BSR/ASHRAE Addendum *ce* to
ANSI/ASHRAE Standard 135-2020

Public Review Draft

Proposed Addendum *ce* to Standard 135-2020, BACnet® - A Data Communication Protocol for Building Automation and Control Networks

First Public Review (August 2021)
(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, 180 Technology Parkway Peachtree Corners, GA 30092

[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 135-2020 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.

135-2020*ce*-1. MS/TP Language Replacement

Rationale

To replace culturally offensive terms in the MS/TP language. This must be done while retaining the "MS/TP" moniker as 1) it's commonly used in the industry; and 2) this prevents vendors from needing to replace hardware parts where "MS/TP" may be used. For example, on printed circuit boards or plastic housings.

Changes to ASHRAE Standard 135: (reference: 135-2020)

[Throughout the standard]

Replace "master-slave/token-passing" with "multidrop serial bus/token passing".
Replace "MS/TP slaves" with "MS/TP subordinates".
Replace "master/slave" with "manager/subordinate".
Replace "MS/TP slave" with "MS/TP subordinate".
Replace "Slave Node" with "Subordinate Node".
Replace "Master Node" with "Manager Node".
Replace "master device" with "manager node".
Replace "master devices" with "manager nodes".
Replace "mastership" with "ownership" (Clause 9.3.1)
Replace "masters" with "managers".
Replace "slave node" with "subordinate node".
Replace "slave nodes" with "subordinate nodes".
Replace "slaves" with "subordinates".
Replace "slave" with "subordinate".
Replace "Poll For Master" with "Poll For Manager"
Replace "Reply to Poll For Master" with "Reply to Poll For Manager"
Replace "SoleMaster" with "SoleManager".
Replace "Nmax_master" with "Nmax_manager".
Replace "Max_Master" with "Max_Manager".
Replace "master-slave" with "manager-subordinate".
Replace "max-master" with "max-manager", including ASN.1 BACnetPropertyIdentifier enumeration.
Replace "Slave_Proxy_Enable" with "Subordinate_Proxy_Enable".
Replace "Manual_Slave_Address_Binding" with "Manual_Subordinate_Address_Binding".
Replace "Auto_Slave_Discovery" with "Auto_Subordinate_Discovery".
Replace "Slave_Address_Binding" with "Subordinate_Address_Binding".
Replace "Slave-Proxy" with "subordinate proxy". (no slash in ASN.1)
Replace "Slave Proxy" with "Subordinate Proxy".
Replace "slave device" with "subordinate node".
Replace "slave devices" with "subordinate nodes".
Replace "RESTART_SLAVE_DISCOVERY" with "RESTART_SUBORDINATE_DISCOVERY".
Replace "restart-slave-discovery" with "restart-subordinate-discovery".
Replace "auto-slave-discovery" with "auto-subordinate-discovery".
Replace "manual-slave-address-binding" with "manual-subordinate-address-binding".
Replace "slave-address-binding" with "subordinate-address-binding".
Replace "slave-proxy-enable" with "subordinate-proxy-enable".
Replace "slave-proxy-supported" with "subordinate-proxy-supported".
Replace "slave-discovery-supported" with "subordinate-discovery-supported".
Replace "time master" with "time manager".

[Change clause 3.2, p 199]
…

time distributor: a server device that transmits time via time synchronization services.

…

[Change clause 3.3, p 199]
…

MS/TP ~~master-slave/token-passing~~*multidrop serial bus/token passing*

…

[Change clause 4.1, p. 22]
…

Collectively these options provide a ~~master/slave~~*manager/subordinate* 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.

[Change clause 6.5.3, p 74]
…

The local broadcast MAC address may be used in response messages, although it is discouraged. It is preferable that a device note the SA associated with the original request and reuse that SA in the response. For MS/TP networks, in order for MS/TP ~~master-devices~~*manager nodes* to use the local broadcast MAC address in a response, a Reply Postponed MAC frame shall be sent in response to the BACnet Data Expecting Reply frame and the response may then be sent when the MS/TP ~~master-devices~~*manager nodes* receives the token. MS/TP ~~slave-devices~~*subordinate nodes* are unable to use the local broadcast MAC address for responses because they never receive the token.

[Change clause 9, p. 87]

9 DATA LINK/PHYSICAL LAYERS: ~~MASTER-SLAVE/TOKEN-PASSING~~*MULTIDROP SERIAL BUS/TOKEN PASSING* (MS/TP) LAN

This clause describes a ~~Master-Slave/Token-Passing~~*Multidrop Serial Bus/Token Passing* (MS/TP) data link protocol, which provides the same services to the network layer as ISO 8802-2 Logical Link Control.

…

[Change clause 9.1.4.1, p. 89]
…

This primitive is the service request primitive for releasing a ~~Master-Node~~*Manager Node* State Machine from the ANSWER_DATA_REQUEST state when no reply is available from the higher layers.

…

[Change clause 9.1.4.4, p. 89]
…

Receipt of this primitive causes the MS/TP ~~Master Node~~*Manager Node* State Machine to leave the ANSWER_DATA_REQUEST state. If a ~~Master Node~~*Manager Node* State Machine is not in the ANSWER_DATA_REQUEST state or a ~~Slave Node~~*Subordinate Node* State Machine is present, then this primitive shall be ignored.

…

[Change clause 9.3, p. 100]
The Frame Type is used to distinguish between different types of MAC frames. Defined types are:

00	Token
01	Poll For ~~Master~~*Manager*
02	Reply To Poll For ~~Master~~*Manager*

…

Frame Types 128 through 255 are available to vendors for proprietary (non-BACnet) frames. Use of proprietary frames might allow a Brand-X controller, for example, to send proprietary frames to other Brand-X controllers that do not implement BACnet while using the same medium to send BACnet frames to a Brand-Y panel that does implement BACnet. Token, Poll For ~~Master~~*Manager,* and Reply To Poll For ~~Master~~*Manager* frames shall be understood by all MS/TP ~~master~~*manager* nodes.

The Destination and Source Addresses are one octet each. A Destination Address of 255 (X'FF') denotes broadcast. A Source Address of 255 is not allowed. Addresses 0 to 127 are valid for both ~~master~~*manager* and ~~slave~~*subordinate* nodes. Addresses 128 to 254 are valid only for slave nodes.

MS/TP devices shall support configurable MAC addresses, and each shall be able to be set to any valid unicast address (0..127 for ~~masters~~*managers* and 0..254 for ~~slaves~~*subordinates*). Where a device has multiple MS/TP ports, the MAC address of each port shall be settable to any valid value regardless of the MAC address settings of the other MS/TP ports.

…

9.3.1 Frame Type 00: Token

The Token frame is used to pass network ~~mastership~~*ownership* to the destination node. The use of the Token frame is described in detail in Clause 9.5.

There are no data octets in Token frames.

9.3.2 Frame Type 01: Poll For ~~Master~~*Manager*

The Poll For ~~Master~~*Manager* frame is transmitted by ~~master~~*manager* nodes during configuration and periodically during normal network operation. It is used to discover the presence of other ~~master~~*manager* nodes on the network and to determine a successor node in the token ring. The use of the Poll For ~~Master~~*Manager* frame in the token network is described in detail in Clause 9.5.

There are no data octets in Poll For ~~Master~~*Manager* frames.

Both ~~master~~*manager* and ~~slave~~*subordinate* nodes shall expect to receive Poll For ~~Master~~*Manager* frames. ~~Master~~*Manager* nodes shall respond to Poll For ~~Master~~*Manager* Frames as described in Clause 9.5.6.2. ~~Slave~~*Subordinate* nodes shall ignore Poll For ~~Master~~*Manager* frames, as described in Clause 9.5.7.2.

9.3.3 Frame Type 02: Reply To Poll For ~~Master~~*Manager*

This frame is transmitted as a reply to the Poll For ~~Master~~*Manager* frame. It is used to indicate that the node sending the frame wishes to enter the token ring. The use of this frame in the token network is described in detail in Clause 9.5.

There are no data octets in Reply To Poll For ~~Master~~*Manager* frames.

…

9.3.6 Frame Type 05: BACnet Data Expecting Reply

This frame is used by ~~master~~*manager* nodes to convey the data parameter of a DL_UNITDATA.request whose DER parameter is TRUE. The length of the data portion of a BACnet Data Expecting Reply frame may range from 0 to 501 octets.

9.3.7 Frame Type 06: BACnet Data Not Expecting Reply

This frame is used to convey the data parameter of a DL_UNITDATA.request whose DER parameter is FALSE. The length of the data portion of a BACnet Data Not Expecting Reply frame may range from 0 to 501 octets.

9.3.8 Frame Type 07: Reply Postponed

This frame is used by ~~master~~*manager* nodes to defer sending a reply to a previously received BACnet Data Expecting Reply frame. The use of this frame in the token network is described in detail in Clause 9.5.6.

There are no data octets in Reply Postponed frames.

9.3.9 Frame Type 32: BACnet Extended Data Expecting Reply

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

9.3.9 Frame Type 33: BACnet Extended Data Not Expecting Reply

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

[Change clause 9.4, p. 102]

…

MS/TP uses a token to control access to a bus network. A ~~master~~*manager* node may initiate the transmission of a data frame when it holds the token. Both ~~master~~*manager* and ~~slaves~~*ubordinate* nodes may transmit data frames in response to requests from master nodes. After sending at most $\mathbf{N_{max\_info\_frames}}$ data frames (and awaiting any expected replies), a ~~master~~*manager* node shall pass the token to the next ~~master~~*manager* node.

…

Most token bus networks, such as ARCNET, do not distinguish between requests and replies: both are passed in the same type of frames, which are sent only when the sending node has the token. Since MS/TP defines ~~slave nodes~~*ubordinate nodes* that never hold the token, a means must be provided to allow replies to be returned from slave devices. For simplicity, the same mechanism is used for replies returned from ~~master~~*manager* nodes.

When a request that expects a reply is sent to an MS/TP node, the sender shall wait for the reply to be returned before passing the token. If the responding node is a ~~master~~*manager*, it may return the reply or it may return a Reply Postponed frame, indicating that the actual reply will be returned later, when the replying node holds the token.

[Change clause 9.5.2, p. 103]

…

PS "Poll Station," the MAC address of the node to which This Station last sent a Poll For ~~Master~~*Manager*. This is used during token maintenance.

…

RetryCount A counter of transmission retries used for Token and Poll For ~~Master~~*Manager* transmission.

…

~~SoleMaster~~*SoleManager* A Boolean flag set to TRUE by the ~~master~~*manager node state* machine if this node is the only known ~~master~~*manager* node.

…

TokenCount The number of tokens received by this node. When this counter reaches the value $\mathbf{N_{poll}}$, the node polls the address range between **TS** and **NS** for additional ~~master~~*manager* nodes. **TokenCount** is set to one at the end of the polling process.

[Change clause 9.5.3, p. 104]

…

$\mathbf{N_{max\_}}$~~**master**~~***manager*** This parameter represents the value of the Max_~~Master~~*Manager* property of the node's Network Port object which represents this MS/TP port. The value of Max_~~Master~~*Manager* specifies the highest allowable address for ~~master~~*manager* nodes. The value of this parameter shall be less than or equal to 127.

$\mathbf{N_{poll}}$ The number of tokens received or used before a Poll For ~~Master~~*Manager* cycle is executed: 50.

…

$\mathbf{T_{usage\_delay}}$ The maximum time a node may wait after reception of the token or a Poll For ~~Master~~*Manager* frame before sending the first octet of a frame: 15 milliseconds.

$\mathbf{T_{usage\_timeout}}$ The time without a DataAvailable or ReceiveError event that a sending node must wait for a receiving node to begin using a token or replying to a Poll For ~~Master~~*Manager* frame: 20 milliseconds. Sending node implementations may use larger values for this timeout, not to exceed 35 milliseconds.

[Change Clause 9.5.4, p. 106]

…

The Receive Frame state machine operates independently from the MS/TP ~~Master~~*Manager* Node or ~~Slave~~*Subordinate* Node machine, communicating with it by means of flags and other variables. The description assumes that the ~~Master~~*Manager* Node or ~~Slave~~*Subordinate* Node state machine can process received frames and other indications from the Receive Frame state machine before the next frame begins. The means by which this behavior is implemented are a local matter.

[Change Clause 9.5.6, p. 113]

9.5.6 ~~Master~~*Manager* Node Finite State Machine

The description of operation is as a finite state machine. Figure 9-4 shows the ~~Master~~*Manager* Node state machine, which is described fully in this clause. Each state is given a name, specified in all capital letters. Transitions are also named, in mixed upper- and lowercase letters. Transitions are described as a series of conditions followed by a series of actions to be taken if the conditions are met. The final action in each transition is entry into a new state, which may be the same as the current state.

A ~~master~~*manager* node that supports segmentation shall respond to each BACnet Data Expecting Reply frame with either a Reply Postponed frame or a data frame. This response releases the MS/TP ~~Master~~*Manager* Node State Machine that is holding the token from the WAIT_FOR_REPLY state and allows it to send the next MS/TP frame.

[Replace Figure 9-4, p. 113]

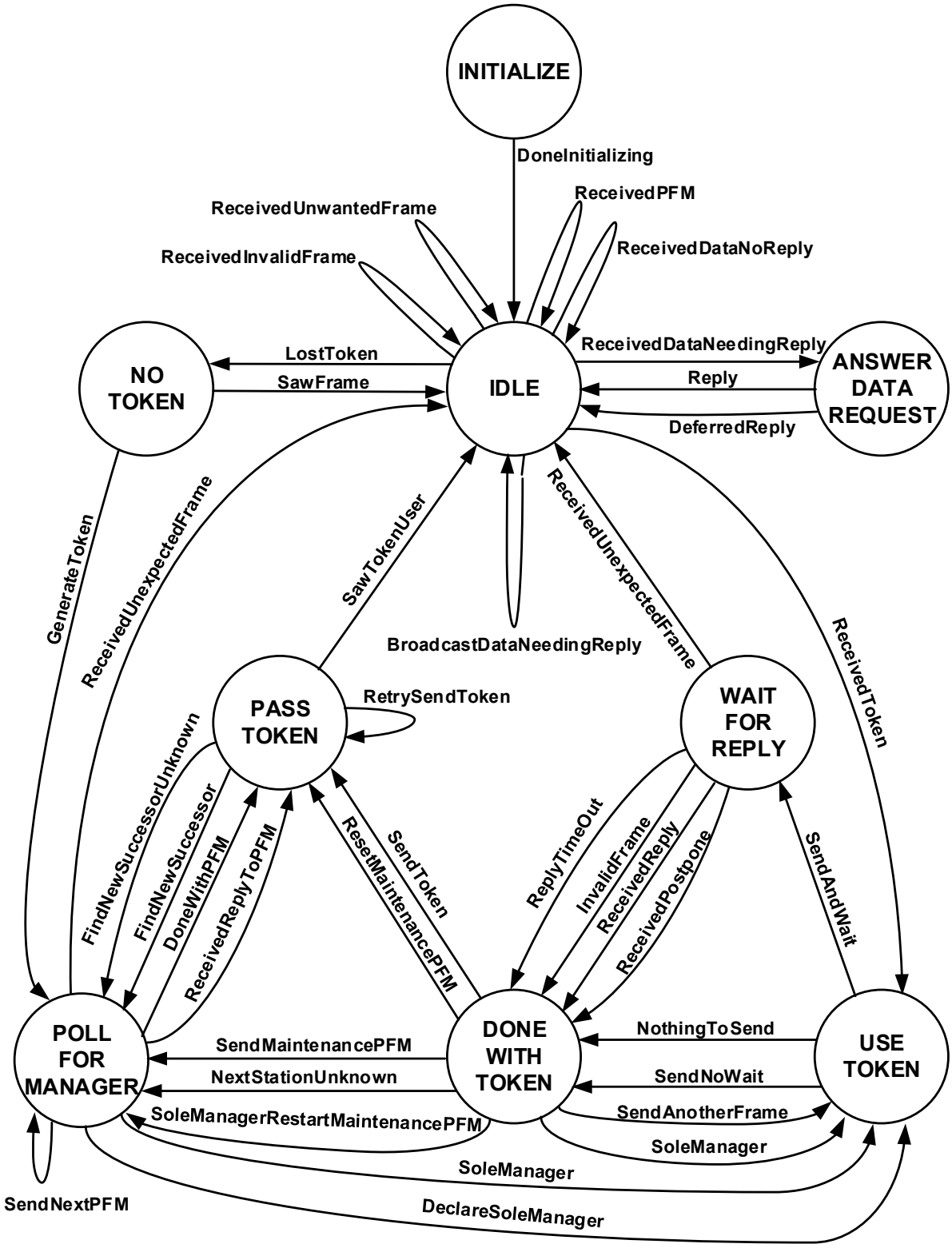

Figure 9-4. ~~Master~~*Manager* Node State Machine.

[Change clause 9.5.6.1, p. 114]

When a ~~master~~*manager* node is powered up or reset, it shall unconditionally enter the INITIALIZE state.

DoneInitializing
Unconditionally,

set TS to the node's station address, set NS equal to TS (indicating that the next station is unknown), set PS equal to TS, set TokenCount to $\mathbf{N_{poll}}$ (thus causing a Poll For ~~Master~~*Manager* to be sent when this node first receives the token), set ~~SoleMaster~~*SoleManager* to FALSE, set ReceivedValidFrame and ReceivedInvalidFrame to FALSE, and enter the IDLE state.

[Change clause 9.5.6.2, p. 114]
…
(d) DestinationAddress is equal to TS and FrameType is equal to Reply To Poll For ~~Master~~*Manager* or Reply Postponed, then an unexpected or unwanted frame was received. Set ReceivedValidFrame to FALSE, and enter the IDLE state to wait for the next frame.

ReceivedToken
If ReceivedValidFrame is TRUE and DestinationAddress is equal to TS (this station) and FrameType is equal to Token,

then set ReceivedValidFrame to FALSE; set FrameCount to zero; set ~~SoleMaster~~*SoleManager* to FALSE; and enter the USE_TOKEN state.

ReceivedPFM
If ReceivedValidFrame is TRUE and DestinationAddress is equal to TS (this station) and FrameType is equal to Poll For ~~Master~~*Manager*,

then call SendFrame to transmit a Reply To Poll For ~~Master~~*Manager* frame to the node whose address is specified by SourceAddress (Source Address of the Poll); set ReceivedValidFrame to FALSE; and enter the IDLE state.
…

[Change clause 9.5.2, p. 103]
…
PS "Poll Station," the MAC address of the node to which This Station last sent a Poll For ~~Master~~*Manager*. This is used during token maintenance.

…

RetryCount A counter of transmission retries used for Token and Poll For ~~Master~~*Manager* transmission.

SilenceTimer A timer with nominal 5 millisecond resolution used to measure and generate silence on the medium between octets. It is incremented by a timer process and is cleared by the Receive State Machine when activity is detected and by the SendFrame procedure as each octet is transmitted. Since the timer resolution is limited and the timer is not necessarily synchronized to other machine events, a timer value of N will actually denote intervals between N-1 and N.

Sole~~Master~~*Manager* A Boolean flag set to TRUE by the master machine if this node is the only known ~~master~~*manager* node.

SourceAddress Used to store the Source Address of a received frame.

TokenCount The number of tokens received by this node. When this counter reaches the value $\mathbf{N_{poll}}$, the node polls the address range between **TS** and **NS** for additional ~~master~~*manager* nodes. **TokenCount** is set to one at the end of the polling process.

[Change clause 9.5.6.5, p. 116]
The DONE_WITH_TOKEN state either sends another data frame, passes the token, or initiates a Poll For ~~Master~~*Manager* cycle.

SendAnotherFrame
If FrameCount is less than $\mathbf{N_{max\_info\_frames}}$,

then this node may send another information frame before passing the token. Enter the USE_TOKEN state.

NextStationUnknown
If FrameCount is greater than or equal to $\mathbf{N_{max\_info\_frames}}$, ~~SoleMaster~~*SoleManager* is FALSE and NS is equal to TS,

then the next station to which the token should be sent is unknown. Set PS to (TS+1) modulo ($\mathbf{N_{max\_}}$~~$\mathbf{master}$~~$\mathbf{\textit{manager}}$+1); call SendFrame to transmit a Poll For ~~Master~~*Manager* frame to PS; set RetryCount to zero; and enter the POLL_FOR_~~MASTER~~*MANAGER* state.

~~SoleMaster~~*SoleManager*
If FrameCount is greater than or equal to $\mathbf{N_{max\_info\_frames}}$ and TokenCount is less than $\mathbf{N_{poll}}$ and ~~SoleMaster~~*SoleManager* is TRUE,

then there are no other known ~~master~~*manager* nodes to which the token may be sent (true ~~master~~*manager*-~~slave~~*subordinate* operation). Set FrameCount to zero, increment TokenCount, and enter the USE_TOKEN state.

SendToken
If FrameCount is greater than or equal to $\mathbf{N_{max\_info\_frames}}$ and TokenCount is less than $\mathbf{N_{poll}}$ and SoleMaster is FALSE, or if NS is equal to (TS+1) modulo ($\mathbf{N_{max\_}}$~~$\mathbf{master}$~~$\mathbf{\textit{manager}}$+1),

then increment TokenCount; call SendFrame to transmit a Token frame to NS; set RetryCount and EventCount to zero; and enter the PASS_TOKEN state. (The comparison of NS and TS+1 eliminates the Poll For ~~Master~~*Manager* if there are no addresses between TS and NS, since there is no address at which a new ~~master~~*manager* node may be found in that case).

SendMaintenancePFM
If FrameCount is greater than or equal to $\mathbf{N_{max\_info\_frames}}$ and TokenCount is greater than or equal to $\mathbf{N_{poll}}$ and (PS+1) modulo ($\mathbf{N_{max\_}}$~~$\mathbf{master}$~~$\mathbf{\textit{manager}}$+1) is not equal to NS,

then set PS to (PS+1) modulo ($\mathbf{N_{max\_}}$~~$\mathbf{master}$~~$\mathbf{\textit{manager}}$+1); call SendFrame to transmit a Poll For ~~Master~~*Manager* frame to PS; set RetryCount to zero; and enter the POLL_FOR_~~MASTER~~*MANAGER* state.

ResetMaintenancePFM
If FrameCount is greater than or equal to $\mathbf{N_{max\_info\_frames}}$ and TokenCount is greater than or equal to $\mathbf{N_{poll}}$ and (PS+1) modulo ($\mathbf{N_{max\_}}$~~$\mathbf{master}$~~$\mathbf{\textit{manager}}$+1) is equal to NS, and ~~SoleMaster~~*SoleManager* is FALSE,

then set PS to TS; call SendFrame to transmit a Token frame to NS; set RetryCount and EventCount to zero; set TokenCount to one; and enter the PASS_TOKEN state.

~~SoleMasterRestartMaintenancePFM~~*SoleManagerRestartMaintenancePFM*
If FrameCount is greater than or equal to $\mathbf{N_{max\_info\_frames}}$, and TokenCount is greater than or equal to $\mathbf{N_{poll}}$, and (PS+1) modulo ($\mathbf{N_{max\_}}$~~$\mathbf{master}$~~$\mathbf{\textit{manager}}$+1) is equal to NS, and ~~SoleMaster~~*SoleManager* is TRUE,

then set PS to (NS +1) modulo ($\mathbf{N}_{max\_~~master~~manager}$+1) ; call SendFrame to transmit a Poll For ~~Master~~*Manager* to PS; set NS to TS (no known successor node); set RetryCount to zero; set TokenCount to one; and enter the POLL_FOR_~~MASTER~~*MANAGER* state to find a new successor to TS.

[Change clause 9.5.6.6, p. 117]

…

FindNewSuccessorUnknown

If SilenceTimer is greater than or equal to $\mathbf{T}_{usage\_timeout}$ and RetryCount is greater than or equal to $\mathbf{N}_{retry\_token}$, and (NS+1) modulo ($\mathbf{N}_{max\_~~master~~manager}$+1) is equal to TS,

then assume that NS has failed. Set PS to (TS+1) modulo ($\mathbf{N}_{max\_~~master~~manager}$+1); call SendFrame to transmit a Poll For ~~Master~~*Manager* frame to PS; set NS to TS (no known successor node); set RetryCount and TokenCount to zero; and enter the POLL_FOR_~~MASTER~~*MANAGER* state to find a new successor to TS.

FindNewSuccessor

If SilenceTimer is greater than or equal to $\mathbf{T}_{usage\_timeout}$ and RetryCount is greater than or equal to $\mathbf{N}_{retry\_token}$,

then assume that NS has failed. Set PS to (NS+1) modulo ($\mathbf{N}_{max\_~~master~~manager}$+1); call SendFrame to transmit a Poll For ~~Master~~*Manager* frame to PS; set NS to TS (no known successor node); set RetryCount and TokenCount to zero; and enter the POLL_FOR_~~MASTER~~*MANAGER* state to find a new successor to TS.

[Change clause 9.5.6.7, p. 117]

…

GenerateToken

If SilenceTimer is greater than or equal to $\mathbf{T}_{no\_token}$+($\mathbf{T}_{slot}$*TS) and SilenceTimer is less than $\mathbf{T}_{no\_token}$+($\mathbf{T}_{slot}$*(TS+1)),

then assume that this node is the lowest numerical address on the network and is empowered to create a token. Set PS to (TS+1) modulo ($\mathbf{N}_{max\_~~master~~manager}$+1); call SendFrame to transmit a Poll For ~~Master~~*Manager* frame to PS; set NS to TS (indicating that the next station is unknown); set RetryCount and TokenCount to zero; and enter the POLL_FOR_~~MASTER~~*MANAGER* state to find a new successor to TS.

[Change clause 9.5.6.8, p. 118]

9.5.6.8 POLL_FOR_~~MASTER~~*MANAGER*

In the POLL_FOR_~~MASTER~~*MANAGER* state, the node listens for a reply to a previously sent Poll For ~~Master~~*Manager* frame in order to find a successor node.

ReceivedReplyToPFM

If ReceivedValidFrame is TRUE and DestinationAddress is equal to TS (this station) and FrameType is equal to Reply To Poll For ~~Master~~*Manager*,

then set ~~SoleMaster~~*SoleManager* to FALSE; set NS equal to SourceAddress; set EventCount to zero; call SendFrame to transmit a Token frame to NS; set PS to the value of TS; set TokenCount and RetryCount to zero; set ReceivedValidFrame to FALSE; and enter the PASS_TOKEN state.

ReceivedUnexpectedFrame

If ReceivedValidFrame is TRUE and either

(a) DestinationAddress is not equal to TS or
(b) FrameType is not equal to Reply To Poll For ~~Master~~*Manager*,

then an unexpected frame was received. This may indicate the presence of multiple tokens. Set ReceivedValidFrame to FALSE and enter the IDLE state to synchronize with the network. This action drops the token.

~~SoleMaster~~*SoleManager*
If ~~SoleMaster~~*SoleManager* is TRUE and either

(a) SilenceTimer is greater than or equal to $\mathbf{T}_{\text{usage\_timeout}}$ or
(b) ReceivedInvalidFrame is TRUE,

then there was no valid reply to the periodic poll by the sole known ~~master~~*manager* for other ~~masters~~*managers*. Set FrameCount to zero, set ReceivedInvalidFrame to FALSE, and enter the USE_TOKEN state.

DoneWithPFM
If ~~SoleMaster~~*SoleManager* is FALSE and NS is not equal to TS and either:

(a) SilenceTimer is greater than or equal to $\mathbf{T}_{\text{usage\_timeout}}$ or
(b) ReceivedInvalidFrame is TRUE,

then there was no valid reply to the maintenance poll for a ~~master~~*manager* at address PS. Set EventCount to zero; call SendFrame to transmit a Token frame to NS; set RetryCount to zero; set ReceivedInvalidFrame to FALSE; and enter the PASS_TOKEN state.

SendNextPFM
If ~~SoleMaster~~*SoleManager* is FALSE and NS is equal to TS (no known successor node) and (PS+1) modulo ($\mathbf{N}_{\text{max\_}}$~~master~~*manager*+1) is not equal to TS and either:

(a) SilenceTimer greater than or equal to $\mathbf{T}_{\text{usage\_timeout}}$ or
(b) ReceivedInvalidFrame is TRUE,

then set PS to (PS+1) modulo ($\mathbf{N}_{\text{max\_}}$~~master~~*manager*+1); call SendFrame to transmit a Poll For ~~Master~~*Manager* frame to PS; set RetryCount to zero; set ReceivedInvalidFrame to FALSE; and re-enter the current state.

Declare ~~SoleMaster~~*SoleManager*
If ~~SoleMaster~~*SoleManager* is FALSE and NS is equal to TS (no known successor node) and (PS+1) modulo ($\mathbf{N}_{\text{max\_}}$~~master~~*manager*+1) is equal to TS and either

(a) SilenceTimer is greater than or equal to $\mathbf{T}_{\text{usage\_timeout}}$ or
(b) ReceivedInvalidFrame is TRUE,

then set SoleMaster TRUE to indicate that this station is the only master; set FrameCount to zero; set ReceivedInvalidFrame to FALSE; and enter the USE_TOKEN state.

[Change clause 9.5.7, p. 119]

9.5.7 ~~Slave~~*Subordinate* Node Finite State Machine

The state machine for a slave node is similar to, but considerably simpler than, that for a ~~master~~*manager* node. A ~~slave~~*subordinate* node shall neither transmit nor receive segmented messages. If a ~~slave~~*subordinate* node receives a segmented BACnet-Confirmed-Request-PDU, the node shall respond with a BACnet-Abort-PDU specifying abort-reason "segmentation not supported." Figure 9-5 shows the ~~Slave~~*Subordinate* Node state machine, which is described fully in the following text.

[Change caption 9-5, p 120]

Figure 9-5. ~~Slave~~*Subordinate* Node State Machine

[Change clause 9.5.7.1, p. 120]
When a ~~slave~~*subordinate* node is powered up or reset, it shall unconditionally enter the INITIALIZE state.

[Change clause 9.5.7.2, p. 120]
…
(c) FrameType has a value of Token, Poll For ~~Master~~*Manager*, Reply To Poll For ~~Master~~*Manager*, Reply Postponed, or a FrameType not known to this node,
…

[Change clause 9.11, p. 128]
…
(a) ~~master or slave~~*manager or subordinate* implementation

[Change clause 9.11.1, p. 128]

9.11.1 ~~Master or Slave~~*Manager or Subordinate* Implementation

The PICS shall indicate whether the device implements a ~~Master~~*Manager* Node State Machine or ~~Slave~~*Subordinate* Node State Machine or either.

[Change Annex A]
…
☐ MS/TP ~~master~~ (Clause 9)
☐ ~~Master~~*Manager Node* ☐ ~~Slave~~*Subordinate Node*

[Change clause 12.11, p 221]
…

Max_~~Master~~*Manager*	Unsigned(0..127)	O[6]

…
[6] These properties are required if the device is an MS/TP ~~master~~*manager* node.

[Change clause 12.11.32, p 226]

12.11.32 Max_~~Master~~*Manager*

This property, of type Unsigned, shall be present if the device is a ~~master~~*manager* node on an MS/TP network. The value of this property shall reflect the value of the Max_~~Master~~*Manager* property of the Network Port object with the lowest object instance whose Network_Type is MSTP and whose Out_Of_Service is FALSE. See Clause **Error! Reference source not found.**~~12.56.51~~.
…

[Change clause 12.11.33, p 226]

12.11.33 Max_Info_Frames

This property, of type Unsigned, shall be present if the device is a ~~master~~*manager* node on an MS/TP network. The value of this property shall reflect the value of the Max_Info_Frames property of the Network Port object with the lowest object instance whose Network_Type is MSTP and whose Out_Of_Service is FALSE. See Clause **Error! Reference source not found.**~~12.56.52~~.

[Change clause 12.28, p. 341]
…
While the Load Control object is designed to allow independent operation, it is possible that there will exist within a building (or even within a device) a hierarchy of Load Control objects, where one Load Control object receives a load shed command, possibly from a non-BACnet client (e.g., a utility), and the controller which hosts that object (the ~~master~~*manager*) in turn will be responsible for managing and issuing requests to other Load Control objects. There may be a negotiation between the ~~master~~*manager* and its subordinate Load Control objects. The ~~master~~*manager* uses WriteProperty or WritePropertyMultiple to set shed request parameters in the subordinate Load Control objects. A subordinate Load Control object would then set its Expected_Shed_Level property to the value that it expects to be able to achieve after Start_Time. Before Start_Time, the ~~master~~*manager* object can read the Expected_Shed_Level properties of its subordinates to determine expected compliance with the request. After Start_Time plus Duty_Window, the Actual_Shed_Level properties of the subordinate objects will reflect the actual amount shed in the past Duty_Window. If by reading these properties the ~~master~~*manager* Load Control object determines that one or more subordinate objects cannot completely comply with the request, the ~~master~~*manager* may choose to modify the shed requests to subordinates, such that the overall shed target is achieved. For instance, it may request that another object shed a greater amount of its load or it may choose to request that the noncompliant device shed a greater amount. This negotiation could be repeated at each successive level in the hierarchy. If the subordinate Load Control objects also support intrinsic reporting, expected or actual instances of non-compliance can be reported to the ~~master~~*manager* object using event notifications.

Where large loads are concerned, it is expected that the ~~master~~*manager* Load Control object will employ sequencing to distribute the startup and shutdown of managed loads. When the load control ~~master~~*manager* is used in a gateway to a non-BACnet load control client, such as a utility company, the gateway shall accept and process any start randomization commands and accordingly distribute the initiation of load control requests to its subordinate Load Control objects.
…

[Change clause 12.28.10, p. 346]
…
The Load Control object's available shed actions are described by the Shed_Level_Descriptions array and are mapped to the BACnet visible values of Requested_Shed_Level by the Shed_Levels array. The SHED_INACTIVE state shall always be represented by the value 0, which is not represented in the Shed_Levels or Shed_Level_Descriptions arrays. If Requested_Shed_Level choice is AMOUNT, the value of Requested_Shed_Level shall be interpreted as an amount, in kilowatts, by which to reduce power usage. Load Control objects are required to support the LEVEL choice. Support for the PERCENT and AMOUNT choices is optional. This allows a ~~master~~*manager* to be guaranteed the ability to write to the Load Control object by using the LEVEL choice.

[Change clause 12.28.26, p. 347]
…
This property, of type BACnetShedLevel, indicates the amount of power that the object expects to be able to shed in response to a load shed request. When the object is in the SHED_INACTIVE state, this value shall be equal to the default value of Requested_Shed_Level. When a shed request is pending or active, Expected_Shed_Level shall be equal to the shed level the object expects to be able to achieve at Start_Time. Expected_Shed_Level allows a client (e.g., a ~~master~~*manager*-level Load Control object) to determine if a pending shed request needs to be modified in order to achieve the requested shed level, in the event that Expected_Shed_Level is less than the Requested_Shed_Level. The units for Expected_Shed_Level are the same as the units for Requested_Shed_Level.

[Change clause 12.56, p. 242]
NOTE TO REVEIWER: This clause needs updating once changes to 12.56 made in 135-2020 Addendum CC is published.
[Change clause 21.6, p. 930]

…
auto-~~slave~~*subordinate*-discovery (169),
…
manual-slave*subordinate* -address-binding (170),
…
max-~~master~~*manager* (64),
…
~~slave~~*subordinate*-address-binding (171),
~~slave~~*subordinate*-proxy-enable (172),
…
-- see max-~~master~~*manager* (64),
…
-- see auto-~~slave~~*subordinate*-discovery (169),
-- see manual-~~slave~~*subordinate* -address-binding (170),
-- see ~~slave~~*subordinate*-address-binding (171),
-- see ~~slave~~*subordinate*-proxy-enable (172),

[Change clause 25, p 961]
IETF RFC 8163 (2017), Transmission of IPv6 over ~~Master-Slave/Token-Passing (~~MS/TP~~)~~ Networks, Internet Engineering Task Force

[Change Annex K]
…
K.5.31 BIBB - Device Management-~~Slave Proxy~~*Subordinate Proxy*-View and Modify-A (DM-SP-VM-A)

The A device displays and modifies the ~~slave~~*subordinate node* proxy related properties in a ~~slave~~*subordinate node* proxy device.

BACnet Service	Initiate	Execute
ReadProperty	x	
ReadRange	x	
AddListElement	x	
RemoveListElement	x	
WriteProperty	x	

Devices claiming conformance to DM-SP-VM-A shall be able to read and present the ~~Slave_Proxy_Enable, Manual_Slave_Address_Binding, Auto_Slave_Discovery and the Slave_Address_Binding~~ *Subordinate_Proxy_Enable, Manual_Subordinate_Address_Binding, Auto_Subordinate_Discovery and the Subordinate_Proxy_Address_Binding* properties of Device and Network Port object types (support for interaction with Network Port object types is only required for devices claiming conformance to a Protocol_Revision of 17 or greater.)

When a device is proxying for a large number of MS/TP ~~slave devices~~*subordinate nodes*, the ~~Manual_Slave_Address_Binding~~*Manual_Subordinate_Address_Binding* and ~~Slave_Address_Binding~~*Subordinate_Address_Binding* properties can contain very large values. For this reason, the A device shall support initiation of the ReadRange service so as to be able to retrieve large values for these properties.

The A device shall be capable of configuring the ~~Slave_Proxy_Enable, Manual_Slave_Address_Binding, Auto_Slave_Discovery, and Slave_Address_Binding~~*Subordinate_Proxy_Enable, Manual_Subordinate_Address_Binding, Subordinate_Auto_Discovery and the Subordinate_Address_Binding* properties.

A device claiming support for DM-SP-VM-A is interoperable with devices that support DM-SP-B.

K.5.32 BIBB - Device Management ~~Slave Proxy~~*Subordinate Proxy*-B (DM-SP-B)

The B device implements the ~~slave~~*subordinate node* proxy functionality and provides I-Am messages on behalf of MS/TP ~~slave devices~~*subordinate nodes*.

BACnet Service	Initiate	Execute
ReadProperty	x	x
ReadRange		x
AddListElement		x
RemoveListElement		x
WriteProperty		x
Who-Is		x
I-Am	x	

Devices claiming conformance to DM-~~SP~~*MP*-B shall support the ~~Slave_Proxy_Enable, Manual_Slave_Address_Binding, Auto_Slave_Discovery and the Slave_Address_Binding~~ *Subordinate_Proxy_Enable, Manual_Subordinate_Address_Binding, Auto_Subordinate_Discovery and the Subordinate_Proxy_Address_Binding* properties. ~~Slave~~*Subordinate* proxies shall be capable of performing the ~~slave proxy~~*subordinate node* proxy function on any or all of the directly connected MS/TP networks.

Devices claiming conformance to this BIBB shall be capable of proxying for at least 32 MS/TP ~~slaves devices~~*subordinate nodes* per *the* directly connected MS/TP network.

When the B device is proxying for a large number of MS/TP ~~slave devices~~*subordinate nodes*, the ~~Manual_Slave_Address_Binding and Slave_Address_Binding~~ *Manual_Subordinate_Address_Binding, and the Subordinate_Address_Binding* properties can contain very large values. For this reason, the B device shall support execution of the ReadRange service.

A device claiming support for DM-SP-B is interoperable with devices that support DM-SP-VM-A and with MS/TP ~~slave devices~~*subordinate nodes*.

[Change clause Y.21.3, p 1361]

…

"time-~~master~~*distributor*"	If the device is a Time ~~Master~~*Distributor*

…

135-2020*ce*-2. CSML Name Aliasing

Rationale

To allow previously published data items (properties, etc.), to be revised to have better or clearer names while maintaining compatibility with existing data representations, the Abstract Data Model defined in Annex Y is extended to allow name aliases that can be used as alternate names for the data item. Aliased names are protocol revision independent and are thus significantly different from the “Data Revisions” mechanism described in Clause Y.19.

[Change **Clause Y.4.1**, p. 1316]

Y.4.1 'name'
This metadata, of type String, provides a name for the data. All data in the data model has a name, and the name shall be unique among the data's siblings. However, a name may or may not be required in serializations, based on the data's context. For example, a name is required for definitions, and for Sequence, Choice, Composition, Collection, and Object members, but not for Array, List, and SequenceOf members, where missing names will be provided by the consumer of the serialization. *See Clause Y.4.1.X for a description of name aliasing.*

[Add new clauses under **Clause Y.4.1**, p. 1316]

Y.4.1.X Alternate Names
A data item can be defined to have alternate names for compatibility with older data sets when the name of the data item has changed and the previous names are still allowed. This is independent of the data revisioning mechanism defined in Clause Y.19 and any ‘dataRev’ metadata for the representation. A list of allowable alternate names can be specified with the ‘aliases’ metadata item.

Y.4.1.X.1 'aliases'
This metadata, of type StringSet, provides a set of alternate names for the data. To allow unambiguous data representations, the requirement that all names shall be unique among siblings shall not be violated by the use of aliases. Any of the strings in the ‘aliases’ set shall be accepted as the name of the data item when consuming a representation; however, the aliases are generally considered to be deprecated and the ‘name’ should be used in newly generated representations, unless the generator is aware that the consumer requires a specific alias for compatibility. For example, in request/response scenarios, the generator might infer this requirement from the requestor’s use of the alias.

135-2020*ce*-3. Remove writableWhen and requiredWhen

Rationale

The 'writableWhen' and 'requiredWhen' metadata items were designed to capture the requirements made by footnotes in Clause 12 tables in a machine-readable way. However, they still required manually coded interpretations since they were only an enumeration rather than some kind of machine evaluable expression. Rather than continuously updating the enumeration as the standard evolves, this feature is being removed.

[Change subclauses of **Clause Y.21**, p. 1360]

Y.21 BACnet-Specific Metadata

Data that is used to model data from BACnet sources have an additional set of available metadata that is specific to BACnet objects and services.

Y.21.1 'writableWhen'

This optional metadata, of type Enumerated, *has been deprecated. If present, it shall be ignored.* ~~indicates the conditions under which the data value may be writable. The choices for the value and their meanings are defined in the following table. The choices for the value and their meanings are defined in the following table.~~

~~**Table Y-6.** Values for Metadata 'writableWhen'~~

~~Metadata Value~~	~~Meaning~~
~~"out-of-service"~~	~~When Out_Of_Service property is TRUE~~
~~"commandable"~~	~~When this property is commandable~~
~~"other"~~	~~Non-standard requirement. Descriptive text should be provided by 'writableWhenText' metadata.~~

~~If the 'writableWhenText' metadata is present then the implied value for this metadata is "other". Therefore, the 'writableWhenText' metadata may be present without requiring the presence of the 'writableWhen' metadata. However, if a value for the 'writableWhen' metadata is specified and is not equal to "other", then the 'writableWhenText' metadata is not inherited from a definition and the 'writableWhenText' metadata shall not be specified in the same context.~~

~~When this metadata is equal to "other", the optional 'writableWhenText' metadata can be used to provide display text to describe the nonstandard condition.~~

~~A common case in Clause 12 objects is the requirement that Present_Value be writable when Out_Of_Service is true.~~

~~<Definitions>~~
~~ <Object name="0-AnalogInputObject">~~
~~   …~~
~~   <Real name="present-value" writableWhen="out-of-service" … />~~
~~   …~~
~~ </Object>~~
~~</Definitions>~~

Y.21.2 'writableWhenText'

This optional localizable metadata, of type String, is used to provide display text for the writability condition ~~when the 'writableWhen' metadata has the value of "other". While the 'writableWhen' metadata is an enumeration of fixed strings as defined by this standard, the~~ *The* 'writableWhenText' metadata can contain arbitrary text that shall consist of plain printable characters with no formatting markup or line breaks.

For example, ~~if the writability condition is not one of the standard conditions, then the 'writableWhen' metadata has the value of "other" and the members of~~ the 'writableWhenText' *metadata* (encoded in XML as <WritableWhenText>

elements) *can be used to* provide ~~the~~ display text *for the writability condition* (the default locale in this example is "en").

<Unsigned name="trend-memory-allocation" ~~writableWhen="other"~~ >
<WritableWhenText>The Device object's Device Status property is "download required"</WritableWhenText>
</Unsigned>

~~If a 'writableWhenText' metadata is present in a context without the 'writableWhen' metadata, the 'writableWhen' metadata is implicitly assigned the value "other".~~

~~For example, the following shows that it is not necessary to include writableWhen="other" in a context with a 'writableWhenText' string (encoded in XML as a <WritableWhenText> element).~~

~~<Unsigned name="aux-input" >~~
~~<WritableWhenText>The "Aux Disable" property is TRUE</WritableWhenText>~~
~~</Unsigned>~~

Y.21.3 'requiredWhen'

This optional metadata, of type Enumerated, *has been deprecated. If present, it shall be ignored.* ~~indicates the conditions under which optional data shall be present. The choices for the value and their meanings are defined in Table Y-7.~~

~~Table Y-7.~~ ~~Values for Metadata 'requiredWhen'~~

~~Metadata Value~~	~~Meaning~~
~~"intrinsic-supported"~~	~~If the object supports intrinsic reporting~~
~~"cov-notify-supported"~~	~~If the object supports COV reporting~~
~~"cov-subscribe-supported"~~	~~If the device supports execution of either the SubscribeCOV or SubscribeCOVProperty service~~
~~"present-value-commandable"~~	~~If Present_Value is commandable~~
~~"segmentation-supported"~~	~~If Segmentation of any kind is supported~~
~~"virtual-terminal-supported"~~	~~If Virtual Terminal services are supported~~
~~"time-sync-execution"~~	~~If the device supports the execution of the TimeSynchronization service~~
~~"utc-time-sync-execution"~~	~~If the device supports the execution of the UTCTimeSynchronization service~~
~~"time-master"~~	~~If the device is a Time Master~~
~~"backup-restore-supported"~~	~~If the device supports the backup and restore procedures~~
~~"slave-proxy-supported"~~	~~If the device is capable of being a Slave Proxy device~~
~~"slave-discovery-supported"~~	~~If the device is capable of being a Slave Proxy device that implements automatic discovery of slaves~~
~~"other"~~	~~Non-standard requirement. Descriptive text should be provided by requiredWhenText metadata.~~

~~If the 'requiredWhenText' metadata is specified then the implied value for this metadata is "other". Therefore, the 'requiredWhenText' metadata may be present without requiring the presence of the 'requiredWhen' metadata. However, if a value for the 'requiredWhen' metadata is specified and is not equal to "other", then the 'requiredWhenText' metadata is not inherited and a 'requiredWhenText' metadata shall not be specified in the same context.~~

~~When this metadata is equal to "other", optional 'requiredWhenText' metadata can be used to provide display text to describe the nonstandard condition.~~

~~Many properties in Clause 12 objects have their presence dependent on a standard condition.~~

~~<Definitions>~~
~~<Object name="0-DeviceObject">~~

~~…~~
~~<Unsigned name="max-segments-accepted" optional="true"~~
~~requiredWhen="segmentation-supported" … />~~
~~…~~
~~</Object>~~
~~</Definitions>~~

Y.21.4 'requiredWhenText'

This optional localizable metadata, of type String, is used to provide display text for the presence requirements ~~when the 'requiredWhen' metadata has the value of "other". While the 'requiredWhen' metadata is an enumeration of fixed strings as defined by this standard, the~~ *The* 'requiredWhenText' metadata can contain arbitrary text that shall consist of plain printable characters with no formatting markup or line breaks.

For example, ~~if the presence requirement is not one of the standard conditions, then the 'requiredWhen' metadata has the value of "other" and the members of~~ the 'requiredWhenText' metadata (encoded in XML as <RequiredWhenText> elements) *can be used to* provide the display text *for the presence requirement condition* (the default locale in this example is "en").

<Unsigned name="auxbaud" optional="true" ~~requiredWhen="other"~~ >
<RequiredWhenText>The device is configured as a gateway</RequiredWhenText>
</Unsigned>

~~If a 'requiredWhenText' metadata is present in a context without the 'requiredWhen' metadata, then the 'requiredWhen' metadata is implicitly assigned the value "other".~~

~~For example, the following shows that it is not necessary to include requiredWhen="other" in a context with a 'requiredWhenText' member (encoded in XML as a <RequiredWhenText> element).~~

~~<Unsigned name="aux-limit" >~~
~~<RequiredWhenText>The object is configured to support aux input<RequiredWhenText>~~
~~</Unsigned>~~

[Add a new entry to **History of Revisions**, p. 1429]

(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 *ce* to ANSI/ASHRAE 135-2020** 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. MS/TP Language Replacement. 2. CSML Name Aliasing. 3. Remove writableWhen and requiredWhen.