BSR/ASHRAE Addendum *bu* to
ANSI/ASHRAE Standard 135-2016

Public Review Draft

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

First Public Review (August 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 ASHARE of any product, service, process, procedure, or design, and ASHRAE expressly disclaims such.

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

[This foreword 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 X, Y, Z, X1, X2, N, 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.

135-2016*bu*-1. Introduce BACnetARRAY of BACnetLIST collection property data type

Rationale

The definition of BACnetLIST does not cover the case when a list is an element of a BACnetARRAY. In addition, the AddListElement, RemoveListElement, and ReadRange services are using the term "list", where it remains unclear what this exactly is.

A new collection property type BACnetARRAY of BACnetLIST of <datatype> is introduced, which is specified to be the required and only datatype being applicable for the array index in the AddListElement, RemoveListElement, and ReadRange services. Currently, there are no standard properties that are a BACnetARRAY of BACnetLIST.

In addition, the AddListElement, RemoveListElement, and ReadRange services language in Clause 15 is made consistent with these definitions.

[Change **Clause 12.1.5**, p. 154]

12.1.5 Array and List Properties

Some of the properties of certain BACnet objects need to represent a collection of data elements of the same type, rather than a single primitive data value or a complex datatype constructed from other datatypes. In some instances, the size of this collection of data elements is fixed, while in other instances the number of elements may be variable. In some cases the elements may need to be accessed individually or their order may be important. BACnet provides ~~two~~ *three* forms of datatypes for properties that represent a collection of data elements of the same type: ~~"BACnetARRAY" and "BACnetLIST."~~ *"BACnetARRAY", "BACnetLIST", and "BACnetARRAY of BACnetLIST"*.

[Insert new **Clause 12.1.5.X**, p. 154]

12.1.5.X Array of List Properties

A "BACnetARRAY of BACnetLIST" datatype is a structured datatype consisting of a BACnetARRAY, where all array elements are a sequence of data elements having the same datatype. When accessing a property of this datatype, then accessing an array element shall access the entire sequence of the array element. Only the AddListElement, RemoveListElement, and ReadRange services allow access to individual list elements within an array element of a property of this datatype. Array elements of this datatype shall behave as a BACnetLIST. Properties of this datatype require special encoding. See Clause 20.2.X.

[Change **Clause 15.1**, p. 679]

15.1 AddListElement Service

The AddListElement service is used by a client BACnet-user to add one or more list elements to an object property that is a ~~list~~ *BACnetLIST or is a BACnetARRAY of BACnetLIST*.

[Change **Clause 15.1.1.1.1**, p. 679]

15.1.1.1.1 Object Identifier

This parameter, of type BACnetObjectIdentifier, shall provide the means of identifying the object whose specified ~~list~~ property is to be modified by this service.

[Change **Clause 15.1.1.1.3**, p. 679]

15.1.1.1.3 Property Array Index

If the property identified above is of datatype ~~array~~ *BACnetARRAY of BACnetLIST*, this conditional parameter of type Unsigned shall be present and shall indicate the array index of the *array* element of the referenced property to be modified by this service. Otherwise, it shall be omitted. *The index value, if this parameter is present, shall not be zero.*

[Change **Clause 15.1.1.1.4**, p. 679]

15.1.1.1.4 List of Elements

This parameter specifies one or more elements that shall be added to the property specified by the 'Property Identifier' parameter, *or to the array element of the property if the 'Property Array Index' parameter is present*. The datatype of the elements of this parameter is determined by the definition of the object type for the object specified by the 'Object Identifier' parameter.

[Change **Clause 15.1.1.2**, p. 679]

15.1.1.2 Result(+)

The 'Result(+)' parameter shall indicate that the service request succeeded and all of the specified elements were added to the ~~list~~ *BACnetLIST*.

[Change **Clause 15.1.1.3**, p. 680]

15.1.1.3 Result(-)

The 'Result(-)' parameter shall indicate that the service request failed and none of the specified elements were added to the ~~list~~ *BACnetLIST*. The reason for failure is specified by the 'Error Type' parameter.

15.1.1.3.1 Error Type

This parameter consists of two component parameters: (1) an 'Error Class' and (2) an 'Error Code'. See Clause 18. The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
. . .		
There is not enough free memory for the element.	RESOURCES	NO_SPACE_TO_ADD_LIST_ELEMENT
The property or specified array element is not a ~~list~~ *BACnetLIST*.	SERVICES	PROPERTY_IS_NOT_A_LIST
An array index is provided but the property is not ~~an array~~ *a BACnetARRAY of BACnetLIST*.	PROPERTY	PROPERTY_IS_NOT_AN_ARRAY
An array index is provided that is outside the range existing in the property.	PROPERTY	INVALID_ARRAY_INDEX

15.1.1.3.2 First Failed Element Number

This parameter, of type Unsigned, shall convey the numerical position, starting at 1, of the offending element in the 'List of Elements' parameter received in the request. If the request is considered invalid for reasons other than the 'List of Elements' parameter, the 'First Failed Element Number' *parameter* shall be equal to zero.

[Change **Clause 15.1.2**, p. 680]

15.1.2 Service Procedure

After verifying the validity of the request, the responding BACnet-user shall attempt to modify the object identified in the 'Object Identifier' parameter. If the identified object exists and has the property specified in the 'Property Identifier' parameter, *and if present has the array element specified in the 'Property Array Index' parameter*, an attempt shall be made to add all of the elements specified in the 'List of Elements' parameter to the specified property *or array element of the property*. If this attempt is successful, a 'Result(+)' primitive shall be issued.

When comparing elements in the ~~List of Elements~~ *'List of Elements' parameter* with elements in the specified property *or array element of the property*, the complete element shall be compared unless the property description specifies otherwise. If one or more of the elements is already present in the ~~list~~ *BACnetLIST*, it shall be updated with the provided element, that is, the existing element is over-written with the provided element. Optionally, if the provided element is exactly the same as the existing element in every way, it can be ignored, that is, not added to the ~~list~~ *BACnetLIST*. Ignoring an element that already exists shall not cause the service to fail.

If the specified object does not exist, the specified property does not exist, *the specified array element does not exist,* or the specified property *or array element* is not a ~~list~~ *BACnetLIST*, then the service shall fail and a 'Result(-)' response primitive shall be issued. If one or more elements cannot be added to, or updated in, the ~~list~~ *BACnetLIST*, a 'Result(-)' response primitive shall be issued and no elements shall be added to, or updated in, the ~~list~~ *BACnetLIST*.

The effect of this service shall be to add to, or update in, the ~~list~~ *BACnetLIST* all of the specified elements, or to neither add nor update any elements at all.

[Change **Clause 15.2**, p. 681]

15.2 RemoveListElement Service

The RemoveListElement service is used by a client BACnet-user to remove one or more elements from the property of an object that is a ~~list~~ *BACnetLIST or is a BACnetARRAY of BACnetLIST*. ~~If an element is itself a list, the entire element shall be removed. This service does not operate on nested lists.~~

[Change **Clause 15.2.1.1.1**, p. 681]

15.2.1.1.1 Object Identifier

This parameter, of type BACnetObjectIdentifier, shall provide the means of identifying the object whose specified ~~list~~ property is to be modified by this service.

[Change **Clause 15.2.1.1.3**, p. 681]

15.2.1.1.3 Property Array Index

If the property identified above is of datatype ~~array~~ *BACnetARRAY of BACnetLIST*, this conditional parameter of type Unsigned shall be present and shall indicate the array index of the *array* element of the referenced property to be modified by this service. Otherwise, it shall be omitted. *The index value, if this parameter is present, shall not be zero.*

[Change **Clause 15.2.1.1.4**, p. 681]

15.2.1.1.4 List of Elements

This parameter specifies one or more elements that shall be removed from the property specified in the 'Property Identifier' parameter, *or from the array element of the property if the 'Property Array Index' parameter is present*. The datatype of the elements of this parameter is determined by the definition of the object type for the object specified by the 'Object Identifier' parameter.

[Change **Clause 15.2.1.3**, p. 681]

15.2.1.3 Result(-)

The 'Result(-)' parameter shall indicate that the service request failed. The reason for failure is specified by the 'Error Type' parameter. None of the elements of the specified ~~object~~ *BACnetLIST* shall be removed.

15.2.1.3.1 Error Type

This parameter consists of two component parameters: (1) an 'Error Class' and (2) an 'Error Code'. See Clause 18. The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
. . .		
A list element to be removed is not present.	SERVICES	LIST_ELEMENT_NOT_FOUND
The property or specified array element is not a ~~list~~ *BACnetLIST*.	SERVICES	PROPERTY_IS_NOT_A_LIST
An array index is provided but the property is not ~~an array~~ *a BACnetARRAY of BACnetLIST*.	PROPERTY	PROPERTY_IS_NOT_AN_ARRAY
An array index is provided that is outside the range existing in the property.	PROPERTY	INVALID_ARRAY_INDEX

15.2.1.3.2 First Failed Element Number

This parameter, of type Unsigned, shall convey the numerical position, starting at 1, of the offending element in the 'List of Elements' parameter received in the request. If the request is considered invalid for reasons other than the 'List of Elements' parameter, the 'First Failed Element Number' *parameter* shall be equal to zero.

[Change **Clause 15.2.2**, p. 682]

15.2.2 Service Procedure

After verifying the validity of the request, the responding BACnet-user shall attempt to modify the object identified in the 'Object Identifier' parameter. If the identified object exists and it has the property specified in the 'Property Identifier' parameter, *and if present has the array element specified in the 'Property Array Index' parameter*, an attempt shall be made to remove the elements in the 'List of Elements' from the property *or array element of the property* of the object.

When comparing elements of the service with entries in the affected ~~list~~ *BACnetLIST*, the complete element shall be compared unless the property description specifies otherwise. If one or more of the elements does not exist or cannot be removed because of insufficient authority, none of the elements shall be removed and a 'Result(-)' response primitive shall be issued.

[Change **Clause 15.8**, p. 692]

15.8 ReadRange Service

The ReadRange service is used by a client BACnet-user to read a specific range of data items representing a subset of data available within a specified object property. The service may be used with any ~~list~~ *BACnetLIST* or ~~array of lists~~ *BACnetARRAY of BACnetLIST* property.

[Change **Clause 15.8.1.1.3**, p. 692]

15.8.1.1.3 Property Array Index

If the property identified above is of datatype ~~array of lists~~ *BACnetARRAY of BACnetLIST*, this optional parameter of type Unsigned shall indicate the array index of the element of the property referenced by this service. If the property identified above is not of datatype ~~array of lists~~ *BACnetARRAY of BACnetLIST*, this parameter shall be omitted. The index value, *if this parameter is present,* shall not be zero.

[Change **Clause 15.8.1.1.4**, p. 692]

15.8.1.1.4 Range

This optional parameter shall convey criteria for the consecutive range *of* items within the referenced ~~property~~ *BACnetLIST* that are to be returned, as described in Clause 15.8.2. The 'Range' parameter is shown in Table 15-14. The terminology and symbology used in this table are explained in Clause 5.6.

Table 15-14. Structure of the 'Range' Parameter

. . .

15.8.1.1.4.1 By Position

The 'By Position' parameter shall indicate that the particular items to be read are referenced by an index. *For indexing of BACnetLIST items, see Clause 12.1.5.2.*

15.8.1.1.4.1.1 Reference Index

The 'Reference Index' parameter specifies the index of the first (if 'Count' is positive) or last (if 'Count' is negative) item to be read.

15.8.1.1.4.1.2 Count

The absolute value of the 'Count' parameter specifies the number of ~~records~~ *items* to be read. If 'Count' is positive, the ~~record~~ *item* specified by 'Reference Index' shall be the first ~~record~~ *item* read and returned; if 'Count' is negative the ~~record~~ *item* specified by 'Reference Index' shall be the last ~~record~~ *item*. 'Count' may not be zero.

. . .

15.8.1.1.4.2 By Sequence Number

The 'By Sequence Number' parameter shall indicate that the particular items to be read are referenced by a sequence number and that the response shall include the sequence number of the first returned item. This differs semantically from the 'By Position' parameter choice. The Reference ~~Number~~ *Index* provided in the 'By Position' choice references an item by its position in the list. In contrast, the Reference *Sequence* Number provided in the 'By Sequence Number' choice references an item by its sequence number, which it is given when the item is added to the list. Not all *BACnetLIST* ~~lists~~ *properties or array elements that are a BACnetLIST* implement the concept of a sequence number. An example of a ~~list~~ *BACnetLIST* that does implement the concept of a sequence number is the Log_Buffer property of the Trend Log object.

15.8.1.1.4.2.1 Reference Sequence Number

The 'Reference Sequence Number' parameter specifies the sequence number of the first (if 'Count' is positive) or last (if 'Count' is negative) item to be read.

15.8.1.1.4.2.1 Count

The absolute value of the 'Count' parameter specifies the number of ~~records~~ *items* to be read. If 'Count' is positive, the ~~record~~ *item* specified by 'Reference Sequence Number' shall be the first and oldest ~~record~~ *item* read and returned. If 'Count' is negative the ~~record~~ *item* specified by 'Reference Sequence Number' shall be the last and newest ~~record~~ *item* read and returned. 'Count' shall not be zero.

. . .

15.8.1.1.4.3 By Time

The 'By Time' parameter shall indicate that the particular item to be read is referenced by timestamp and that the Sequence Number of the item shall be returned in the response. This form of the service is expected to be used when searching lists that are loosely indexed by time.

15.8.1.1.4.3.1 Reference Time

If 'Count' is positive, the first ~~record~~ *item* to be read shall be the first ~~record~~ *item* with a timestamp newer than the time specified by the 'Reference Time' parameter. If 'Count' is negative, the last ~~record~~ *item* to be read shall be the newest ~~record~~ *item* with a timestamp older than the time specified by the 'Reference Time' parameter. This parameter shall contain a specific datetime value.

15.8.1.1.4.3.2 Count

The absolute value of the 'Count' parameter specifies the number of ~~records~~ *items* to be read. If 'Count' is positive, the first ~~record~~ *item* with a timestamp newer than the time specified by 'Reference Time' shall be the first and oldest ~~record~~ *item* read and returned; if 'Count' is negative, the newest ~~record~~ *item* with a timestamp older than the time specified by 'Reference Time' shall be the last and newest ~~record~~ *item*. 'Count' shall not be zero.

. . .

[Change **Clause 15.8.1.2**, p. 696]

15.8.1.2 Result(+)

The 'Result(+)' parameter shall indicate that the service request succeeded. A successful result includes the following parameters.

. . .

15.8.1.2.3 Property Array Index

If the property identified above is of datatype ~~array of lists~~ *BACnetARRAY of BACnetLIST*, this parameter of type Unsigned shall indicate the array index of the *array* element of the property referenced by this service. If the property identified above is not of datatype ~~array of lists~~ *BACnetARRAY of BACnetLIST*, this parameter shall be omitted.

15.8.1.2.4 Result Flags

This parameter, of type BACnetResultFlags, shall convey several flags that describe characteristics of the response data:

{FIRST_ITEM, LAST_ITEM, MORE_ITEMS}

The FIRST_ITEM flag indicates whether this response includes the first list ~~or array element~~ *item* (in the case of positional indexing), or the oldest timestamped item (in the case of time indexing).

The LAST_ITEM flag indicates whether this response includes the last list ~~or array element~~ *item* (in the case of positional indexing), or the newest timestamped item (in the case of time indexing)

The MORE_ITEMS flag indicates whether more items matched the request but were not transmittable within the PDU.

15.8.1.2.5 Item Count

This parameter, of type Unsigned, represents the number of items that were returned.

15.8.1.2.6 Item Data

This parameter consists of a list of the requested data.

15.8.1.2.7 First Sequence Number

This parameter, of type Unsigned32, specifies the sequence number of the first item returned. This parameter is only included if the 'Range' parameter of the request was of the type 'By Sequence Number' or 'By Time' and 'Item Count' is greater than 0.

[Change **Clause 15.8.1.3.1**, p. 697]

15.8.1.3.1 Error Type

This parameter consists of two component parameters: (1) the 'Error Class' and (2) the 'Error Code'. See Clause 18. The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
Specified property does not exist.	PROPERTY	UNKNOWN_PROPERTY
The specified property is currently not readable by the requester.	PROPERTY	READ_ACCESS_DENIED
Property is not a ~~list or array of lists~~ *BACnetLIST or BACnetARRAY of BACnetLIST*	SERVICES	PROPERTY_IS_NOT_A_LIST
An array index is provided but the property is not ~~an array~~ *a BACnetARRAY of BACnetLIST.*	PROPERTY	PROPERTY_IS_NOT_AN_ARRAY
An array index is provided that is outside the range existing in the property.	PROPERTY	INVALID_ARRAY_INDEX
The list items do not have a sequence number but a request 'By Sequence Number' was received.	*PROPERTY*	*LIST_ITEM_NOT_NUMBERED*
The list items are not timestamped but a request 'By Time' was received.	*PROPERTY*	*LIST_ITEM_NOT_TIMESTAMPED*

[Change **Clause 15.8.2**, p. 697]

15.8.2 Service Procedure

The responding BACnet-user shall first verify the validity of the 'Object Identifier', 'Property Identifier' and 'Property Array Index' parameters and return a 'Result(-)' response with the appropriate error class and code if the object or property is unknown, if the referenced ~~data~~ *property or array element* is not a ~~list or array~~ *BACnetLIST, the list items do not support the range selection type, or i*f it is currently inaccessible for another reason.

If the 'Range' parameter is not present, then the responding BACnet-user shall read and attempt to return all of the available items in the ~~list or array~~ *BACnetLIST*.

If the 'Range' parameter is present and specifies the 'By Position' parameters, then the responding BACnet-user shall read and attempt to return all of the items specified. The items specified include the item at the index specified by 'Reference Index' plus up to 'Count' - 1 items following if 'Count' is positive, or up to -1 - 'Count' items preceding if 'Count' is

negative. ~~The first element of a list shall be associated with index 1.~~ *For indexing of BACnetLIST items, see Clause 12.1.5.2.*

If the 'Range' parameter is present and specifies the 'By Time' parameter, then the responding BACnet-user shall read and attempt to return all of the items specified. If 'By Time' parameters are specified and the ~~property values~~ *items* are not timestamped an error shall be returned. If 'Count' is positive, the ~~records~~ *items* specified include the first ~~records~~ *items* with a timestamp newer than 'Reference Time' plus up to 'Count'-1 items following. If 'Count' is negative, the ~~records~~ *items* specified include the newest ~~record~~ *item* with a timestamp older than 'Reference Time' and up to -1-'Count' ~~records~~ *items* preceding. The sequence number of the first item returned shall be included in the response. The items shall be returned in chronological order.

If the 'Range' parameter is present and specifies the 'By Sequence Number' parameters, then the responding BACnet-user shall read and attempt to return all of the items specified. *If 'By Sequence Number' parameters are specified and the items are not sequence numbered, an error shall be returned.* The items specified are all items with a sequence number in the range 'Reference Sequence Number' to 'Reference Sequence Number' plus 'Count'-1 if 'Count' is positive, or in the range 'Reference Sequence Number' plus 'Count'+1 to 'Reference Sequence' if 'Count' is negative.

To avoid missing items when using chained time-based reads, the first item in the desired set should be found using the 'By Time' form of the 'Range' parameter. Subsequent requests to retrieve the remaining items in the desired set should use the 'By Sequence Number' form of the 'Range' parameter. The reason for this is that ~~lists~~ *BACnetLIST items* that include a timestamp but are ordered by time of arrival may have ~~entries~~ *items* with out-of-order timestamps due to negative time changes in the local device's clock. If items are read that match the request parameters but cannot be returned in the response, the 'Result Flags' parameter shall contain the MORE_ITEMS flag set to TRUE, otherwise it shall be FALSE. Remaining items may be obtained with subsequent requests specifying appropriately chosen parameters.

The returned response shall convey the number of items read and returned using the 'Item Count' parameter. The actual items shall be returned in the 'Item Data' parameter. If the returned response includes the first positional index and a 'By Position' request had been made, or the oldest sequence number and a 'By Sequence Number' or 'By Time' request had been made, then the 'Result Flags' parameter shall contain the FIRST_ITEM flag set to TRUE; otherwise it shall be FALSE.

If the returned response includes the last positional index and a 'By Position' request had been made, or the newest sequence number and a 'By Sequence Number' or 'By Time' request had been made, then the 'Result Flags' shall contain the LAST_ITEM flag set to TRUE; otherwise it shall be FALSE.

If there are no items in the list that match the 'Range' parameter criteria, then a Result(+) shall be returned with an 'Item Count' of 0 and no 'First Sequence Number' parameter.

[Add to **Clause 18.3**, p. 736]

18.3Error Class - PROPERTY

This Error Class pertains to problems related to identifying, accessing, and manipulating the properties of BACnet objects, whether BACnet-defined or not. Since these errors generally apply to individual property characteristics, they do not necessarily signal that an entire service request has failed.

. . .

LIST_ITEM_NOT_NUMBERED *- List items were selected by sequence number, but the list items do not support a sequence number.*

LIST_ITEM_NOT_TIMESTAMPED *- List items were selected by time, but the list items do not support a timestamp.*

[Move content of **Clause 20.2.20** to new **Clause 20.2.1.X**, p. 770]

~~20.2.20~~ *20.2.1.X* Summary of the Tagging Rules

...

[Add new **Clause 20.2.X**, p.781]

20.2.X Encoding of a BACnetARRAY of BACnetLIST datatype

A BACnetARRAY of BACnetLIST property as a whole is encoded to allow recognition of array elements. Since each list element is of the same datatype, regular Sequence-Of encoding of both the array elements and the contents of the array elements would result in a sequence of values where the array elements cannot be distinguished. Therefore, a BACnetARRAY of BACnetLIST property shall be encoded, when accessed as a whole, as if it had been defined in ASN.1 as follows:

```
Property-Value ::= SEQUENCE OF SEQUENCE {
    array-element  [0] SEQUENCE OF <datatype>
    }
```

In this ASN.1 production, the <datatype> is a placeholder for the datatype defined for the single list elements in the array elements of the BACnetARRAY of BACnetLIST property.

The context tag [0] shall always be encoded as an open/close context tag, creating a frame for list elements and delimiting array elements.

Example: BACnetARRAY of BACnetLIST of primitive data list elements

```
    Property =          BACnetARRAY of BACnetLIST of Unsigned
    Implied ASN.1:   Property-Value ::= SEQUENCE OF SEQUENCE {
                                            array-element  [0] SEQUENCE OF Unsigned
                                            }
    Value =              ((1,2,3),(),(4,5,6))
    Encoded Tag =    X'0E' (Opening Tag)
            Application Tag = Unsigned Integer (Tag Number = 2)
            Encoded Tag =     X'21'
            Encoded Data =    X'01'
            Application Tag = Unsigned Integer (Tag Number = 2)
            Encoded Tag =     X'21'
            Encoded Data =    X'02'
            Application Tag =  Unsigned Integer (Tag Number = 2)
            Encoded Tag =     X'21'
            Encoded Data =    X'03'
    Encoded Tag =    X'0F' (Closing Tag)
    Encoded Tag =    X'0E' (Opening Tag)
    Encoded Tag =    X'0F' (Closing Tag)
    Encoded Tag =    X'0E' (Opening Tag)
            Application Tag = Unsigned Integer (Tag Number = 2)
            Encoded Tag =     X'21'
            Encoded Data =    X'04'
            Application Tag = Unsigned Integer (Tag Number = 2)
            Encoded Tag =     X'21'
            Encoded Data =    X'05'
            Application Tag =  Unsigned Integer (Tag Number = 2)
            Encoded Tag =     X'21'
```


Encoded Data = X'06'
Encoded Tag = X'0F' (Closing Tag)

[Change **Clause 21, "Error"** production, 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
. . .
list-element-not-found (81),
list-item-not-numbered *(?),*
list-item-not-timestamped *(?),*
log-buffer-full (75),
. . .
-- numerical order reference
. . .
-- *see list-item-not-numbered* *(?),*
-- *see list-item-not-timestamped* *(?),*
...
}
-- 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.
}

135-2016*bu*-2 Clarifications on character and value encoding issues

Rationale

The BACnet standard indicates that received events cannot be rejected or otherwise failed by the receiver due to the character set chosen, but does not include a similar restriction for character encoding issues.

In addition, there is no error or reject enumeration defined for invalid value encoding.

The proposed changes add restrictions on event processing to disallow failing due to character encoding errors, requires consistent handling of character encoding corrections, and adds error and reject enumerations for bad value encoding.

[Add to **Clause 3.2 Terms Defined for this Standard**, p. 2, maintaining alphabetical order]

invalid character: Character-level encoding within a BACnet character string that is not defined by the governing body of the character encoding, or that breaks the rules of the governing body, or is not allowed by the rules of BACnet such as a null character when printable characters are required.

[Change **Clause 12.1.4 Asymmetry in Property Values**, p. 154]

In some devices, property values may be stored internally in a different form than indicated by the property datatype. For example, real numbers may be stored internally as integers, or with a more limited precision. This may result in the situation where a property value is changed by one of the WriteProperty services but a subsequent read returns a slightly different value. This behavior is acceptable as long as a "best effort" is made to store the written value specified.

Character strings may also be stored or processed internally using a different character set than indicated by the service request or may be modified when stored or processed in order to eliminate invalid characters. Any changes made to a character string as part of a write operation shall also be made to any character string received in another service request that is to be used for comparison or matching to a changed written character string.

For example, if a value written to an Object_Name property can be stored in a changed form, then a Who-Has service with an Object_Name parameter shall undergo the same changes such that a Who-Has containing the character string that was written to the Object_Name property will match the Object_Name value as it was stored in the object. Other examples of related character string values that are subject to the above requirement are the Present_Value, Alarm_Values, and Fault_Values properties of a CharacterString Value object.

[Insert new **Clause 12.1.X** before Clause 12.1.8, p. 157

12.1.X Handling of invalid data encoding or excessive data length

Some BACnet data values are encoded using standards such as ISO 10646 (UTF-8) or ANSI/IEEE-754 (floating point) that do not allow for all possible combinations of bit values. In addition, limitations may exist for the display or use of some BACnet data values in a particular implementation. If a service or result contains data that is properly tagged, but has either invalid value encoding or is too long, the receiver of the data shall handle the data in a way that is consistent with the service procedure described for the service.

Some services such as AcknowledgeAlarm, ConfirmedEventNotification or UnconfirmedEventNotification shall be processed using recovery mechanisms that are a local matter. Other confirmed services such as WriteProperty may respond with a Result(-) if recovery mechanisms are not used. If a confirmed service Error Type clause lists the error INVALID_DATA_ENCODING, then it is permitted to respond with a Result(-) if invalid data encoding is detected.

Detection of invalid data value encoding and data value recovery mechanisms are not required, but are permitted,

[Change AcknowledgeAlarm service **Clause 13.5.2 Service Procedure**, p. 639]

13.5.2 Service Procedure

After verifying the validity of the request, the responding BACnet-user shall attempt to locate the specified object. If the object exists and if the 'Time Stamp' parameter matches the most recent time for the event being acknowledged, then the bit in the Acked_Transitions property of the object that corresponds to the value of the 'Event State Acknowledged' parameter shall be set to 1, a 'Result(+)' primitive shall be issued, and an event notification with a 'Notify Type' parameter equal to ACK_NOTIFICATION shall be issued. Otherwise, a 'Result(-)' primitive shall be issued. An acknowledgment notification shall use the same type of service (confirmed or unconfirmed) directed to the same recipients to which a confirmed or unconfirmed event notification for the same transition type would be sent. The Time Stamp conveyed in the acknowledgment notification shall not be derived from the Time Stamp of the original event notification, but rather the time at which the acknowledgment notification is generated.

A device shall neither fail to process, nor issue a Result(-), due to an AcknowledgeAlarm service request containing an 'Acknowledgment Source' parameter in an unsupported character set *or having invalid characters*. In this case, it is a local matter whether the 'Acknowledgment Source' parameter is *modified,* used as ~~provided~~ *provided,* or whether a character string in a supported character set of length 0 is used in its place.

[Change ConfirmedCOVNotification service **Clause 13.6.1.3.1 Error Type**, p. 641]

13.6.1.3.1 Error Type

This parameter shall consist of two component parameters: (1) the 'Error Class' and (2) the 'Error Code'. See Clause 18.

The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
No subscription exists for the specified object, property, and process identifier. Devices may ignore this condition and return a BACnet-SimpleACK-PDU.	SERVICES	UNKNOWN_SUBSCRIPTION
The data value encoding of a property value is not valid.	*SERVICES*	*INVALID_DATA_ENCODING*

[Change ConfirmedEventNotification service **Clause 13.8.2 Service Procedure**, p. 645]

13.8.2 Service Procedure

After verifying the validity of the request, the responding BACnet-user shall take whatever local actions have been assigned to the indicated event occurrence and issue a 'Result(+)' service primitive. If the service request cannot be executed, a 'Result(-)' service primitive shall be issued indicating the encountered error. A device shall neither fail to process, nor issue a Result(-), due to a ConfirmedEventNotification service request containing a 'Message Text' parameter in an unsupported character set *or having invalid characters*. In this case, the consumption of the ‘Message Text’ parameter is a local matter.

[Change UnconfirmedEventNotification service **Clause 13.9.2 Service Procedure**, p. 647]

13.9.2 Service Procedure

Since this is an unconfirmed service, no response primitives are expected. Actions taken in response to this notification are a local matter. A device shall not fail to process a request due to a 'Message Text' parameter in an unsupported character set *or having invalid characters*. In this case, the consumption of the ‘Message Text’ parameter is a local matter.

[Change ConfirmedCOVNotificationMultiple service **Clause 13.17.1.3.1 Error Type**, p. 668]

13.17.1.3.1 Error Type

This parameter shall consist of two component parameters: (1) the 'Error Class' and (2) the 'Error Code'. See Clause 18.

The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
No subscription exists for one of the specified objects, properties, and process identifier. Devices may ignore this condition and return a BACnet-SimpleACK-PDU.	SERVICES	UNKNOWN_SUBSCRIPTION
The data value encoding of a property value is not valid.	*SERVICES*	*INVALID_DATA_ENCODING*

[Change AddListElement service **Clause 15.1.1.3.1 Error Type**, p. 680

15.1.1.3.1 Error Type

This parameter consists of two component parameters: (1) an 'Error Class' and (2) an 'Error Code'. See Clause 18. The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
. . .		
The data being written has a datatype not supported by the property.	PROPERTY	DATATYPE_NOT_SUPPORTED
The data value encoding is not valid for the datatype of the property.	*PROPERTY*	*INVALID_DATA_ENCODING*
. . .		

[Change CreateObject service **Clause 15.3.1.3.1 Error Type**, p. 683]

15.3.1.3.1 Error Type

This parameter consists of two component parameters: (1) an 'Error Class' and (2) an 'Error Code'. See Clause 18. The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
. . .		
The data value encoding is not valid for the datatype of the property.	*PROPERTY*	*INVALID_DATA_ENCODING*

[Change WriteProperty service **Clause 15.9.1.3.1 Error Type**, p. 700]

15.9.1.3.1 Error Type

This parameter consists of two component parameters: (1) an 'Error Class' and (2) an 'Error Code'. See Clause 18. The 'Error Class' and 'Error Code' to be returned for specific situations are as follows:

Situation	Error Class	Error Code
. . .		
The data value encoding is not valid for the datatype of the property.	*PROPERTY*	*INVALID_DATA_ENCODING*
The Priority parameter is not within the defined range of 1..16. This condition may be ignored if the property is not commandable.	SERVICES	PARAMETER_OUT_OF_RANGE

[Change WritePropertyMultiple service **Clause 15.10.1.3.1 Error Type**, p. 702]

15.10.1.3.1 Error Type

This parameter consists of two component parameters: (1) an 'Error Class' and (2) an 'Error Code'. See Clause 18. The 'Error Class' and 'Error Code' to be returned in a 'Result(-)' for specific situations are as follows:

Situation	Error Class	Error Code
. . .		
The data value encoding is not valid for the datatype of the property.	*PROPERTY*	*INVALID_DATA_ENCODING*
The Priority parameter is not within the defined range of 1..16. This condition may be ignored if the property is not commandable.	SERVICES	PARAMETER_OUT_OF_RANGE
A syntax error is encountered in the message after one or more properties have been successfully written.	SERVICES	INVALID_TAG

[Add to **Clause 18.3 Error Class – PROPERTY**, pp. 736-737, maintaining alphabetical order]

INVALID_DATA_ENCODING - The data value encoding of a property value specified in a service parameter is not valid for the datatype of the property.

[Add to **Clause 18.6 Error Class – SERVICES**, pp. 739-740, maintaining alphabetical order]

INVALID_DATA_ENCODING - The data value encoding of a value specified in a service parameter is not valid for the datatype of the parameter.

[Add to **Clause 18.9 Reject Reason**, p. 743, maintaining alphabetical order]

INVALID_DATA_ENCODING - Generated in response to a confirmed request APDU that conveys a parameter whose data value encoding is not valid for the datatype of the parameter.

[Change **Clause 21 BACnetRejectReason** production, p. 796]

```
BACnetRejectReason ::= ENUMERATED {
    other                   (0),
    buffer-overflow         (1),
```



```
        inconsistent-parameters         (2),
        invalid-parameter-data-type     (3),
        invalid-tag                     (4),
        missing-required-parameter      (5),
        parameter-out-of-range          (6),
        too-many-arguments              (7),
        undefined-enumeration           (8),
        unrecognized-service            (9),
        invalid-data-encoding           (n),
        ...
        }
        -- 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.
```

[Change **Clause 21 Error** production, pp. 798-803]

```
    Error ::= SEQUENCE {
    -- NOTE: The valid combinations of error-class and error-code are defined in Clause 18.
        error-class     ENUMERATED {
                        device          (0),
                        object          (1),
                        property        (2),
                        resources       (3),
                        security        (4),
                        services        (5),
                        vt              (6),
                        communication   (7),
                        ...
                        },
                        -- 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
                        abort-apdu-too-long                     (123),
                        abort-application-exceeded-reply-time   (124),
    …
                        incorrect-key                           (98),
                        internal-error                          (131),
                        invalid-array-index                     (42),
                        invalid-configuration-data              (46),
                        invalid-data-encoding                   (n),
                        invalid-data-type                       (9),
    …
                        -- see abort-insufficient-security      (135),
                        -- see abort-security-error             (136),
                        -- see duplicate-entry                  (137),
                        -- see invalid-value-in-this-state      (138),
                        -- see invalid-data-encoding            (n),
                        ...
```


135-2016*bu*-3 Clarify transmission of unconfirmed COV notifications

Rationale

The standard is not clear about unconfirmed COV notification requests with respect to the relationship between subscriber-process-id parameter of zero and the use of unicast vs. broadcast addressing.

The proposed changes add clarifying language that makes these distinctions and restrictions.

[Add new **Clause 13.1.2**, p. 590]

13.1.2 Transmitting Unconfirmed COV Notification Requests

There are three circumstances under which a device may transmit an UnconfirmedCOVNotification request:

1) The device has an active COV or COV-multiple subscription that specifies 'Issue-Confirmed-Notifications' is FALSE, and a change of value occurs to the subscribed-to object or property or a notification delay elapses;
2) The device is configured, as a local matter, to send unsubscribed UnconfirmedCOVNotification or UnconfirmedCOVNotificationMultiple requests;
3) The device performs the device restart procedure as defined in Clause 19.3.

In the first case, since the subscriber device is known, the UnconfirmedCOVNotification or UnconfirmedCOVNotificationMultiple request shall be transmitted as a unicast to the subscriber device.

In the second case, the server may elect to deliver the unsubscribed UnconfirmedCOVNotification request using a unicast or broadcast address as configured. The 'Process Identifier' parameter for requests being broadcast shall be zero.

[Add to Clause 13.7, UnconfirmedCOVNotification Service, p. 642]
[Add to Clause 13.14, SubscribeCOV Service, p. 657]
[Add to Clause 13.15, SubscribeCOVProperty Service, p. 659]
[Add to Clause 13.16, SubscribeCOVPropertyMultiple Service, p. 662]
[Add to Clause 13.18, UnconfirmedCOVNotificationMultiple Service, p. 670]

. . .

When issuing COV notifications as a result of a subscription or resubscription, the notification shall always be unicast to the subscriber (See Clause 13.1.2).

135-2016*bu*-4 Clarify logging of event notifications

Rationale

A BACnet Event Log is perfectly suited to be used for storing information generated locally, including event notifications never transmitted as an APDU.

The amended language of the second paragraph of the Event Log preamble Clause 12.27 adds a sentence to explicitly support this.

[Change **Clause 3.2, event-initiating object** entry, p. 2]

event-initiating object: an object that ~~is configured to monitor its event state and can report changes in its~~ *generates* event ~~state.~~ *notifications.*

[Change **Clause 12.27** as follows]

12.27 Event Log Object Type

An Event Log object records event notification~~s~~ with timestamps and other pertinent data in an internal buffer for subsequent retrieval. Each timestamped buffer entry is called an event log "record."

Each Event Log object maintains an internal, optionally fixed-size buffer. This buffer fills or grows as event log records are added. If the buffer becomes full, the least recent records are overwritten when new records are added, or collection may be set to stop. Event log records are transferred as BACnetEventLogRecords using the ReadRange service. The buffer may be cleared by writing a zero to the Record_Count property. The determination of which *event* notifications are placed into the log is a local ~~matter.~~ *matter, including those never transmitted as an APDU.* Each record in the buffer has an implied SequenceNumber that is equal to the value of the Total_Record_Count property immediately after the record is added.

. . .

135-2016*bu*-5 Clarify recording of status events in log buffers

Rationale

The current language is not clear on recording of status events in log buffers when logging is enabled or disabled.

The proposed changes on the Enable property clarify this language to cover all situations.

[Change **Clause 12.25.5**, p. 301]

12.25.5 Enable

This property, of type BOOLEAN, indicates and controls whether (TRUE) or not (FALSE) logging of events and collected data is enabled. Logging occurs if and only if Enable is TRUE, Local_Time is on or after Start_Time, and Local_Time is before Stop_Time. If Start_Time contains an unspecified datetime, then it shall be considered equal to 'the start of time'. If Stop_Time contains an unspecified datetime, then it shall be considered equal to 'the end of time'. *As an exception, log* ~~Log~~ records of type ~~logstatus~~ *log-status in which the BUFFER_PURGED flag is set to TRUE shall always be recorded* ~~are recorded without regard to the value of the Enable property~~.

. . .

[Change **Clause 12.27.8**, p. 319]

12.27.8 Enable

This property, of type BOOLEAN, indicates and controls whether (TRUE) or not (FALSE) logging of events is enabled. Logging occurs if and only if Enable is TRUE, Local_Time is on or after Start_Time, and Local_Time is before Stop_Time. If Start_Time contains an unspecified datetime, then it shall be considered equal to 'the start of time'. If Stop_Time contains an unspecified datetime, then it shall be considered equal to 'the end of time'. *As an exception,* ~~Log~~ *log* records of type log-status *in which the BUFFER_PURGED flag is set to TRUE shall always be recorded* ~~are recorded without regard to the value of the Enable property~~.

. . .

[Change **Clause 12.30.8**, p. 341]

12.30.8 Enable

This property, of type BOOLEAN, indicates and controls whether (TRUE) or not (FALSE) logging of events and collected data is enabled. Logging occurs if and only if Enable is TRUE, Local_Time is on or after Start_Time, and Local_Time is before Stop_Time. If Start_Time contains an unspecified datetime, then it shall be considered equal to 'the start of time'. If Stop_Time contains an unspecified datetime, then it shall be considered equal to 'the end of time'. As an exception, ~~Ll~~og records of type ~~logstatus~~ *log-status in which the BUFFER_PURGED flag is set to TRUE shall always be recorded*~~are recorded without regard to the value of the Enable property~~.

. . .

135-2016*bu*-6. Clarify the Event Enrollment object reliability evaluation.

Rationale

In Clause 12.12.21, the wording is unclear as to the value of the Reliability property of an Event Enrollment object when the monitored object is in FAULT event state.

The proposed changes clarify this language.

[Change **Clause 12.12**, p. 222]

12.12 Event Enrollment Object Type

. . .

For the reliability indication by the Reliability property of the Event Enrollment object, internal unreliable operation such as configuration error or communication failure takes precedence over reliability indication for the monitored object ~~(i.e. MONITORED_OBJECT_FAULT)~~. Fault indications determined by the fault algorithm, if any, have least precedence.

. . .

[Change **Clause 12.12.21**, p. 227]

12.12.21 Reliability
This property, of type BACnetReliability, provides an indication ~~of~~ *of:*

- *the ability* ~~reliability~~ of the Event Enrollment object to perform its monitoring function,
- ~~in addition to~~ the reliability of the monitored object, *and*
- *the result of the fault algorithm, if one is in use*.

~~When determining the reliability of an Event Enrollment object, first the reliability of the Event Enrollment object itself shall be checked, followed by the reliability of the monitored object, and finally the reliability conditions are checked by the fault algorithm, if one is in use. If one of these evaluations encounters a value other than NO_FAULT_DETECTED, the sequence of evaluations shall be stopped and that reliability value shall be stored in the Reliability property.~~

The reliability-evaluation in an Event Enrollment object is processed in the following order:

1) *Evaluate the reliability of the Event Enrollment object itself.*
2) *If the reliability is NO_FAULT_DETECTED, then if the reliability of the monitored object is not NO_FAULT_DETECTED, then the reliability shall be MONITORED_OBJECT_FAULT.*
3) *If the reliability is NO_FAULT_DETECTED, then evaluate the fault algorithm, if one is in use, to determine the reliability.*

If a fault algorithm is applied, then this property shall be the pCurrentReliability parameter for the object's fault algorithm. See Clause 13.4 for fault algorithm parameter descriptions.

135-2016*bu*-7. Clarify the Global Group object reliability evaluation.

Rationale

Clause 12.1.8 clearly states that internally generated faults take precedence over fault algorithms, while Clause 12.50.11 for the Global Group object's Reliability property states it is a local matter what fault type takes precedence.

The proposed changes remove duplicate and contradictory language in Clause 12.50.11.

[Change **Clause 12.1.8**, p. 156]

12.1.8 Reliability

Several object types defined in this clause have a property called "Reliability" that indicates the existence of fault conditions for the object. Reliability-evaluation is the process of determining the value for this property. The first stage of reliability-evaluation is internal to the object and is completely defined by the device's vendor. The second stage, which is only found in certain object types, is the application of a fault algorithm. *Faults detected in the first stage shall take precedence over faults detected in the second stage.* See Clause 13.4 for fault algorithm definitions and see the object type definitions to determine the fault algorithm supported by any particular object type. The different values that the Reliability property can take on are described below. Note that not all values are applicable to all object types.

. . .

[Change **Clause 12.50.11**, p. 470]

12.50.11 Reliability

This property, of type BACnetReliability, provides an indication of whether the Present_Value is "reliable" as far as the BACnet device or operator can determine. ~~If the FAULT flag of the Member_Status_Flags has a value of TRUE, then the value of this property shall be MEMBER_FAULT.~~ If one or more group member values cannot be updated because of a communication failure, the value of this property shall be COMMUNICATION_FAILURE. ~~If the conditions for a MEMBER_FAULT and a COMMUNICATION_FAILURE are both present, the selection of which value to use is a local matter.~~

If a fault algorithm is applied, then this property shall be the pCurrentReliability parameter for the object's fault algorithm. See Clause 13.4 for fault algorithm parameter descriptions.

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

HISTORY OF REVISIONS

. . .	. . .	**. . .**
1	X	**Addendum *bu* to ANSI/ASHRAE Standard 135-2016** Approved by ASHRAE on MONTH DAY, 20XX; and by the American National Standards Institute on MONTH DAY, 20XX. 1. Introduce BACnetARRAY of BACnetLIST collection property data type. 2. Clarifications on character and value encoding issues. 3. Clarify transmission of unconfirmed COV notifications. 4. Clarify logging of event notifications. 5. Clarify recording of status events in log buffers. 6. Clarify the Event Enrollment object reliability evaluation. 7. Clarify the Global Group object reliability evaluation.