INTERNATIONAL STANDARD

ISO 16484-5

Second edition
2007-03-15

AMENDMENT 1
2009-05-15

Building automation and control systems —

Part 5: Data communication protocol

AMENDMENT 1

Systèmes d'automatisation et de gestion technique du bâtiment —

Partie 5: Protocole de communication de données

AMENDEMENT 1

Reference number
ISO 16484-5:2007/Amd.1:2009(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO 16484-5:2007 was prepared by Technical Committee ISO/TC 205, *Building environment design*.

Introduction

The purpose of this Addendum is to add a number of independent substantive changes to the BACnet standard. The changes are summarized below.

135-2004*a*-1, p. 1: Revise Life Safety Point and Life Safety Zone object types to modify their behaviour when placed out of service.

135-2004*c*-1, p. 4: Add BACnet/WS Web Services Interface.

135-2004*d*-1, p. 39: Add a new Structured View object type.

135-2004*d*-2, p. 45: Allow acknowledgement of unseen TO-OFFNORMAL event notifications.

135-2004*d*-3, p. 46: Relax the Private Transfer and Text Message BIBB requirements.

135-2004*d*-4, p. 47: Exclude LIFE_SAFETY and BUFFER_READY notifications from the Alarm Notifications BIBBs.

135-2004*d*-5, p. 49: Establish the minimum requirements for a BACnet device with an application layer.

135-2004*d*-6, p. 51: Remove the requirement for the DM-DOB-A BIBB from the B-OWS and B-BC device profiles.

135-2004*d*-7, p. 52: Relax mandated values for APDU timeouts and retries when configurable, and change default values.

135-2004*d*-8, p. 53: Fix EventCount handling error in MS/TP Master Node State Machine.

135-2004*d*-9, p. 54: Permit routers to use a local network number in Device_Address_Binding.

135-2004*d*-10, p. 55: Identify conditionally writable properties.

135-2004*d*-11, p. 56: Specify Error returns for the AcknowledgeAlarm service.

135-2004*e*-1, p. 58: Add a new Load Control object type.

135-2004*f*-1, p. 71: Add new Access Door object type.

In this Amendment, text being added to existing clauses of ISO 16484-5 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Plain type is used throughout where entirely new subclauses are added.

Building automation and control systems —

Part 5:
Data communication protocol

AMENDMENT 1

135-2004*a*-1. Revise Life Safety Point and Life Safety Zone object types for out-of-service operation.

Addendum 135-2004*a*-1

[Change **Table 12-18**, p. 194]

Table 12-18. Properties of the Life Safety Point Object Type

Property Identifier	Property Datatype	Conformance Code
...	...	...
Present_Value	BACnetLifeSafetyState	~~R~~[+]*R*
Tracking_Value	BACnetLifeSafetyState	~~O~~*R*[*1*]
...	...	...

[Change **12.15.4**, p. 195]

12.15.4 Present_Value

This property, of type BACnetLifeSafetyState, reflects the state of the Life Safety Point object. The means of deriving the Present_Value shall be a local matter. Present_Value may latch non-NORMAL state values until reset. ~~The Present_Value property shall be writable when Out_Of_Service is TRUE.~~

[Change **12.15.5**, p. 195]

12.15.5 Tracking_Value

This ~~optional~~ property, of type BACnetLifeSafetyState, reflects the non-latched state of the Life Safety Point object. The means of deriving the state shall be a local matter. Unlike Present_Value, which may latch non-NORMAL state values until reset, Tracking_Value shall continuously track changes in the state. *The Tracking_Value property shall be writable when Out_Of_Service is TRUE.*

[Change **12.15.11**, p. 196]

12.15.11 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the input(s) or process the object represents is not in service. This means that changes to the ~~Present_Value~~ *Tracking_Value* property are decoupled from the input(s) or process when the value of Out_Of_Service is TRUE. In addition, the Reliability property and the corresponding state of the FAULT flag of the Status_Flags property shall be decoupled when Out_Of_Service is TRUE. While the Out_Of_Service property is TRUE, the ~~Present_Value~~ *Tracking_Value* and Reliability properties may be changed to any value as a means of simulating specific fixed conditions or for testing purposes. Other functions that depend on the state of the ~~Present_Value~~ *Tracking_Value* or Reliability properties shall respond to changes made to these properties while Out_Of_Service is TRUE, as if those changes had occurred to the input(s) or process.

[Change **Table 12-19**, p. 200]

Table 12-19. Properties of the Life Safety Zone Object Type

Property Identifier	Property Datatype	Conformance Code
...	...	...
Present_Value	BACnetLifeSafetyState	~~R~~[1] *R*
Tracking_Value	BACnetLifeSafetyState	~~O~~ *R*[1]
...	...	...

[Change **12.16.4**, p. 201]

12.16.4 Present_Value

This property, of type BACnetLifeSafetyState, reflects the state of the Life Safety Zone object. The means of deriving the Present_Value shall be a local matter. Present_Value may latch non-NORMAL state values until reset. ~~The Present_Value property shall be writable when Out_Of_Service is TRUE.~~

[Change **12.16.5**, p. 201]

12.16.5 Tracking_Value

This ~~optional~~ property, of type BACnetLifeSafetyState, reflects the non-latched state of the Life Safety Zone object. The means of deriving the state shall be a local matter. Unlike Present_Value, which may latch non-NORMAL state values until reset, Tracking_Value shall continuously track changes in the state. *The Tracking_Value property shall be writable when Out_Of_Service is TRUE.*

[Change **12.16.11**, p. 202]

12.16.11 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the input(s) or process the object represents is not in service. This means that changes to the ~~Present_Value~~ *Tracking_Value* property are decoupled from the input(s) or process when the value of Out_Of_Service is TRUE. In addition, the Reliability property and the corresponding state of the FAULT flag of the Status_Flags property shall be decoupled when Out_Of_Service is TRUE. While the Out_Of_Service property is TRUE, the ~~Present_Value~~ *Tracking_Value* and Reliability properties may be changed to any value as a means of simulating specific fixed conditions or for testing purposes. Other functions that depend on the state of the ~~Present_Value~~ *Tracking_Value* or Reliability properties shall respond to changes made to these properties while Out_Of_Service is TRUE, as if those changes had occurred to the input(s) or process.

[Change **Annex C**, p.459]

LIFE-SAFETY-POINT :: = SEQUENCE {

…		
present-value	[85]	BACnetLifeSafetyState,
tracking-value	[164]	BACnetLifeSafetyState ~~OPTIONAL~~,
description	[28]	CharacterString OPTIONAL,
…		
}		

[Change **Annex C**, p.460]

LIFE-SAFETY-ZONE :: = SEQUENCE {

…		
present-value	[85]	BACnetLifeSafetyState,
tracking-value	[164]	BACnetLifeSafetyState ~~OPTIONAL~~,
description	[28]	CharacterString OPTIONAL,
…		
}		

135-2004*c*-1. Adding BACnet/WS Web Services Interface

Rationale
"Web services" is emerging as the predominant technology for the integration of a wide variety of enterprise information. This addendum defines a standard means of using Web services to integrate facility data from disparate data sources, including BACnet networks, with a variety of business enterprise applications.

Addendum 135-2004*c*-1

[Add new **Annex N**]

ANNEX N - BACnet/WS WEB SERVICES INTERFACE (NORMATIVE)

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

This annex defines a data model and Web service interface for integrating facility data from disparate data sources with a variety of business management applications. The data model and access services are generic and can be used to model and access data from any source, whether the server owns the data locally or is acting as a gateway to other standard or proprietary protocols.

Implementations of the services described in this standard shall conform to the Web Services Interoperability Organization (WS-I) Basic Profile 1.0, which specifies the use of Simple Object Access Protocol (SOAP) 1.1 over Hypertext Transfer Protocol -- HTTP/1.1 (RFC2616) and encodes the data for transport using Extensible Markup Language (XML) 1.0 (Second Edition), which uses the datatypes and the lexical and canonical representations defined by the World Wide Web Consortium XML Schema.

Clients may determine the version of the BACnet/WS standard that a server implements by querying a specific numerical value as defined in clause N.9. The numerical value for the version described in this document is 1.

There are three distinct usages of datatype names in this standard. Datatype names beginning with a lowercase letter, such as "string", and "nonNegativeInteger", refer to datatypes defined by the XML Schema standard. Datatype names beginning with an uppercase letter, such as "Real" or "Multistate" refer to the value types defined in Clause N.8.9. Datatype names used in a "typical language binding signature" are arbitrary and are for illustrative purposes only.

N.1 Data Model

The data structures and methods used to store information internally in a BACnet/WS server are a local matter. However, in order to exchange that information using Web services, this standard establishes a minimal set of requirements for the structuring and association of data exchanged with a BACnet/WS server.

A node is the fundamental primitive data element in the BACnet/WS data model. Nodes are arranged into a hierarchy in the data model. The topmost node in the hierarchy is known as the root node. A root node has children, but no parent. Every other node has a single parent and may optionally have children. The network visible state of a node is exposed as a collection of attributes.

Any node may have a value. The possible types for a node's value are limited to the primitive datatypes "String", "OctetString", "Real", "Integer", "Multistate", "Boolean", "Date", "Time", "DateTime", and "Duration". Nodes that have a value may also have other attributes related to that value, such as minimum, writable, etc.

An attribute is a single aspect or quality of a node, such as its value or its writability. Every node exposes a collection of attributes. Some attributes are required for all nodes, and some are conditionally required based on the value of other attributes. Some of the attributes are localizable and may return different values based on an option in a service request. Attributes are described more fully in Clause N.8.

Attributes may themselves have attributes that define a single aspect or quality of the original attribute. This standard supports this recursion syntactically, but does not define or require that any of the standardized attributes have attributes themselves at this time. Servers may provide proprietary attributes for any node or attribute at any level in the hierarchy.

A path is a character string that is used to identify a node or an attribute of a node. The hierarchy of nodes is reflected in a path as a hierarchy of identifiers arranged as a delimited series, similar to the arrangement of identifiers in a Uniform Resource Locator (URL) for the World Wide Web. A path like "/East Wing/AHU #5/Discharge Temp" identifies a node, and a path like "/East Wing/AHU #5/Discharge Temp:InAlarm" identifies the InAlarm attribute of that node. Paths are described more fully in Clause N.2.

To allow for an arbitrary number of logical arrangements of nodes, a single node may logically appear to be in more than one place in the hierarchy through the use of a reference node. Reference nodes may be used to build alternate logical arrangements of nodes since the children of a reference node may differ from that of its referent node. Reference nodes are described more fully in Clause N.4.

The arrangement of data nodes into hierarchies and the naming of those nodes is generally a local matter. However, this standard also defines a number of standardized nodes with standardized names and locations that allow clients to obtain basic information about the server itself. These standardized nodes are described more fully in clause N.9.

N.2 Paths

A path is a character string that is used to identify a node or a specific attribute. The hierarchy of nodes is reflected in a path as a hierarchy of node identifiers arranged as a delimited series separated by forward slash ("/") characters. Similarly, the hierarchy of attributes is reflected in a path as a hierarchy of attribute identifiers arranged as a delimited series separated by colon (":") characters.

Certain services accept an optional attribute path on the end of a node path. If an attribute path is not specified to those services, the Value attribute is assumed. The attribute path is separated from the node path with a colon.

The concatenated path form is:

[/node-identifier[/node-identifier]…][:attribute-identifier[:attribute-identifier]…]

where square brackets indicate optionality and "…" indicates repetition of the previous element.

Examples: "/aaa" "/aaa/bbb" "/aaa/bbb/ccc:Description" "/aaa/bbb/ccc:Description:.foo"

All identifiers are case sensitive and shall be of non-zero length. Identifiers are not localizable and are not affected by the "locale" or "canonical" service options. A path with no node identifier ("") refers to the root of the hierarchy, and ":attribute-identifier" is the syntax for accessing the attributes of the root node.

Only printable characters may be used to construct path identifiers, and, as an additional restriction, all characters equivalent to the ANSI X3.4 "control characters" (those less than X'20') are not allowed, and neither are any characters equivalent to the following ANSI X3.4 characters: / \ : ; | < > * ? " [ ] { }

Node identifiers beginning with a period (".") character and attribute identifiers not beginning with a period (".") character are reserved for use by ASHRAE. This restriction separates node and attribute identifiers that are defined by this standard from those that are defined by the server, perhaps based on user input. Server defined node identifiers shall not start with a period, so that "/aaa/.first-floor" is invalid but "/aaa/first-floor" is valid. Conversely, all server defined attribute identifiers shall start with a period, so that "/aaa:MyNewAttribute" is invalid but "/aaa:.MyNewAttribute" is valid. This asymmetry is based on the expected common usage where most node identifiers will be server defined and most attributes are standard, making the use of periods the exception rather than the norm.

Space characters are allowed and are significant in identifiers; however, it is recommended that identifiers should not begin or end with space characters.

N.3 Normalized Points

Most building automation protocols, both standard and proprietary, have the concept of organizing data into "points" that have "values." In addition to their values, points often contain data such as "point description" or "point is in alarm." But these data may be named, structured, and/or accessed differently in different protocols.

To ensure that a Web service client can retrieve data without knowing these naming and access-method details, this standard defines "normalized points." This means that the common attributes of points available in the majority of building data models are exposed using a common set of names.

In this data model, nodes with a NodeType (see N.8.5) of "Point" are required to have a value and have a common collection of attributes that may be used to map to these data from other protocols. Some data may not be available in some protocols, in which case either the normalized attribute is absent, or it has a reasonable default value.

N.4 Reference Nodes

A node that refers to another node somewhere else in the hierarchy is termed a "reference node." The node to which it refers is its "referent node". A reference node reflects most of the attributes of its "referent node", including its type, so that for most purposes, the reference node is indistinguishable from its referent node. The use of reference nodes allows a node's data to appear in more than one place in the hierarchy.

Multiple hierarchies may be supported on a server. Automated discovery of those hierarchies may be done by starting at the root, or any other starting point, and using the Children attribute to enumerate the available nodes in a structured fashion. Two or more paths in different hierarchies may express different relationships for a single object. To denote this, and so that apparent duplicates of an object can be discerned, a node can refer to another node somewhere else in the hierarchy. It is arbitrary and a local matter which node is the referent node and which is the reference node. Multiple reference nodes can point to the same referent node, or alternately can daisy chain, one to one another, ultimately leading to a referent they all have in common which is not a reference node. There shall be at least one referent node which is not a reference node, as it is forbidden to create a loop of references.

One network-visible distinction between a reference node and its referent node is in the presence of a Reference attribute in the reference node. This attribute contains a path to the referent node. The Reference attribute is present in a node if and only if that node is a reference node.

In most cases, the distinction of whether a node is a reference node or not is unnecessary. But in those cases where the client needs to make a distinction, it can check for the presence of a Reference and act accordingly. A client can also determine, for any given node, if there are reference nodes that refer to it. This may be done with the Aliases attribute.

Except for the attributes Children, Aliases, Attributes, and Reference, any attribute read from the reference node will have the same value as when read from the referent node. The reason for this is that, when references are used to create different relationships between nodes, the nodes are not fundamentally changed by that association. Therefore, only the attributes involved in expressing the relationships between nodes, namely Children, Aliases, Attributes, and Reference, are expected to be different depending on which path was used to access the node. The Attributes node only changes as needed to reflect the changing presence or absence of the Children, Aliases or Reference attributes. Otherwise, the contents of the Attributes attribute is unchanged.

A reference node may point to another reference node, but it is not allowed to refer to itself, nor is it allowed to create a loop of references.

For example, the paths "/Geographic/East Wing/Air Handler 5/Discharge Temp" and "/Cooling/Chiller Manager/Air Handler 5/Terminal Box 345-A" express two different relationships for Air Handler 5. If the geographic relationship was modeled first, then for the cooling distribution relationship, the node identified by "/Cooling/Chiller Manager/Air Handler 5" would be a reference node with its Reference Attribute containing the path "/Geographic/East Wing/Air Handler 5".

N.5 Localization

BACnet/WS supports the creation of products that are specifically designed for particular regions of the world. The designation of a natural language, paired with a set of notational customs, such as date and number formats, is referred to as a "locale". A BACnet/WS server may support multiple locales simultaneously, and several of the attributes of a node are accessible for different locales (see clauses N.11.4, N.11.5, and N.11.6). For example, in a server that supports multiple locales, the DisplayName attribute can be used to get a user interface presentation name for the node in more than one language. Specifying a locale in a service also allows the client to request dates, times and numbers in a format appropriate to that locale.

N.6 Security

BACnet/WS does not define its own authentication mechanism; rather, this standard specifies the use of lower level Web service authentication methods defined by other standards. Some servers might not support or require any authentication at all. Others might provide authentication by means of a simple username and password using HTTP Basic authentication (defined by section 2 of HTTP Authentication: Basic and Digest Access Authentication) secured through an SSL (Secure Sockets Layer, defined by SSL Protocol Version 3.0) or TLS (Transport Layer Security, defined by TLS Protocol Version 1.0) connection. Some servers may be secured through public key certificates or more advanced options that are currently in development or yet to be defined.

For specification simplicity and increased interoperability, servers shall claim support for one or both of the following authentication and authorization mechanisms: “None”; “HTTP Basic through SSL or TLS”.

In addition to authentication, some forms of authorization can also occur before the Web services defined by this standard are invoked. For example, some Web services host environments (e.g., Application Servers) can be configured to limit users’ access to certain services based on HTTP path or SOAP method.

The content and format of errors returned from these lower level authentication and authorization methods varies and is not specified by this standard since the services defined by this standard were never invoked.

When a Web service request successfully passes through the lower levels, and the services defined by this standard are invoked, additional authentication and authorization operations may be performed by those services and the content and format of errors resulting from such operations are fully defined by this standard. The configuration of authentication and authorization policies, at any level, is a local matter.

N.7 Sessions

The Web services defined by this standard are stateless and establish no sessions between clients and servers. There is no requirement for any information to be retained on the server from one service invocation to the next. Service options such as "locale" that could be held in a session on the server are instead maintained by the client in a service options string that is provided to the server for each service invocation.

N.8 Attributes

A node is exposed to Web services as a collection of named attributes. There are two forms of attributes: those that are a primitive datatype, and those that are an array of primitive datatypes. Only the Value attribute is writable with the services defined by this standard.

While some attributes are specified as optional, the presence of those attributes on a given node is not expected to change dynamically. Clients can assume that the collection of available attributes will remain relatively stable in operation and normally will be changed only by a reconfiguration or reprogramming of the server and not in the normal course of operation. For example, even though the default value for the InAlarm attribute is "false", the InAlarm attribute is not expected to be absent when the node is not in alarm and present only when the node is in alarm. Generally, if an attribute can have a value that is different from its default during the normal course of operation, then the attribute should be present at all times.

The server may provide proprietary attributes for any node or attribute anywhere in the hierarchy of the data model. Proprietary attributes shall begin with a period ('.') character to distinguish them from standard attributes. The datatype and set of possible values for these attributes are not defined by this standard.

N.8.1 Primitive Attributes

The datatype of a primitive attribute in this standard is defined using its XML Schema datatype name, such as "boolean", "nonNegativeInteger", and "double". See Clause N.10 for details of how these are encoded for use in Web services.

The datatype of some attributes, such as Value and Minimum, is dependent on the value of the ValueType attribute. This is more fully described in Clause N.8.9.

N.8.2 Enumerated Attributes

Some primitive attributes are enumerations. Enumerated attributes are of datatype XML Schema "string", but the set of allowed values is defined by this standard. Additionally, some enumerated attributes are localizable (see N.5). In that case, the non-localized set of values is defined by this standard, but the localized strings are a local matter.

N.8.3 Array Attributes

Array attributes are attributes that contain an array of primitive values. Each element in the array has the same primitive datatype. The contents of an array attribute may be accessed either as an array of separate elements or as a single concatenation of all the elements.

The datatype of an array element in this standard is defined using its XML Schema datatype name, such as "boolean", "nonNegativeInteger", and "double". See Clause N.10 for details of how these are encoded for use in Web services.

When array attributes are accessed with a service that returns an array, such as getArray, the array elements are returned as individual strings. However, when accessed with a service that returns a single string, such as getValue, the array values are concatenated into a single string by separating the array elements with a ';' (semicolon) character, for example, "high;medium;low". The values of the individual array elements are not permitted to contain semicolons.

The server shall retain a constant order for the elements of an array attribute. Clients of services such as getArrayRange can therefore depend on this behavior to read the array an element at a time.

N.8.4 Attribute Summary

Some attributes are always required, and some are conditionally required, based on criteria outlined in the following table. The datatype referred to in the table is an XML Schema datatype name. See Clause N.10 for more information on encoding for Web services. Attributes that are not listed as Localizable are never affected by the "locale" service option (see clause N.11.4) and are always encoded in their non-localized canonical form (see clause N.11.6).

Table N-1. Attribute Summary

Attribute Identifier	Datatype	Array	Enumerated	Localizable	Presence
"NodeType"	string	No	Yes	No	Required
"NodeSubtype"	string	No	No	Yes	Optional
"DisplayName"	string	No	No	Yes	Optional
"Description"	string	No	No	Yes	Optional
"ValueType"	string	No	Yes	No	Required
"Value"	(varies - see N.8.9)	No	No	Yes	Required if ValueType is not "None"
"Units"	string	No	Yes	Yes	Required if ValueType is "Real" or "Integer"
"Writable"	boolean	No	No	No	Required if ValueType is not "None"
"InAlarm"	boolean	No	No	No	Optional
"Minimum"	(varies - see N.8.9)	No	No	Yes	Optional
"Maximum"	(varies - see N.8.9)	No	No	Yes	Optional
"Resolution"	(varies - see N.8.9)	No	No	Yes	Optional
"MinimumLength"	nonNegativeInteger	No	No	No	Optional and only present if ValueType is "String"
"MaximumLength"	nonNegativeInteger	No	No	No	Optional and only present if ValueType is "String"
"IsMultiLine"	boolean	No	No	No	Optional
"Attributes"	string	Yes	No	No	Required
"WritableValues"	string	Yes	No	Yes	Required if ValueType is "Multistate" or "Boolean" and Writable is true
"PossibleValues"	string	Yes	No	Yes	Required if ValueType is "Multistate" or "Boolean"
"Overridden"	boolean	No	No	No	Optional
"ValueAge"	double (seconds)	No	No	Yes	Optional
"Aliases"	string	Yes	No	No	Required if there are reference nodes referring to this node (see Clause N.4)
"Children"	string	Yes	No	No	Optional
"Reference"	string	No	No	No	Present if and only if the node is a reference node (see Clause N.4)
"HasHistory"	boolean	No	No	No	Required if ValueType is not "None"
"SinglyWritableLocales"	string	Yes	No	No	Present if and only if ValueType is "String" and Writable is true
"HasDynamicChildren"	boolean	No	No	No	Optional

N.8.5 NodeType

This required attribute indicates the general classification of a node. It is intended as a hint to a client application about the contents of a node, and is not intended to convey an exact definition. The list of values for this attribute is not extensible. Further refinement of classification is provided by the NodeSubtype attribute. The allowable values for this attribute are:

{"Unknown", "System", "Network", "Device", "Functional", "Organizational", "Area", "Equipment", "Point", "Collection", "Property", "Other"}

The "Unknown" type may be used for data that originated in another source and for which no type information is known. The "System" type may be used to designate an entire mechanical system. The "Network" type may be used to represent a communications network, and the "Device" type could be used to represent a physical device on that network. The "Functional" type can be used to represent a single system component such as a control module or a logical component such as a function block. The "Organizational" type is intended to represent business concepts such as departments or people. The "Area" type represents a geographical concept such as a campus, building, floor, etc. A "Point" represents a single point of data, either a physical input or output of a control or monitoring device, or a software calculation or configuration setting. An "Equipment" type may be used to represent a single piece of equipment that may be a collection of "Points". A "Collection" is just a generic container used to group things together such as a collection of references to all space temperatures in a building. The "Property" type is intended to

model data that is logically part of the parent node. The "Other" type is used for everything that does not fit into one of these broad categories.

N.8.6 NodeSubtype

This optional attribute is a string of printable characters whose content is not restricted. It provides a more specific classification of the node. For example, when the NodeType attribute has a value of "Area", the NodeSubtype attribute could have a value such as "Campus", "Building", or "Floor". This attribute may be localized, possibly returning different locale-appropriate values when a "locale" service option is specified.

N.8.7 DisplayName

This required attribute is a string of printable characters whose content is not restricted. It is used to provide a short (10-30 character) descriptive name or title for display to humans in user interfaces. It should be localized if localization is supported, returning possibly different locale-appropriate values when a "locale" service option is specified. A client may retrieve this attribute in any locale the server supports for use in creating multilingual displays. The values of the DisplayName attributes do not need to be unique among sibling nodes.

A DisplayName attribute may be different from the path identifier used to access the node. For example, for the node identified by the path "/Building 12/Room 225", the DisplayName could be "Bob's Office" in one locale and "Bureau de Bob" in another locale, or it could just be "Room 225" in all locales.

N.8.8 Description

This optional attribute is a string of printable characters whose content is not restricted. This attribute may be localized, possibly returning different locale-appropriate values when a "locale" service option is specified.

N.8.9 ValueType

This required attribute indicates the datatype of the Value attribute and attributes restricting the Value attribute. If the node has no value, then this attribute shall have the value "None". The list of values for this attribute is not extensible. The allowable values for this attribute are:

{"None", "String", "OctetString", "Real", "Integer", "Multistate", "Boolean", "Date", "Time", "DateTime", "Duration"}

The "None" type is used when the node does not have a value. The "String" type is used for nodes that have character string values that are intended to be human readable. An "OctetString" is used to contain arbitrary binary data that is typically not human readable. A "Real" is a floating point value, for example 75.6. An "Integer" is for values that that are expressed in whole numbers, for example, 1234. A "Multistate" is a value that is a choice from a set of named states, for example, {"high", "medium", "low"}. A "Boolean" is a choice between exactly two named states, such as "on" and "off", one of which is considered true and the other false. A "Date" is used to represent values that are calendar dates. A "Time" is used to represent a time of day. A "DateTime" is used to represent an exact moment in time, specifying both a date and a time. A "Duration" represents a time span, such as "5 seconds."

The representation of all value types other than "None" and "OctetString" may be affected by the "locale" service option if the server supports localization for a particular locale or locales. See clauses N.5 and N.11.4.

The effect of this attribute on the datatype of Value and related attributes is summarized in the following table. The datatypes referred to in the table are XML Schema datatype names. See Clause N.10 for more information on encoding of Web services. Attributes whose datatype is listed as n/a in the table shall not be present in the node.

Table N-2. Effect of ValueType Attribute

ValueType Attribute Value	Value Attribute Datatype	Minimum Attribute Datatype	Maximum Attribute Datatype	Resolution Attribute Datatype
"None"	n/a	n/a	n/a	n/a
"String"	string	n/a	n/a	n/a
"OctetString"	base64Binary	n/a	n/a	n/a
"Real"	double	double	double	double
"Integer"	integer	integer	integer	integer
"Multistate"	string	n/a	n/a	n/a
"Boolean"	boolean	n/a	n/a	n/a
"Date"	date	date	date	integer (days)
"Time"	time	time	time	double (seconds)
"DateTime"	dateTime	dateTime	dateTime	double (seconds)
"Duration"	double (seconds)	double (seconds)	double (seconds)	double (seconds)

N.8.10 Value

This optional attribute represents the value of the node. The datatype of this attribute is indicated by the ValueType attribute. The Value attribute is present if and only if the value of the ValueType attribute is not "None". When the ValueType attribute of the node is "String" or "Multistate", then the values of this attribute may be localized based on the "locale" service option. See Clause N.11.4.

N.8.11 Units

This optional attribute defines the engineering units for the Value attribute of the node. If the ValueType attribute is "Real" or "Integer", then this attribute is required to be present, but may have the value of "no-units". This attribute may optionally be present for other values of the ValueType attribute.

This attribute's value is available in two forms. If the "canonical" service option is false, then the value of this attribute is a string whose contents are not restricted and may be appropriate to the requested locale. If the "canonical" service option is true, then the value of this attribute is restricted to be exactly equal to one of the enumeration identifiers, such as "degrees-Celsius", "inches-of-water", etc., which are defined by the ASN.1 production for BACnetEngineeringUnits in Clause 21.

This attribute is extensible to support units other than those defined by this standard. In the case where the units of the node's value does not match one of the units defined in this standard, the value returned for this attribute when the "canonical" service option is true shall be "other", and the value returned when the "canonical" service option is false shall be a string whose contents are not restricted and may be appropriate to the requested locale.

N.8.12 Writable

This optional attribute indicates whether the Value attribute is writable through Web services. This attribute shall be present if and only if the Value attribute is present.

N.8.13 InAlarm

This optional attribute indicates whether this node is "in alarm" or not. The meaning of "in alarm" is a local matter. If the concept of "in alarm" is not appropriate to this node, then this attribute shall not be present.

N.8.14 Minimum

This optional attribute indicates the minimum value of the Value attribute. The datatype of this attribute is defined in Clause N.8.9.

N.8.15 Maximum

This optional attribute indicates the maximum value of the Value attribute. The datatype of this attribute is defined in Clause N.8.9.

N.8.16 Resolution

This optional attribute indicates the smallest change that can be represented in the value of the Value attribute. The datatype of this attribute is defined in Clause N.8.9.