node-type              (?),
        -- see bacnet-sc-primary-hub            (?),
        -- see bacnet-sc-primary-hub-connection (?),
        -- see bacnet-sc-reconnect-timeout      (?),
        -- see ca-certificates                  (?),
        -- see max-bvlc-length-accepted         (?),
        -- see operational-certificate          (?),
        -- see private-key                      (?),
        -- see certificate-signing-request      (?),
        -- see certificate-signing-parameters   (?),
        -- see credentials-ref                  (?),
        -- see websocket-connections            (?),
        -- see websocket-maximum-lifetime       (?),
        -- see websocket-payload-type           (?),
        -- see websocket-pingpong               (?),
        -- see websocket-plain-server-ports     (?),
        -- see websocket-protocol               (?),
        -- see websocket-secure-server-ports    (?),
        ...
        }
-- The special property identifiers ...
```

[Insert into production **BACnetPropertyStates** in **Clause 21**, p. 862]

```
        connection-status                       [?] BACnetConnectionStatus,
        bacnet-sc-hub-status                    [?] BACnetSCHubStatus,
```

135-2016*bj*-6. New Error Codes for BACnet/SC

Rationale

This addendum section introduces new error codes for BACnet/SC

[Add new error codes to **Clause 18.7**, p. 740]

18.7 Error Class - COMMUNICATION

...

NOT_THE_ACTIVE_HUB *- The node received a message that would require it to be the active hub, but the node is not the active hub at this time.*

DATA_NOT_AVAILABLE *- The data as requested is not available for the requester.*

HTTP_UNEXPECTED_RESPONSE_CODE *- Server reports unexpected response code.*

HTTP_NO_UPGRADE *- Server does not accept upgrade to the WebSocket protocol.*

HTTP_RESOURCE_NOT_LOCAL *- Redirect to another location of the peer WebSocket port received.*

HTTP_PROXY_AUTHENTICATION_FAILED *- Proxy Authentication failed*

HTTP_RESPONSE_TIMEOUT *- No response from server within timeout.*

HTTP_RESPONSE_SYNTAX_ERROR *- Syntax error in HTTP response received.*

HTTP_RESPONSE_VALUE_ERROR *- Errors in values of the HTTP response received.*

HTTP_RESPONSE_MISSING_HEADER *- Missing header fields in response.*

HTTP_WEBSOCKET_HEADER_ERROR *- Response contains any other error in HTTP header fields.*

HTTP_UPGRADE_REQUIRED *- No upgrade request was received by the server.*

HTTP_UPGRADE_ERROR *- Upgrading to the WebSocket protocol failed.*

HTTP_TEMPORARY_UNAVAILABLE *- No more HTTP connections are available currently.*

HTTP_NOT_A_SERVER *- No inbound requests are supported. The host is not an HTTP server.*

HTTP_OTHER_ERROR *- Some unspecified error occurred in HTTP.*

WEBSOCKET_SCHEME_NOT_SUPPORTED *- The WebSocket URI presented to the WebSocket port indicates a scheme whose respective protocol variant is not supported by the WebSocket port.*

WEBSOCKET_CLOSE_ERROR *- An error occurred in closing the WebSocket connection.*

WEBSOCKET_CLOSED_BY_PEER *- The WebSocket connection was closed by the peer.*

WEBSOCKET_ENDPOINT_LEAVES *- An endpoint is "going away", such as a server going down or a client having left the WebSocket connection.*

WEBSOCKET_PROTOCOL_ERROR *- An endpoint has closed the WebSocket connection due to a protocol error.*

WEBSOCKET_DATA_NOT_ACCEPTED *- An endpoint has closed the WebSocket connection due to data of a type not accepted.*

WEBSOCKET_CLOSED_ABNORMALLY *- The WebSocket connection was closed abnormally.*

WEBSOCKET_DATA_INCONSISTENT *- An endpoint has closed the WebSocket connection due to data received that is inconsistent with the type of the message.*

WEBSOCKET_DATA_AGAINST_POLICY *- An endpoint has closed the WebSocket connection due to data received that is violating its policy.*

WEBSOCKET_FRAME_TOO_LONG *- An endpoint has closed the WebSocket connection due to data received that is too long to be processed.*

WEBSOCKET_EXTENSION_MISSING *- The initiating peer failed to establish the WebSocket connection due to extensions not confirmed by the answering peer.*

WEBSOCKET_REQUEST_UNAVAILABLE *- The answering peer has closed the WebSocket connection due to a condition that prevented it from executing the request received.*

WEBSOCKET_OTHER_ERROR *- Some unspecified error occurred, and the WebSocket connection is closed.*

TLS_CLIENT_CERTIFICATE_ERROR *- The client certificate contains an error that prevents it from being authenticated.*

TLS_SERVER_CERTIFICATE_ERROR *- The server certificate contains an error that prevents it from being authenticated.*

TLS_CLIENT_AUTHENTICATION_FAILED *- Authentication of the client failed.*

TLS_SERVER_AUTHENTICATION_FAILED *- Authentication of the server failed.*

TLS_CLIENT_CERTIFICATE_EXPIRED *- Client certificate validity window does not include current time.*

TLS_SERVER_CERTIFICATE_EXPIRED *- Server certificate validity window does not include current time.*

TLS_CLIENT_CERTIFICATE_REVOKED *- Client certificate is revoked.*

TLS_SERVER_CERTIFICATE_REVOKED *- Server certificate is revoked.*

TLS_OTHER_ERROR *- Some unspecified error occurred in TLS.*

TCP_CONNECT_TIMEOUT *- The TCP connection could not be established due to no response within timeout.*

TCP_CONNECTION_REFUSED *- The TCP connection is not accepted by the peer.*

TCP_CLOSED_BY_LOCAL *- The TCP connection was closed by the local endpoint.*

TCP_CLOSED_OTHER *- The TCP connection was closed due to some unspecified reason.*

TCP_OTHER_ERROR *- Some unspecified error occurred in TCP.*

IP_ADDRESS_NOT_REACHABLE *- The IP address of the peer is not reachable.*

IP_OTHER_ERROR *- Some unspecified error occurred on the IP protocol level.*

135-2016*bj*-7. Interoperability Specification Extensions for BACnet/SC

Rationale

This addendum section defines the PICS modifications for BACnet/SC.

[Annex A PICS Changes for BACnet/SC]

[Change **Annex A**, p. 936]

BACnet Protocol Implementation Conformance Statement

...

BACnet Standardized Device Profiles Supported (Annex L):

☐ **BACnet Cross-Domain Advanced Operator Workstation (B-XAWS)**

…

☐ **BACnet Access Control Credential Reader (B-ACCR)**
☐ ***BACnet Secure Connect Primary Hub (B-SCPH)***
☐ ***BACnet Secure Connect Failover Hub (B-SCFH)***

☐ **BACnet General (B-GENERAL)**

...

***BACnet* Data Link Layer Options:**

...
☐ Point-To-Point, modem, (Clause 10), baud rate(s): _____
☐ *BACnet Secure Connect, Primary Hub (Annex YY)*
Maximum number of simultaneous WebSocket connections accepted: _____

☐ *BACnet Secure Connect, Failover Hub (Annex YY)*
Maximum number of simultaneous WebSocket connections initiated: _____
Maximum number of simultaneous WebSocket connections accepted: _____

☐ *BACnet Secure Connect, Regular Node (Annex YY)*
Maximum number of simultaneous WebSocket connections initiated: _____
Maximum number of simultaneous WebSocket connections accepted: _____

☐ *Transport Layer Security V1.2 supported (RFC 5246)*
Cipher suites supported, using the cipher suite names as of the TLS Cipher Suite Registry at IANA (See RFC 5246):
_____
_____

☐ *Transport Layer Security versions higher than V1.2 supported*

The TLS versions higher than V1.2 supported, including the supported cipher suites for the version, using the cipher suite names as of the TLS Cipher Suite Registry at IANA (See RFC 5246):

_____

_____

☐ *Generates private keys internally, and provides matching certificate signing requests.*

☐ *DNS host name resolution supported (RFC 1123)*
☐ *mDNS host name resolution supported (RFC 6762)*

[Annex K Network Management BIBB Additions for B-SCPH and B-SCFH]

[Add new BIBBs to **Clause K.5**, p. 1076]

K.5.X1 BIBB - Network Management-Secure Connect Primary Hub-B (NM-SCPH-B)

The B device supports at least one BACnet/SC network port and is capable of supporting the BACnet/SC primary hub function on at least one of these ports. The B device is capable of accepting WebSocket connections and performing the hub switch function.

Devices claiming conformance to this BIBB shall meet the minimum requirements for a BACnet device as described by this standard and specifically by Clause 22.

BACnet/SC BVLL Function	Initiate	Execute
Unicast-NPDU	x	x
Original-Broadcast-NPDU	x	x
Forwarded-Broadcast-NPDU	x	x
Address-Resolution		x
Address-Resolution-ACK	x	
Advertisement	x	
Advertisement-Solicitation		x

K.5.X2 BIBB - Network Management-Secure Connect Failover Hub-B (NM-SCFH-B)

The B device supports at least one BACnet/SC network port and is capable of supporting the BACnet/SC failover hub function on at least one of these ports. The B device is capable of accepting WebSocket connections, initiating a WebSocket connection to the primary hub, performing the hub function when the primary hub is inactive, and supporting the failover procedures as the failover hub.

Devices claiming conformance to this BIBB shall meet the minimum requirements for a BACnet device as described by this standard and specifically by Clause 22.

BACnet/SC BVLL Function	Initiate	Execute
Unicast-NPDU	x	x
Original-Broadcast-NPDU	x	x
Forwarded-Broadcast-NPDU	x	x
Address-Resolution		x
Address-Resolution-ACK	x	
Advertisement	x	
Advertisement-Solicitation		x

[Annex L Device Profile Changes for BACnet/SC Hub]

[Change **Annex L**, p.1079]

...

BACnet device profiles are categorized into families:

- Operator Interfaces. This family is composed of B-XAWS, B-AWS, B-OWS, and B-OD.
- Life Safety Operator Interfaces. This family is composed of B-ALSWS, B-LSWS, and B-LSAP.
- Access Control Operator Interfaces. This family is composed of B-XAWS, B-AACWS, B-ACWS, and B-ACSD.
- Controllers. This family is composed of B-BC, B-AAC, B-ASC, B-SA, and B-SS.
- Life Safety Controllers. This family is composed of B-ALSC and B-LSC.
- Access Control Controllers. This family is composed of B-AACC and B-ACC.
- Miscellaneous. This family is composed of B-RTR, B-GW, B-BBMD, B-ACDC, ~~and~~ B-ACCR, *B-SCPH, and B-SCFH*.

[Change **Clause L.7**, p. 1092]

L.7 Miscellaneous Profiles

The following table indicates which BIBBs shall be supported by the device types of this family, for each interoperability area.

Data Sharing

B-RTR	**...**	**B-ACCR**	***B-SCPH***	***B-SCFH***
...			*DS-RP-B*	*DS-RP-B*
			DS-WP-B	*DS-WP-B*

Alarm & Event Management

B-RTR	**...**	**B-ACCR**	***B-SCPH***	***B-SCFH***

Scheduling

B-RTR	**...**	**B-ACCR**	***B-SCPH***	***B-SCFH***

Trending

B-RTR	**...**	**B-ACCR**	***B-SCPH***	***B-SCFH***

Device & Network Management

B-RTR	**...**	**B-ACCR**	***B-SCPH***	***B-SCFH***
...		...	*DM-DDB-B*	*DM-DDB-B*
			DM-DOB-B	*DM-DOB-B*
			DM-DCC-B	*DM-DCC-B*
			NM-SCPH-B	*NM-SCFH-B*

[1] ...

[Add new **Clauses L.7.X1 and L.7.X2, ,** p. 1094]

L.7.X1 BACnet Secure Connect Primary Hub (B-SCPH)

A B-SCPH is a device that is able to perform the BACnet Secure Connect primary hub function as defined in Clause YY.1.1. It accepts secured Web Socket connections.

Data Sharing
• Ability to provide the values of any of its BACnet objects.
• Ability to allow modification of some or all of its BACnet objects by another device.

Alarm and Event Management
• No requirement

Scheduling
• No requirement

Trending
• No requirement

Device and Network Management
• Ability to respond to queries about its status
• Ability to respond to requests for information about any of its objects
• Ability to respond to communication control messages
• Ability to perform the BACnet/SC primary hub function.

L.7.X2 BACnet Secure Connect Failover Hub (B-SCFH)

A B-SCPH is a device that is able to perform the BACnet Secure Connect failover hub function as defined in Annex YY. It accepts secured Web Socket connections when being the active hub, and is able to initiate a WebSocket connection to the primary hub.

Data Sharing
• Ability to provide the values of any of its BACnet objects.
• Ability to allow modification of some or all of its BACnet objects by another device.

Alarm and Event Management
• No requirement

Scheduling
• No requirement

Trending
• No requirement

Device and Network Management
• Ability to respond to queries about its status
• Ability to respond to requests for information about any of its objects
• Ability to respond to communication control messages
• Ability to perform the BACnet/SC failover hub function.

[Add a new entry to **History of Revisions**, p. 1364]

(This History of Revisions is not part of this standard. It is merely informative and does not contain requirements necessary for conformance to the standard.)

HISTORY OF REVISIONS

…	…	**…**
1	X	**Addendum bj to ANSI/ASHRAE 135-2016** Approved by the ASHRAE Standards Committee MONTH X, 20XX; by the ASHRAE Board of Directors MONTH X, 20XX; and by the American National Standards Institute MONTH X, 20XX. 1. Introduce BACnet Secure Connect Datalink Layer Option 2. Introduce BACnet/SC in the Network Layer Specifications 3. Add new Annex YY for the BACnet Secure Connect Datalink Layer Option 4. Extend the Network Port Object Type for BACnet/SC 5. Add and Extend ASN.1 Types for BACnet/SC 6. New Error Codes for BACnet/SC 7. Interoperability Specification Extensions for BACnet/SC

