BSR/ASHRAE Addendum *ct* to
ANSI/ASHRAE Standard 135-2024

Public Review Draft

Proposed Addendum *ct* to Standard 135-2024, BACnet® - A Data Communication Protocol for Building Automation and Control Networks

First Publication Public Review (February 2025)
(Draft shows Proposed Changes to Current Standard)

ASHRAE, 180 Technology Parkway NW, Peachtree Corners, GA 20092

[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 135-2024 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.

135-2024*ct*-1 BACnet Resource Description Framework

Rationale

While the SI-WG (ASHRAE 223p) is developing an RDF data model that can be used to build a graph of building equipment with interconnected components, these views are created for human interpretation and not necessarily machine readable.

This addendum describes the serialization of primitive data elements and constructed data using RDF.

[**Change** Clause 3.3, pg. 21]

…

RBAC	role-based access control
RDF	*resource description framework*
REQ	request primitive

…

[**Add** new Annex, Annex YY, pg ?]

YY RDF DATA FORMAT (NORMATIVE)

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

This annex defines the structuring of BACnet content as Resource Description Framework (RDF) data for exchange with Semantic Web applications. The encoding examples in this annex use the Terse RDF Triple Language (Turtle). Other serialization formats such as RDF/XML, N-Triples, JSON-LD are syntactically different but convey identical content.

The Turtle serialization format specifies that IRIs may be written as relative or absolute IRIs or prefixed names. The prefix label is defined by using the "@prefix" directive followed by a colon followed by a partial IRI. The encoding examples used in this document assume the following prefix statement:

```
  @prefix bacnet: <https://data.ashrae.org/bacnet/> .
```

A prefixed name is turned into an IRI by concatenating the IRI associated with the prefix and the local part. The '@prefix' directive associates a prefix label with an IRI. Subsequent '@prefix' directives may re-map the same prefix label.

YY.1 Encoding Primitive Data

Primitive data is encoded as specific IRIs or RDF literals.

YY.1.1 Encoding of a Null Value

The NULL value is encoded as the IRI `bacnet:Null`.

YY.1.2 Encoding of a Boolean Value

Boolean values are encoded using the `xsd:boolean` datatype, for example `"true"^^xsd:boolean` or `"false"^^xsd:boolean`. See XSD Clause 3.3.2.

YY.1.3 Encoding of an Unsigned Integer Value

Unsigned integer values are encoded using the `xsd:unsignedInt` datatype, for example `"12"^^xsd:unsignedInt`. See XSD Clause 3.4.22.

YY.1.4 Encoding of a Signed Integer Value

Signed integer values are encoded using the `xsd:int` datatype, for example `"-12"^^xsd:int`. See XSD Clause 3.4.17.

YY.1.5 Encoding of a Real Number Value

Real number values are encoded using the `xsd:float` datatype, for example `"75.3"^^xsd:float`. See XSD Clause 3.2.4.

YY.1.6 Encoding of a Double Precision Real Value

Double precision number values are encoded using the `xsd:double` datatype, for example `"478.08"^^xsd:double`. See XSD Clause 3.2.5.

YY.1.7 Encoding of an Octet String Value

Octet string values are encoded using the xsd:hexBinary datatype, for example `"DEADBEEF"^^xsd:hexBinary`. See XSD Clause 3.3.15.

YY.1.8 Encoding of a Character String Value

Character strings are encoded as quoted literals optionally followed by a language tag, for example `"Fan Speed"` is a plain literal and `"vitesse du ventilateur"@fn` is French. Lauguage tags are defined by [RFC-3066] and are always in lowercase. See RDF Clause 3.3.

YY.1.9 Encoding of a Bit String Value

Bit strings are encoded as an `rdf:Seq` container with members encoded as named bits. Standard bit names are IRIs constructed as the bit string name followed by a period followed by the bit name as provided in the ASN.1, for example `bacnet:LogStatus.buffer-purged`. Vendors may follow other conventions for named bits.

YY.1.10 Encoding of an Enumerated Value

Enumerated values are IRIs. Standard enumeration value names are constructed using the enumeration name followed by a period followed by the enumeration name as described in the ASN.1, for example, `bacnet:EventState.high-limit`. Vendors may follow other conventions for enumerations.

YY.1.11 Encoding of a Date Value

Date values are encoded using the `xsd:date` type, for example `"2022-11-03"^^xsd:date`. See XSD Clause 3.3.9.

YY.1.12 Encoding of a Date Pattern Value

Date data that contains one or more “unspecified” fields is encoded as a `bacnet:datePattern` type, for example `"2022-11-* 1"^^bacnet:datePattern`. See Clause Y.12.14.

YY.1.12 Encoding of a Time Value

Time values with a zero hundredths value are encoded using the `xsd:time` type, for example `"15:30:05"^^xsd:time`. See XSD Clause 3.3.8.

YY.1.12 Encoding of a Time Pattern Value

Time data that contains one or more “unspecified” fields or with a non-zero hundredths value is encoded as a `bacnet:timePattern` type, for example `"15:30:*.*"^^bacnet:timePattern`. See Clause Y.12.18.

YY.1.13 Encoding of a DateTime Value

DateTime values are encoded using the `xsd:dateTime` type, for example `"2022-11-03T13:45:00.0"^^xsd:dateTime`. See XSD Clause 3.2.7.

YY.1.14 Encoding of a DateTime Pattern Value

Date data that contains one or more "unspecified" fields is encoded as a `bacnet:dateTimePattern` type, for example `"2022-11-* 1 13:45:00"^^bacnet:dateTimePattern`. See Clause Y.12.16.

YY.1.15 Encoding of an Object Identifier Value

Object identifiers are encoded as a plain literal strings of the form "T,N" where T represents the type and N represents the object instance number. The object type formatted as a decimal number with no leading zeroes, or a standard type name exactly equal to the names specified in the definition for BACnetObjectTypes in Clause 21. See Y.20.1.

YY.2 Encoding Constructed Data

RDF encodings of the ASN.1 productions in Clause 21 follow the "name-centric" approach to mapping SEQUENCE and CHOICE structured types outlined in Clause Z.1.1 which describes JSON encoding. The context specific tags in the ASN.1 notation are not used in the RDF encoding.

YY.2.1 Encoding of a Sequence Value

An instance of a SEQUENCE is a set of RDF statements with a common subject resource identifier, each sequence element is a `bacnet:` prefixed predicate and the object is a literal encoded per YY.1 or another RDF node. Sequences may be explicitly typed by the sequence name replacing the BACnet prefix with the `bacnet:` namespace prefix.

For example, a BACnetDeviceObjectReference is a sequence with an optional device-identifier element and a required object-identifier element. This is an example of an explicitly-typed RDF blank node with just the object-identifier:

```
[] a bacnet:DeviceObjectReference ;
      bacnet:object-identifier "analog-value,1" .
```

This is a similar example with an implicit node type and both device and object identifiers are provided:

```
[] bacnet:device-identifier "device,123" ;
      bacnet:object-identifier "analog-value,1" .
```

YY.2.2 Encoding of a Sequence-Of Value

Encoding a SEQUENCE OF type is an `rdf:Seq` where each element is the same type.

For example, the Structured View Object Type (see Clause 12.29) contains a Subordinate_Tags property which is an array BACnetNameValueCollection, which in turn contains a members element which is a SEQUENCE OF BACnetNameValue items:

```
<ex:svo-1> bacnet:object-type bacnet:ObjectType.structured-view ;
      bacnet:object-identifier "structured-view,1" ;
      bacnet:subordinate-tags (
            [ bacnet:members (
                  [ bacnet:name "temp" ]
                  [ bacnet:name "room" ; bacnet:value "B03" ]
               )
```

```
                  ]
             ) .
```

YY.2.3 Encoding of a Choice Value

An instance of a CHOICE is an RDF statement where the chosen sequence element is a `bacnet:` prefixed predicate and the object is a literal encoded per YY.1 or another RDF node. Choices may be explicitly-typed by the choice name replacing the BACnet prefix with the `bacnet:` namespace prefix. See Clause YY.5.

For example, a BACnetHostAddress is a choice of none, an IP address, or an Internet host name. This is an example of an explicitly-typed RDF blank node with the "none" choice:

```
    [] a bacnet:HostAddress ;
         bacnet:none bacnet:Null .
```

This is a similar example where the node type is implicit and an IPv4 address is provided:

```
    [] bacnet:ip-address "C0A8010A"^^xsd:hexBinary .
```

YY.2.4 Encoding of a Value of the ANY Type

The value of an ANY type is typically a primitive data element encoded as an RDF literal according to Clause YY.1. Other data structures are encoded as nodes with primitive data elements (like a SEQUENCE) or lists of nodes composed of primitive data elements (like a SEQUENCE OF) according to Clause YY.2.

YY.2.5 Encoding of an Array and List Value

Array and List values are encoded as an `rdf:Seq` of values.

YY.3 Encoding Objects

Objects are encoded as a set of RDF statements with a common subject resource identifier, a predicate IRI with the `bacnet:` namespace prefix and property identifier as specified in the BACnetPropertyIdentifier ASN.1 production in Clause 21, and primitive property value encoded according to Clause YY.1 or constructed value according to Clause YY.2. See ZZ.1 for an example of an encoded object.

YY.4 Encoding Devices

Devices are encoded as an RDF node with the relation `bacnet:contains` referencing each of the objects in the device. See YY.3 for object encoding.

YY.5 BACnet RDFS Schema References

RDF Schema (RDFS) provides a data modeling vocabulary for RDF data. This vocabulary provides a "class" concept and predicates to explicitly specify that an RDF resource is an instance of the set of members of the class and rules for the relationships between the sets members of classes.

The BACnet RDFS Schema provides a collection of class names that correspond to the constructed type names in Clause 21 in addition to object type names in Clause 12. These class names can be used to provide explicit type information when the type cannot be implicitly determined.

RDF statements use the `rdf:type` predicate to associate a resource identifier to a BACnet RDFS Schema class.

YY.5.1 ASN.1 Constructed Type Names

Clause 21.6 defines constructed types that all use the "BACnet" name prefix which follow the ASN.1 naming conventions, for example, an access rule of an Access Rights Object is called "BACnetAccessRule". For each constructed type, there is a corresponding IRI reference of an RDF Schema Class using the `bacnet:` prefix, for example:

```
bacnet:AccessRule rdf:type rdfs:Class .
```

This class IRI can be used to explicitly state that a specific resource is a member of the class extension of the class. This is an example of a partially specified access rule, not all of the required elements are provided:

```
<ex:ar-1> rdf:type bacnet:AccessRule ;
      bacnet:enable true .
```

YY.5.2 ASN.1 Constructed Anonymous Type Names

In some cases, the ASN.1 definition of a constructed type includes anonymous elements that are defined inline with the element. For example, the BACnetAccessRule defines the "time-range-specifier" element to be an instance of an anonymous enumerated type. For these elements, the BACnet RDFS Schema has a class name that is the ASN.1 sequence name followed by a period followed by the element name translated to title-case:

```
bacnet:AccessRule.TimeRangeSpecifier rdfs:subClassOf bacnet:EnumerationKind .
/.
```

Following the naming convention for enumerated values in Clause YY.1.10, the IRI for the "always" enumeration value is the `rdfs:Class` name followed by a period followed by the enumeration name:

```
<ex:ar-1> rdf:type bacnet:AccessRule ;
      bacnet:time-range-specifier
            bacnet:AccessRule.TimeRangeSpecifier.always ;
      bacnet:enable true .
```

YY.5.2 Object Type Names

The BACnet RDFS Schema specifies RDFS class names that correspond to the object type name in title-case with the embedded hypen removed in the case of multi-state objects. For example, Clause 12.1.4 defines an Analog Value Object Type and the corresponding class name is

```
bacnet:AnalogValueObject rdfs:subClassOf bacnet:Object .
```

YY.5.3 BACnet RDFS Schema Classes

BACnet/RDF provides a set of classes that are common concepts in the standard defined in Clause 3.2 and elsewhere but not formally labeled.

YY.5.3.1 bacnet:Device Class

This is the class of all devices, physical or virtual, that supports digital communication using the BACnet protocol.

YY.5.3.2 bacnet:Object Class

This is the abstract superclass of all BACnet object classes. Each member of this class must also be a member of one and only one of its subclasses.

YY.5.3.3 bacnet:EnumerationKind Class

This is the abstract superclass of all BACnet enumeration kinds.

YY.5.3.4 bacnet:EnumerationValue Class

This is the abstract superclass of all BACnet enumeration values.

YY.5.4 BACnet RDFS Schema Predicates

BACnet/RDF provides predicates that are common relationships in the standard defined in Clause 3.2 and elsewhere but not formally labeled.

YY.5.4.1 bacnet:contains Predicate

This predicate relates a bacnet:Device object to an object within the device. For example, this specifies that ex:dev-23 is a bacnet:Device instance that has a bacnet:AnalogValueObject instance with an instance number 45:

```
      <ex:dev-23> bacnet:contains [
            bacnet:object-identifier "analog-value,45" ;
            ] .
```

The same content with explicit type information:

```
      <ex:dev-23> rdf:type bacnet:Device ;
            bacnet:contains [
                  rdf:type bacnet:Object, bacnet:AnalogValueObject ;
                  bacnet:object-identifier "analog-value,45" ;
            ] .
```

More formally defined using RDFS:

```
      bacnet:contains rdf:type rdf:Property ;
            rdfs:domain bacnet:Device ;
            rdfs:range bacnet:Object .
```

YY.5.4.2 bacnet:device-instance Predicate

This predicate relates a bacnet:Device object to its device instance number. For example, this specifies that ex:dev-23 is a bacnet:Device instance that has a bacnet:device-instance instance with an instance number 23:

```
      <ex:dev-23> bacnet:device-instance 23 .
```

YY.5.4.3 bacnet:device-address Predicate

This predicate relates a bacnet:Device object to its BACnet address. For example, this specifies that ex:dev-23 is a bacnet:Device instance that has an IPv4 address 192.168.1.10, port 47808 on BACnet/IPv4 network 5:

```
      <ex:dev-23> bacnet:device-address [
            bacnet:network-number 5 ;
            bacnet:mac-address "C0A8010ABAC0"^^xsd:hexBinary ;
      ] .
```

The bacnet:network-number and bacnet:mac-address have the same meaning and encoding as a bacnet:Address instance, with the additional restriction that the bacnet:mac-address cannot be a string of length zero which would indicate a broadcast address.

YY.5.4.4 bacnet:name Predicate

This predicate relates a bacnet:PropertyIdentifierEnumerationValue object to its ASN.1 identifier. See Clause YY.5.4.6.

YY.5.4.5 bacnet:predicate Predicate

This predicate relates a bacnet:PropertyIdentifierEnumerationValue object to the predicate used to relate a bacnet:Object to the property value. See Clause YY.5.4.6.

More formally defined using RDFS:

```
      bacnet:predicate rdf:type rdf:Property ;
            rdfs:domain bacnet:PropertyIdentifierEnumerationValue ;
            rdfs:range rdf:Property .
```

YY.5.4.6 bacnet:value Predicate

This predicate relates a bacnet:PropertyIdentifierEnumerationValue object to the enumeration value. For example, this specifies that the Present Value property has the ASN.1 name present-value, uses the IRI bacnet:present-value to relate an object to a value, and has the encoded value 77:

```
      bacnet:PropertyIdentifier.object-name a
              bacnet:PropertyIdentifierEnumerationValue ;
          bacnet:name "object-name" ;
          bacnet:predicate bacnet:object-name ;
          bacnet:value "77"^^xsd:nonNegativeInteger .
          .
```

[**Add** new Annex, Annex ZZ, pg. ?]

ZZ – EXAMPLES OF RDF ENCODING (INFORMATIVE)

(This annex is not part of this standard but is included for informative purposes only.)

This annex provides examples of RDF encoding of objects and properties in ANNEX YY. These examples are provided in the Terse RDF Triple Language (Turtle) and assume the following common prefix declarations:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix bacnet: <http://data.ashrae.org/bacnet/2020#> .
```

For the purpose of providing named IRI nodes, the examples assume the following prefix declaration:

```
@prefix ex: <http://example.org/ex> .
```

ZZ.1 Example Object

The following example illustrates an Analog Input Object using the `bacnet:object-type` property to reference the ObjectType enumeration.

```
<ex:aio-1> bacnet:object-type bacnet:ObjectType.analog-input ;
      bacnet:object-identifier "analog-input,1" ;
      bacnet:object-name "OAT" ;
      bacnet:description "Outside air temperature" .
```

ZZ.2 Example Binary Value Object

The following example illustrates a Binary Value Object using `rdf:type` to describe the type of RDF node. The `bacnet:object-type` property value can be inferred.

```
<ex:bv-2> rdf:type bacnet:BinaryValueObject ;
      bacnet:object-identifier "binary-value,2" ;
      bacnet:object-name "OCCENAB" ;
      bacnet:description "Occupancy control enabled" ;
      bacnet:present-value bacnet:BinaryPV.active ;
      bacnet:out-of-service false ;
      bacnet:event-state bacnet:EventState.normal .
```

In this case, the example provides “telemetry data” such as the present value in addition to “configuration data.” What kinds of data must be provided to satisfy the requirements of a particular application is a local matter.

ZZ.3 Example Device Object

The following example illustrates a Device Object with a device-address-binding list containing a reference to another device.

```
<ex:dev-15> rdf:type bacnet:DeviceObject ;
      bacnet:object-identifier "device,15" ;
      bacnet:vendor-identifier 555 ;
      bacnet:device-address-binding (
      [     bacnet:device-identifier "device,16" ;
            bacnet:device-address [
                  bacnet:network-number 6 ;
                  bacnet:mac-address "4A"^^xsd:hexBinary
                  ]
            ]
            ) .
```

ZZ.4 Example Choice

The following example illustrates a reference to an ASN.1 Choice, the Client_COV_Increment property of a Trend Log Object, see Clause 12.25.11.

```
      <ex:log-75> bacnet:client-cov-increment [
            bacnet:real-increment "0.05"^^xsd:float ;
            ] .
```

The alternative choice would be to specify the default increment.

```
      <ex:log-76> bacnet:client-cov-increment [
            bacnet:default-increment bacnet:Null ;
            ] .
```

ZZ.5 Example with anonymous type

Some ASN.1 productions include nested sequences or choices. For example, the BACnetLandingCallStatus sequence contains a “command” element which is an unnamed choice of a “direction” or a “destination”. The following example illustrates an instance with the “direction” choice.

```
      [] bacnet:floor-number 4 ;
            bacnet:command [
                  bacnet:direction bacnet:LiftCarDirection.stopped
            ] ;
            bacnet:floor-text "Tenant Offices" .
```

This same instance may provide explicit rdf:type statements referencing the BACnet Schema (See Clause YY.5):

```
      [] rdf:type bacnet:LandingCallStatus ;
            bacnet:floor-number 4 ;
            bacnet:command [
                  rdf:type bacnet:LandingCallStatus.Command ;
                  bacnet:direction bacnet:LiftCarDirection.stopped
            ] ;
            bacnet:floor-text "Tenant Offices" .
```

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

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

HISTORY OF REVISIONS

. . .	. . .	**. . .**
1	X	**Addendum *ct* to ANSI/ASHRAE Standard 135-2024** Approved by ASHRAE on MONTH DAY, 20XX; and by the American National Standards Institute on MONTH DAY, 20XX. 1. BACnet Resource Description Framework ("BACnet RDF")