ANSI/ASHRAE Addendum c to
ANSI/ASHRAE Standard 135-2004

ASHRAE STANDARD

BACnet®

A Data Communication Protocol for Building Automation and Control Networks

Approved by the ASHRAE Standards Committee on September 29, 2006; by the ASHRAE Board of Directors on September 29, 2006; and by the American National Standards Institute on October 2, 2006.

This standard is under continuous maintenance by a Standing Standard Project Committee (SSPC) for which the Standards Committee has established a documented program for regular publication of addenda or revisions, including procedures for timely, documented, consensus action on requests for change to any part of the standard. The change submittal form, instructions, and deadlines may be obtained in electronic form from the ASHRAE Web site, http://www.ashrae.org, or in paper form from the Manager of Standards. The latest edition of an ASHRAE Standard may be purchased from ASHRAE Customer Service, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. E-mail: orders@ashrae.org. Fax: 404-321-5478. Telephone: 404-636-8400 (worldwide), or toll free 1-800-527-4723 (for orders in US and Canada).

© Copyright 2006 ASHRAE, Inc.

ISSN 1041-2336

Approved American National Standard

ANSI

American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.
1791 Tullie Circle NE, Atlanta, GA 30329
www.ashrae.org

ASHRAE Standing Standard Project Committee 135
Cognizant TC: TC 1.4, Control Theory and Application
SPLS Liaison: Frank E. Jakob

William O. Swan, III, *Chair**
David Robin, *Vice-Chair*
Carl Neilson, *Secretary*
Donald P. Alexander*
Barry B. Bridges*
Coleman L. Brumley, Jr.*
Ernest C. Bryant*
James F. Butler*
Steven T. Bushby
A. J. Capowski
Craig P. Gemmill*
David G. Holmberg*
Robert L. Johnson*
Stephen Karg
J. Damian Ljungquist*
James G. Luth*
Carl J. Ruther*
David B. Thompson*
Daniel A. Traill
J. Michael Whitcomb*
David F. White*

**Denotes members of voting status when the document was approved for publication*

ASHRAE STANDARDS COMMITTEE 2006–2007

David E. Knebel, *Chair*
Stephen D. Kennedy, *Vice-Chair*
Michael F. Beda
Donald L. Brandt
Steven T. Bushby
Paul W. Cabot
Hugh F. Crowther
Samuel D. Cummings, Jr.
Robert G. Doerr
Roger L. Hedrick
John F. Hogan
Eli P. Howard, III
Frank E. Jakob
Jay A. Kohler
James D. Lutz
Carol E. Marriott
Merle F. McBride
Mark P. Modera
Ross D. Montgomery
H. Michael Newman
Stephen V. Santoro
Lawrence J. Schoen
Stephen V. Skalko
Bodh R. Subherwal
Jerry W. White, Jr.
James E. Woods
Richard D. Hermans, *BOD ExO*
Hugh D. McMillan, III, *CO*

Claire B. Ramspeck, *Assistant Director of Technology for Standards and Special Projects*

SPECIAL NOTE

This American National Standard (ANS) is a national voluntary consensus standard developed under the auspices of the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). Consensus is defined by the American National Standards Institute (ANSI), of which ASHRAE is a member and which has approved this standard as an ANS, as "substantial agreement reached by directly and materially affected interest categories. This signifies the concurrence of more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that an effort be made toward their resolution." Compliance with this standard is voluntary until and unless a legal jurisdiction makes compliance mandatory through legislation.

ASHRAE obtains consensus through participation of its national and international members, associated societies, and public review.

ASHRAE Standards are prepared by a Project Committee appointed specifically for the purpose of writing the Standard. The Project Committee Chair and Vice-Chair must be members of ASHRAE; while other committee members may or may not be ASHRAE members, all must be technically qualified in the subject area of the Standard. Every effort is made to balance the concerned interests on all Project Committees.

The Manager of Standards of ASHRAE should be contacted for:

a. interpretation of the contents of this Standard,
b. participation in the next review of the Standard,
c. offering constructive criticism for improving the Standard,
d. permission to reprint portions of the Standard.

DISCLAIMER

ASHRAE uses its best efforts to promulgate Standards and Guidelines for the benefit of the public in light of available information and accepted industry practices. However, ASHRAE does not guarantee, certify, or assure the safety or performance of any products, components, or systems tested, installed, or operated in accordance with ASHRAE's Standards or Guidelines or that any tests conducted under its Standards or Guidelines will be nonhazardous or free from risk.

ASHRAE INDUSTRIAL ADVERTISING POLICY ON STANDARDS

ASHRAE Standards and Guidelines are established to assist industry and the public by offering a uniform method of testing for rating purposes, by suggesting safe practices in designing and installing equipment, by providing proper definitions of this equipment, and by providing other information that may serve to guide the industry. The creation of ASHRAE Standards and Guidelines is determined by the need for them, and conformance to them is completely voluntary.

In referring to this Standard or Guideline and in marking of equipment and in advertising, no claim shall be made, either stated or implied, that the product has been approved by ASHRAE.

[This foreword and the "rationale" on the following page 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 revise ANSI/ASHRAE Standard 135-2004. The modifications in this addendum are the result of change proposals made pursuant to the ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135.

SSPC 135 wishes to recognize the efforts of the following people in developing this addendum: Alexander Andreyev, Howard Coleman, Sharon Dinges, Stuart Donaldson, David Fisher, John L. Hartman, Daniel Heine, Cuong Huynh, Bernhard Isler, Roland Laird, Jerald P. Martocci, Hans-Joachim Mundt, H. Michael Newman, Duffy O'Craven, Joseph S. Majewski, Dave Richards, David Ritter, and Graham Whiting. The committee is also grateful to Andrey Golovin, René Quirighetti, and Takeji Toyoda.

The changes in Addendum 135*c* are summarized below.

135-2004*c*-1. Adding BACnet/WS Web Services Interface, p. 2.

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	Enum-erated	Local-izable	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.

N.8.17 MinimumLength

This optional attribute indicates the minimum length, in characters, for the value of the Value attribute when the ValueType attribute is equal to "String".

N.8.18 MaximumLength

This optional attribute indicates the maximum length, in characters, for the value of the Value attribute when the ValueType attribute is equal to "String".

N.8.19 IsMultiLine

This optional attribute indicates that the value of the Value attribute, when the ValueType attribute is equal to "String", is intended to be capable of containing multiple lines of text. The value might not actually contain multiple lines at any given time, and it is not intended that IsMultiLine change dynamically based on the contents of the value. This attribute is primarily used as a hint to a user interface to display or edit the text in a manner capable of supporting multiple lines.

If the value contains multiple lines, the lines are separated by the character equivalent to the ANSI X3.4 control character known as "new line" or "line feed" (X'0A'). In all cases, the Value attribute is returned as a single string since the Value attribute is not an array attribute.

If IsMultiLine is missing or false, the presence of, acceptance of, or rejection of "new line" characters is a local matter.

N.8.20 Attributes

This required attribute is an array containing all of the names of the attributes present in this node.

N.8.21 WritableValues

This optional attribute is an array containing all of the string values that may be written to the Value attribute of a node whose ValueType is equal to "Multistate" or "Boolean".

N.8.22 PossibleValues

This optional attribute is an array containing all of the possible string values for the Value attribute of a node whose ValueType is equal to "Multistate" or "Boolean". For nodes that have a ValueType attribute equal to "Boolean", the first entry in the array corresponds to "true", and the second entry corresponds to "false".

N.8.23 Overridden

This optional attribute indicates that the value of the Value attribute has been overridden by some means. For physical inputs or outputs, this shall mean that the Value attribute is no longer tracking changes to the physical input or that the physical output is no longer reflecting changes made to the Value attribute.

N.8.24 ValueAge

This optional attribute indicates the time, in seconds, since the time when the value of the Value attribute was last successfully updated in the server. Caching is permitted in gateways; this attribute shall indicate the age of the cached value.

N.8.25 Aliases

This optional attribute contains the collection of paths that identify reference nodes that refer to this node.

N.8.26 Children

This optional attribute is an array that contains the collection of identifiers for the children of this node on a given path. Each of these identifiers can be used to construct a new path to a child node according to the rules set forth in clause N.2.

Note that the child identifiers specified by this attribute do not start with a '/' character, so when constructing a new path to a child node, the '/' separator will need to be used between the original path and the child identifier.

Absence of this attribute shall indicate that the node has no children. Therefore, if the node has children, this attribute is required to be present. If the node has no children, this attribute shall either be absent or present and empty.

N.8.27 Reference

This optional attribute is present if and only if the node is a reference node. The value of this attribute is a path to a referent node. See Clause N.4.

N.8.28 HasHistory

This optional attribute indicates that there are historical records for this node. Clients may use this to determine if the getHistoryPeriodic is applicable to this node.

N.8.29 SinglyWritableLocales

This optional attribute is an array that contains the collection of locales that can be used with the writeSingleLocale service option to set individual localized values for a String node. This attribute is present if and only if the ValueType attribute equals "String" and the Writable attribute is true. The collection of singly writable locales shall be a subset of the collection returned by the getSupportedLocales service.

If the server supports writing values for multiple locales on a given String node, then the SinglyWritableLocales attribute shall contain all of the locales which may be individually written and retained.

If a String node does not support the writing of individual values for different locales, then it is a local matter as to whether the server shall return one of its supported locales or an empty array for this attribute.

If the server declares multiple locales in SinglyWritableLocales and those locales are individually written to with separate values using the writeSingleLocale service option, then the server shall retain those values separately and return the appropriate value, based on the locale service option, when the node is subsequently read.

It is a local matter as to how these values are stored and whether individual storage is preallocated for each singly writable locale or if space is allocated only when separate values are needed. Note that when writing, if the writeSingleLocale service option is false, the logical behavior is that all writable locales are written simultaneously and a server with dynamic allocation may take that opportunity to revert to having only one copy of the string value since all the writable locales will contain the same value.

N.8.30 HasDynamicChildren

This optional attribute indicates that the node has a dynamic collection of children that are expected to change over time. If this attribute is missing or false, then clients can assume that the children nodes are relatively stable and are changed by a reconfiguration or reprogramming of the server and not in the normal course of operation. If this attribute is true, then clients should assume that the children nodes may change at any time and should reread the Children attribute as needed.

N.9 Standard Nodes

While the arrangement of data nodes into hierarchies and the naming of those nodes is generally a local matter, this standard also defines a number of standardized nodes with standardized names and locations that allow clients to obtain basic information about the server. These standard nodes all have names beginning with a period (".") character to distinguish them from other nodes in the server whose presence, structure and behavior is not defined by this standard.

The locations, names, types, and presence requirements of the standard nodes are summarized in the following table.

Table N-3. Standard Nodes

Node Path	ValueType	NodeType	Presence	Meaning of the Value
/.sysinfo	"None"	"Other"	Required	The /.sysinfo node is just a container for the following nodes; it has no value
/.sysinfo/.vendor-name	"String"	"Other"	Required	The name of the vendor of this server (unrestricted contents)
/.sysinfo/.model-name	"String"	"Other"	Required	The model name and/or number of this server (unrestricted contents)
/.sysinfo/.software-version	"String"	"Other"	Required	The version/revision of the software running in this server (unrestricted contents)
/.sysinfo/.standard-version	"Integer"	"Other"	Required	The version of the standard that the server is implementing, as defined in the prolog to this Annex.

N.10 Encodings

This clause defines how data is encoded for use in the Web services defined by this standard.

N.10.1 Canonical Form

This standard defines a canonical form for attribute values to allow for unambiguous machine processing. The localized forms are more suited for presentation to humans, and the canonical forms are more suited for parsing and processing by machines.

The datatypes defined for the various attributes in Clause N.8.4 are XML Schema datatypes. The XML Schema standard ("XML Schema Part 2: Datatypes") defines a "lexical representation" and a "canonical representation" for each of these datatypes. The "canonical form" defined by this standard is equal to one of the XML Schema representations, selected according to the following table. All attributes not indicated as "Localizable" in Clause N.8.4 shall always be encoded in their canonical form.

Table N-4. Examples of Localized and Canonical Forms

XML Schema Datatype	XML Schema encoding rule used for the BACnet/WS Canonical Form	Example value in BACnet/WS Localized Form	Corresponding value in BACnet/WS Canonical Form
double	XML Schema "Lexical Representation"	"7,345.23" or "7 345,23"	"7345.23" or "7.34523E3"
boolean	XML Schema "Canonical Representation"	"On" or "Run"	"true"
integer	XML Schema "Canonical Representation"	"7,345" or "7 345"	"7345"
date	XML Schema "Lexical Representation"	"13-Aug-2005" or "8-13-2005" or "13/08/05"	"2005-08-13"
time	XML Schema "Canonical Representation"	"2:03:04 PM EST" or "14:03:04 EST"	"19:03:04Z"
dateTime	XML Schema "Canonical Representation"	"2:03:04 PM 13-Aug-2005 EST"	"2005-08-13T19:03:04Z"
base64Binary	XML Schema "Lexical Representation"	(no Localized Form)	"ZWcgaW/hZ+UuLi4="

N.10.2 Service Parameters

Web service toolkits (software libraries) typically provide "language bindings" that provide a mapping between the native formats of data values in memory and the encoded format used on the wire in a Web service call.

Many of the services defined by this standard have service parameters (function arguments and return values) that are polymorphic. For example, the same service can be used to return a ValueAge attribute, which is of datatype double, and a Writable attribute, which is of datatype boolean. To accomplish this polymorphism without using complex datatypes on the wire, the Web service method signatures of these services defines these parameters to be the XML Schema datatype "string".

Because these polymorphic service parameters are all declared to be of XML Schema datatype "string", the language bindings will bind all of these parameters to the native representation of a character string.

The information in this standard, combined with the information provided by the ValueType attribute, together give the client all the information it needs to unambiguously map between a polymorphic service parameter and a native format.

The mapping between the canonical form of an attribute value and a polymorphic service parameter string follows the rules defined by the XML Schema standard for encoding datatypes for use in XML instance documents. The result of following these rules is simply that the same sequence of characters is sent on the wire for a polymorphic parameter as would be sent if that parameter had been declared to be of the specific datatype being encoded.

For example: The "Start" service parameter of the getHistoryPeriodic service is declared with a specific XML Schema datatype of "dateTime". The characters sent on the wire for this parameter would be in the form "2004-06-27T19:44Z". In contrast, the return parameter for the getValue service is declared to be an XML Schema datatype of "string". However, if the getValue service is used to read the Value attribute for a node whose ValueType attribute is "DateTime", the characters sent on the wire for the return parameter would also be in the form "2004-06-27T19:44Z".

The mechanism for, and the configuration of, the mapping between the non-canonical (localized) form of an attribute value and a polymorphic service parameter string, such as localized date formats, is a local matter.

N.11 Service Options

Some services accept service options that modify their behavior or their return values.

Individual options are specified in string form as simply "option-name" or "option-name=option-value". For example, "readback", or "locale=en-UK". When multiple options are combined into a single string, they are separated by a semicolon, such as "readback;locale=en-UK". White space is significant and shall not be stripped during parsing. The option-value is not constrained with the exception that it shall not contain a semicolon.

The '=' character and option-value may be omitted for boolean options. If a boolean option name is present without an option-value, then it assumes the value "true". Options with a default value of "true" will have to be explicitly set to "false". If an option-name is specified more than once in the string, the last one takes precedence.

The strings used for option-name and option-value are not subject to the effects of the "locale" and "canonical" options. The option names are from the fixed set defined in this standard. The "Datatype" referred to in the following table is the XML Schema datatype name. This datatype defines the canonical format for the option value when represented as a string.

Table N-5. Service Options

Option Name	Datatype	Default if Not Specified
"readback"	boolean	False
"errorString"	string	(see Clause N.13)
"errorPrefix"	string	empty string
"locale"	string	varies based on server configuration
"writeSingleLocale"	boolean	false
"canonical"	boolean	False
"precision"	nonNegativeInteger	6
"noEmptyArrays"	boolean	False

N.11.1 readback

This option causes services that set a value or values to attempt to read back the value or values just written and return the results.

N.11.2 errorString

This option specifies the string to be returned for errors rather than the default format defined by Clause N.13.

Changing the error string may simplify client calculations or presentations. For example, if the client requires "-1" to be returned for errors to aid in some numerical calculations, it would specify a service option of "errorString=-1". If the client is filling a report and wants blank strings returned for errors, it would specify a service option of "errorString=".

N.11.3 errorPrefix

This option specifies the string to be returned in front of the default format defined by Clause N.13. Changing the error prefix may be desired if the default format could possibly conflict with a real value. Whereas the errorString service option is intended to define the entire contents of the error string, the errorPrefix merely prefixes the default format to allow clients to get the original error information in addition to a customized prefix. If both errorString and errorPrefix are specified, the resultant error string is the errorPrefix followed by the errorString.

N.11.4 locale

This option specifies the locale that shall be used for formatting of date/time values, units, numbers and string values by the server. The format of the locale option is: "locale=language-tag", where language-tag is in the form described by RFC 3066. For example, the locale string for US English is "en-US", and Canadian French is "fr-CA", and the corresponding service options would be formatted as "locale=en-US" and "locale=fr-CA".

The value of the locale service option must match exactly one of the strings returned from the getSupportedLocales service. There is no language fallback or hierarchical matching mechanism.

In services which read data from a node such as the getValue, getValues, or getArray services, the server is required to accept all values for the “locale” option which are returned by the getSupportedLocales service When writing data to a node with services such as the setValue or setValues services, the server shall accept all values for the “locale” option which are returned in the WritableLocales attribute of the node. The error WS_ERR_LOCALE_NOT_SUPPORTED shall be returned if a locale is specified that the server does not support.

The values available in the WritableLocales attribute of a node shall be a subset of the values returned by the getSupportedLocales service.

A server shall be configurable to associate a date, time and numeric formats with each locale. When a localized value is requested, the server shall return the string formatted according to the format for the specified locale. For example, a server should be able to support localized time and date formats such as "2004/06/15 8:00am" or "15-Jun-2004, 08:00:00" and numeric formats such as "1,234.56" or "1 234,56". This will help to ensure that all servers used within an installation will be capable of presenting data in a consistent manner.

In some cases, the "locale" option may be overridden by the "canonical" option. This is described in Clause N.11.6.

N.11.5 writeSingleLocale

This option applies only to setting the values for nodes with a ValueType of "String". The default behavior of a server is to set the value for the Value attribute in all locales, regardless of the "locale" service option. This is safer than setting only one locale because the client might not be aware of which locales are in use, and setting only one might lead to inconsistent values across locales. For clients that are aware of the different locales and want to set different values for the different locales, this service option allows the client to override this default behavior and write only one locale at a time.

If this option is true, then the locale service option, if present, shall be equal to one of the locales listed in the SinglyWritableLocales attribute for the node being written, otherwise, an invalid locale error is returned.

If this option is true and no "locale" option is specified, then string values are set only in the default locale. If the default locale is not one of those listed in the SinglyWritableLocales for the node being written, then an invalid locale error is returned.

N.11.6 canonical

This option is intended to override certain localized string formats. The "canonical form" is a locale-independent standardized form, as defined in Clause N.10.1, that can be parsed in a consistent manner when node values are intended to be processed by machine rather than to be presented to humans.

The interaction between the "locale" and "canonical" options is summarized in the following table. Attributes not listed in this table are not affected.

Table N-6. Locale and Canonical Options

Attribute Name	Effect of "locale"	Effect of "canonical"
Value, when ValueType is "String"	The server may return and accept different values for different locales. For reading, server shall use the "locale" option to select the returned value. For writing, the "locale" option is ignored (all locales are written) unless the "writeSingleLocale" option is true.	Ignored.
Value, when ValueType is "Multistate"	The server may return and accept different values for different locales. For any given locale, these values shall be one of the values returned for the "PossibleValues" attribute for that locale.	Ignored.
Value, when ValueType is "Real", "Integer", "DateTime", "Date", "Time", "Duration", or "Boolean"	Value is formatted according to a server configuration to be appropriate to the requested locale.	Overrides "locale". The format is defined in N.10.1.
Value, when ValueType is "OctetString"	Ignored.	Ignored.
DisplayName	May return different values for different locales.	Ignored.
PossibleValues	May return different values for different locales.	Ignored.
WritableValues	May return different values for different locales.	Ignored.
Units	Value is formatted according to a server configuration to be appropriate to the requested locale.	Overrides "locale". The format is defined in N.8.11.
Description	May return different values for different locales.	Ignored.
ValueAge, Minimum, Maximum, and Resolution	Value is formatted according to a server configuration to be appropriate to the requested locale.	Overrides "locale". The format is defined in N.10.1.

N.11.7 precision

This option specifies the number of digits after the decimal point for the floating point value of any requested attribute. The value shall be rounded, not truncated. For example, "precision=2" makes "123.45673" into "123.46". This applies to fractional seconds in time-related values as well.

N.11.8 noEmptyArrays

This option specifies that the server should not return empty arrays, and should return an error instead. This is primarily for Web services language bindings that do not correctly process arrays with no elements in them.

N.12 Services

This clause defines the Web services that provide the means to access and manipulate the data in the server.

N.12.1 getValue Service

This required service is used to retrieve a single value for a single attribute of a single node. This service always returns its results as a single string.

This service can be used to retrieve primitive attributes, such as Value, and array attributes, such as PossibleValues. The format of this string result is dictated by the attribute's datatype and the service options.

If this service is used for an array attribute, then the array elements shall be concatenated into a single semicolon-delimited string that can be easily split at the client since the element strings are not allowed to contain semicolon characters. If the client would rather retrieve an array of individual strings, it can use the getArray or getArrayRange service instead.

A typical programming language signature for this service is:

CString getValue(CString options, CString path)

N.12.1.1 Structure

The structure of the getValue service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-7. Structure of getValue Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Path	M	M(=)		
Result			M	M(=)

N.12.1.2 Argument

This parameter shall convey the parameters for the getValue confirmed service request.

N.12.1.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.1.2.2 Path

This parameter, of type XML Schema string, shall contain a path as defined in Clause N.2.

N.12.1.3 Result

This parameter, of type XML Schema string, shall contain the results of the service call. This parameter is polymorphically encoded, as defined in Clause N.10.2. The result shall be either a valid value or an error string. The format of error strings is defined by Clause N.13.

N.12.1.4 Service Procedure

The service will attempt to find the node and attribute specified by the Path parameter, and if successful, shall format its value into a string according to the rules specified in Clauses N.8.10, N.10.1, and N.11. If the Path parameter refers to an array attribute, then the formatted string representations of the individual elements are concatenated into a single string using the semicolon (';') character as the delimiter between elements. If an attribute identifier is not specified by the Path parameter, the Value attribute is assumed.

The getValue service, and all the various "get" methods, are allowed to return a result without consulting any other network node, either because the data is cached or because the origin if the data is the server itself. If the server, for any internal reason, is unable to return a value according to its normal means of execution, then the result returned shall be WS_ERR_OTHER. If for an external reason, the server is unable to contact an external source of the data according to its normal means of execution, then the result returned shall be WS_ERR_COMMUNICATION_FAILED. This will be typical when, for example, the server attempts to establish communication with the device serving the data, and that device fails to respond.

The error conditions and responses are summarized in the following table:

Table N-8. Error Conditions for the getValue Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option.	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the Path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
Communication with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to return the requested value, for some other reason.	WS_ERR_OTHER

N.12.2 getValues Service

This optional service is similar to the getValue service with the exception that it takes multiple paths and returns multiple results, one for each path. This service always returns its results as a non-empty array of strings.

A typical programming language signature for this service is:

CString[] getValues(CString options, CString paths[])

N.12.2.1 Structure

The structure of the getValues service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-9. Structure of getValues Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Paths	M	M(=)		
Result			M	M(=)

N.12.2.2 Argument

This parameter shall convey the parameters for the getValues confirmed service request.

N.12.2.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.2.2.2 Paths

This parameter, of type array of XML Schema string, shall contain an array of path strings as defined in Clause N.2.

N.12.2.3 Result

This parameter, of type array of XML Schema string, shall contain the results of the service call. Each entry in the array is either a valid value or an error string. Each entry is polymorphically encoded, as defined in Clause N.10.2. The format of error strings is defined by Clause N.13.

N.12.2.4 Service Procedure

This service will process the entries in the Paths parameter starting with the first entry in the array. Each entry is evaluated separately in the same manner as the getValue service and the results of that evaluation are entered into the corresponding entry in the return array. If there is an error condition that prevents the processing of the Paths parameter, if the Paths parameter is of zero length, or if the server can determine that the same error would be returned for each entry in the return array, then the result of the service shall be an array of one element containing the error string.

The error conditions and responses are summarized in the following table:

Table N-10. Error Conditions for the getValues Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The Paths parameter array has no members.	WS_ERR_LIST_OF_PATHS_IS_EMPTY
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
Communication with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to return the requested value, for some other reason.	WS_ERR_OTHER

N.12.3 getRelativeValues Service

This optional service is similar to the getValues service with the exception that it takes a single base path that specifies a node or attribute, and a list of additional sub paths that are appended to the base path to form a complete path. A typical use of this service would be for the base path to represent a path to a node and the sub paths to be a list of attributes, but the service is not limited to that usage. This service always returns its results as a non-empty array of strings.

A typical programming language signature for this service is:

CString[] getRelativeValues(CString options, CString basePath, CString paths[])

N.12.3.1 Structure

The structure of the getRelativeValues service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-11. Structure of getRelativeValues Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Base Path	M	M(=)		
Paths	M	M(=)		
Result			M	M(=)

N.12.3.2 Argument

This parameter shall convey the parameters for the getRelativeValues confirmed service request.

N.12.3.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.3.2.2 Base Path

This parameter, of type XML Schema string, shall contain either an empty string or a complete and valid path string as defined in Clause N.2, that identifies a node or attribute. This path shall end with a node identifier or an attribute identifier, not a path delimiter ('/' or ':'). If this parameter is an empty string, then each of the paths in the Paths parameter becomes the full path for evaluation.

N.12.3.2.3 Paths

This parameter, of type array of XML Schema string, shall contain an array of path fragments that when appended to the Base Path parameter form a complete and valid path as defined in Clause N.2 . Since the Base Path parameter does not not end with a delimiter, and may be empty, these path fragments shall begin with a delimiter ('/' or ':') in order to form a complete path.

N.12.3.3 Result

This parameter, of type array of XML Schema string, shall contain the results of the service call. Each entry in the array is either a valid value or an error string. Each entry is polymorphically encoded, as defined in Clause N.10.2. The format of error strings is defined by Clause N.13.

N.12.3.4 Service Procedure

This service will process the entries in the Paths parameter starting with the first entry in the array. Each entry is evaluated separately in the same manner as if the getValue service were called with a path equal to the Base Path parameter concatenated with the entry being processed, and the results of that evaluation are entered into the corresponding entry in the return array.

The error conditions and responses are summarized in the following table:

Table N-12. Error Conditions for the getRelativeValues Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The Paths parameter array has no members.	WS_ERR_LIST_OF_PATHS_IS_EMPTY
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
Communication with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to return the requested value, for some other reason.	WS_ERR_OTHER

N.12.4 getArray Service

This optional service can be used to retrieve array attributes such as Children or PossibleValues as an array of strings rather than as a single concatenated string. The format of the strings in the array is dictated by the attribute's datatype and the service options. This service shall not be used on attributes that are not arrays. If the entire array is too large to return with this service, the client can use multiple calls to the getArrayRange service instead.

If this service is provided, then the getArraySize service shall also be provided. This service is required to be provided if the getArrayRange service is provided.

A typical programming language signature for this service would be:

CString[] getArray(CString options, CString path)

N.12.4.1 Structure

The structure of the getArray service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-13. Structure of getArray Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Path	M	M(=)		
Result			M	M(=)

N.12.4.2 Argument

This parameter shall convey the parameters for the getArray confirmed service request.

N.12.4.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.4.2.2 Paths

This parameter, of type XML Schema string, shall contain a path string as defined in Clause N.2.

N.12.4.3 Result

This parameter, of type array of XML Schema string, shall contain the results of the service call. If the service succeeds, the result will be an array of valid result strings. Each entry is polymorphically encoded, as defined in Clause N.10.2. If the array attribute has no members, the result array shall be empty unless the noEmptyArrays service option is true, in which case the result array shall contain a single entry for the WS_ERR_EMPTY_ARRAY error condition. If the service fails, the result will be an array containing a single entry containing the error string. The format of error strings is defined by Clause N.13.

N.12.4.4 Service Procedure

The service will attempt to find the node and attribute specified by the Path parameter, and if successful, will format its value into an array of strings according to the rules specified in Clauses N.8.10, N.10.1, and N.11.

The error conditions and responses are summarized in the following table:

Table N-14. Error Conditions for the getArray Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option.	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the Path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
The requested array contains no data (the array size is 0) and the noEmptyArrays service option is true.	WS_ERR_EMPTY_ARRAY
The attribute specified in the Path parameter is not an array attribute.	WS_ERR_NOT_AN_ARRAY
Communication with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to return the requested value, for some other reason.	WS_ERR_OTHER

If any errors occur, the result of the service shall be an array of one entry containing the error string.

N.12.5 getArrayRange Service

This optional service can be used to retrieve only a portion of an array attribute such as Children or PossibleValues as an array of strings. The format of the strings in the array is dictated by the attribute's datatype and the service options. This service shall not be used on attributes that are not arrays.

If this service is provided, then the getArray and getArraySize service shall also be provided.

A typical programming language signature for this service would be:

CString[] getArrayRange(CString options, CString path, unsigned index, unsigned count)

N.12.5.1 Structure

The structure of the getArrayRange service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-15. Structure of getArrayRange Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Path	M	M(=)		
Index	M	M(=)		
Count	M	M(=)		
Result			M	M(=)

N.12.5.2 Argument

This parameter shall convey the parameters for the getArrayRange confirmed service request.

N.12.5.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.5.2.2 Path

This parameter, of type XML Schema string, shall contain a path string as defined in Clause N.2.

N.12.5.2.3 Index

This parameter, of type XML Schema nonNegativeInteger, shall contain the starting index, where the first entry in the array is index zero.

N.12.5.2.4 Count

This parameter, of type XML Schema nonNegativeInteger, shall contain the number of array entries to return, starting at the Index parameter. A count of zero shall be invalid.

N.12.5.3 Result

This parameter, of type array of XML Schema string, shall contain the results of the service call. If the service succeeds, the result shall be an array of valid result strings. Each entry is polymorphically encoded, as defined in Clause N.10.2. If the service fails, the result shall be an array containing a single entry containing the error string. The format of error strings is defined by Clause N.13.

N.12.5.4 Service Procedure

The service shall attempt to find the node and attribute specified by the Path parameter, and if successful, shall format its value into an array of strings according to the rules specified in Clauses N.8.10, N.10.1, and N.11, starting at the index specified by the Index parameter and proceeding for the number of entries specified by the Count parameter. If fewer than the specified count of entries exist after the specified index, the result array shall be truncated to contain only the valid entries.

The error conditions and responses are summarized in the following table:

Table N-16. Error Conditions for the getArrayRange Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option.	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the Path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
The index parameter is outside the range of indices for the specified attribute.	WS_ERR_INDEX_OUT_OF_RANGE
The count parameter is zero.	WS_ERR_COUNT_IS_ZERO
The requested array range contains no data (the result is of zero length) and the noEmptyArrays service option is true.	WS_ERR_EMPTY_ARRAY
The attribute specified in the Path parameter is not an array attribute.	WS_ERR_NOT_AN_ARRAY
Communication with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to return the requested value, for some other reason.	WS_ERR_OTHER

If any errors occur, the result of the service shall be an array of one entry containing the error string.

N.12.6 getArraySize Service

This optional service can be used to retrieve the number of entries in an array attribute. This service shall not be used for attributes that are not arrays.

This service is required to be provided if the getArray service is provided.

A typical programming language signature for this service is:

CString getArraySize(CString options, CString path)

N.12.6.1 Structure

The structure of the getArraySize service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-17. Structure of getArraySize Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Path	M	M(=)		
Result			M	M(=)

N.12.6.2 Argument

This parameter shall convey the parameters for the getArraySize confirmed service request.

N.12.6.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.6.2.2 Paths

This parameter, of type XML Schema string, shall contain a path string as defined in Clause N.2.

N.12.6.3 Result

This parameter, of type XML Schema string, shall contain the results of the service call. If the service succeeds, the result shall be an XML Schema nonNegativeInteger. This parameter is polymorphically encoded, as defined by Clause N.10.2. If the service fails, the result shall contain the error string. The format of error strings is defined by Clause N.13.

N.12.6.4 Service Procedure

The service shall attempt to find the node and attribute specified by the Path parameter, and if successful, shall return the number of entries in that array attribute.

The error conditions and responses are summarized in the following table:

Table N-18. Error Conditions for the getArraySize Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option.	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the Path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
The attribute specified in the Path parameter is not an array attribute.	WS_ERR_NOT_AN_ARRAY
Communication with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to return the requested value, for some other reason.	WS_ERR_OTHER

N.12.7 setValue Service

This optional service is used to set a new value for a single attribute of a single node. The format of the new value is dictated by the attribute's datatype and the service options. This service always returns its results as a single string.

If the service option "readback" is true, then, after setting the value, this service shall read the value back and the result shall be as if the client had called getValue using the same path and service options. This allows the client to see the effects of any value modification by the server as well as check for errors.

Only the Value attribute is writable.

This service is required to be provided if the setValues service is provided.

A typical programming language signature for this service is:

CString setValue(CString options, CString path, CString Value)

N.12.7.1 Structure

The structure of the setValue service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-19. Structure of setValue Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Path	M	M(=)		
Value	M	M(=)		
Result			M	M(=)

N.12.7.2 Argument

This parameter shall convey the parameters for the setValue confirmed service request.

N.12.7.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.7.2.2 Path

This parameter, of type XML Schema string, shall contain a path as defined in Clause N.2.

N.12.7.2.3 Value

This parameter, of type XML Schema string, shall contain a new value for the node. This parameter is polymorphically encoded, as defined in Clause N.10.2, and is in the same format as that which would be returned by the getValue service for the same path and service options.

N.12.7.3 Result

This parameter, of type XML Schema string, shall contain the results of the service call. The result is either an empty string, a valid value if the "readback" service option is true, or an error string. This parameter is polymorphically encoded, as defined in Clause N.10.2. The format of error strings is defined by Clause N.13.

N.12.7.4 Service Procedure

The service shall attempt to find the node and attribute specified by the Path parameter, and if successful, shall set its value from the given Value parameter according to the formatting rules specified in Clauses N.8.10, N.10.1, and N.11. If an attribute identifier is not specified by the Path parameter, the Value attribute shall be assumed.

If the server supports multiple locales and this service is used to set the value of a node whose ValueType attribute is "String", then the new value shall be set equally for all writable locales unless the "writeSingleLocale" service option is true, in which case it shall be set only for the locale specified by the "locale" service option. See the definitions for the SinglyWritableLocales attribute and the writeSingleLocale service option in clauses N.8.29 and N.11.5 for more information.

If the server supports multiple locales and this service is used to set the value of a node whose ValueType attribute is "Multistate", then the Value parameter shall match exactly one of the strings returned for the WritableValues attribute for the locale specified by the service options.

If multiple locales are supported by the server and this service is used to set the value of a node whose ValueType attribute is "Boolean", then the new value shall match exactly one of the strings returned for the WritableValues attribute

for the locale specified by the service options, or it may be equal to "true" or "false" if the "canonical" service option is TRUE.

If the service option "readback" is true, then, after setting the value, the server shall perform the same operations as prescribed for the getValue service, using the same path and service options. If there is any failure during the readback portion of execution, then the result returned by setValue shall be WS_ERR_READBACK_FAILED.

If the service option "readback" is false, then this service shall return an empty string upon success.

The error conditions and responses are summarized in the following table:

Table N-20. Error Conditions for the setValue Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option.	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the Path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
An attribute other than Value is specified.	WS_ERR_ILLEGAL_ATTRIBUTE
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
The Value attribute is not writable.	WS_ERR_NOT_WRITABLE
The given value is not formatted properly.	WS_ERR_VALUE_FORMAT
The given value is out of range.	WS_ERR_VALUE_OUT_OF_RANGE
Any other error occurred setting the value.	WS_ERR_WRITE_FAILED
The readback failed.	WS_ERR_READBACK_FAILED
Communication with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to update the requested value, for some other reason.	WS_ERR_OTHER

N.12.8 setValues Service

This optional service is similar to the setValue service with the exception that it takes multiple paths and values and returns multiple results, one for each path. This service always returns its results as a non-empty array of strings.

If this service is provided, then the setValue service shall also be provided.

A typical programming language signature for this service is:

CString[] setValues(CString options, CString paths[], CString values[])

N.12.8.1 Structure

The structure of the setValues service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-21. Structure of setValues Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Paths	M	M(=)		
Values				
Result			M	M(=)

N.12.8.2 Argument

This parameter shall convey the parameters for the setValues confirmed service request.

N.12.8.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.8.2.2 Paths

This parameter, of type array of XML Schema string, shall contain an array of path strings as defined in Clause N.2.

N.12.8.2.3 Values

This parameter, of type array of XML Schema string, shall contain an array of new values corresponding to the Paths parameter. Each entry in this array shall be polymorphically encoded, as defined in Clause N.10.2, and shall have the same format as that which would be returned by the getValue service for the corresponding path with the same service options.

N.12.8.3 Result

This parameter, of type array of XML Schema string, shall contain the results of the service call. Each entry in the array is either an empty string, a valid value if the "readback" service option is true, or an error string. Each entry is polymorphically encoded, as defined in Clause N.10.2. The format of error strings is defined by Clause N.13.

N.12.8.4 Service Procedure

This service will process the entries in the Paths parameter and the corresponding entries in the Values parameter, starting with the first entry in each array. Each pair of entries shall be evaluated separately in the same manner as the setValue service and the results entered into a corresponding entry in the return array. If there is an error condition that prevents the processing of the Paths parameter, if the Paths parameter is of zero length, or if the server can determine that the same error would be returned for each entry in the return array, then the result of the service shall be an array of one element containing the error string.

The error conditions and responses are summarized in the following table:

Table N-22. Error Conditions for the setValues Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option.	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The Paths parameter array has no members.	WS_ERR_LIST_OF_PATHS_IS_EMPTY
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the Path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
An attribute other than Value is specified.	WS_ERR_ILLEGAL_ATTRIBUTE
The Value attribute is not writable.	WS_ERR_NOT_WRITABLE
The given value is not formatted properly.	WS_ERR_VALUE_FORMAT
The given value is out of range.	WS_ERR_VALUE_OUT_OF_RANGE
Any other error occurred setting the value.	WS_ERR_WRITE_FAILED
The readback failed.	WS_ERR_READBACK_FAILED
Communications with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to update the requested value, for some other reason.	WS_ERR_OTHER

N.12.9 getHistoryPeriodic

This optional service returns a predictable result of periodic point-in-time trend samples. Each string in the array contains the trended value or an error string in the same format as would be returned from the getValue service for the same path and service options.

The client specifies the sampling for this trend series, regardless of the sampling rate or timestamps of the data stored in the historical records of the server. If there is a mismatch in the requested sample times and the actual sample times, the server shall resample the data, as requested by the client through the Resample Method parameter, to find a value for the requested sample time.

The first sample returned corresponds to the Start parameter, and the remaining samples are spaced apart according to the Interval parameter. The Count parameter specifies the total number of samples to return.

A typical programming language signature for this service is:

CString[] getHistoryPeriodic (CString options, CString path, CDateTime start, double interval, unsigned count, CString resampleMethod)

N.12.9.1 Structure

The structure of the getHistoryPeriodic service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-23. Structure of getHistoryPeriodic Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Path	M	M(=)		
Start	M	M(=)		
Interval	M	M(=)		
Count	M	M(=)		
Resample Method	M	M(=)		
Result			M	M(=)

N.12.9.2 Argument

This parameter shall convey the parameters for the getHistoryPeriodic confirmed service request.

N.12.9.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.9.2.2 Path

This parameter, of type XML Schema string, shall contain a path string as defined in Clause N.2.

N.12.9.2.3 Start

This parameter, of type XML Schema dateTime, shall specify the starting date and time, inclusive, for the results.

N.12.9.2.4 Interval

This parameter, of type XML Schema double, shall specify the time interval, in seconds, between the returned values. An interval of zero is invalid.

N.12.9.2.5 Count

This parameter, of type XML Schema nonNegativeInteger, shall contain the number of values to return. A count of zero is invalid.

N.12.9.2.6 Resample Method

This parameter, of type XML Schema string, shall contain one of the string values described in the following table. Servers shall support all standard resample methods.

Table N-24. getHistoryPeriodic Resample Method Definitions

Parameter Value	Description
"interpolation"	Each data sample returned is determined by straight line interpolation between the real sample before and the real sample after the specified point in time. If the source Trend Log has a fixed interval and one of the real samples is missing then an error shall be returned for the sample. If one of the real samples is an error then an error shall be returned for the sample.
"average"	Each data sample returned is the average of all collected samples within the time period. The time period is of length Interval and is centered on the returned sample time. If all samples are missing from the sample window, then an error shall be returned for the sample. If one or more, but not all, samples are missing from the sample window, the average will be calculated over those that are present.
"after"	Each data sample returned is the value of the closest real sample at or after the specified point in time. If the source Trend Log has a fixed interval and the closest sample after is missing, then an error shall be returned for the sample. If the closest sample after is an error, then an error shall be returned for the sample.
"before"	Each data sample returned is the value of the closest real sample at or before the specified point in time. If the source Trend Log has a fixed interval and the closest sample before is missing, then an error shall be returned for the sample. If the closest sample before is an error, then an error shall be returned for the sample.
"closest"	Each data sample returned is the value of the closest real sample at, before or after the specified point in time. If the source Trend Log has a fixed interval and the closest sample is missing, then an error shall be returned for the sample. If the closest sample is an error, then an error shall be returned for the sample.
"default"	The server shall use the most appropriate resample method. The server is not restricted to the standard resample methods and may use any proprietary method suited to the data.

N.12.9.3 Result

This parameter, of type array of XML Schema string, shall contain the results of the service call. If the service succeeds, the result shall be an array of valid result strings. Each member of the array is polymorphically encoded, as defined in Clause N.10.2. If the service fails, the result shall be an array containing a single entry containing the error string. The format of error strings is defined by Clause N.13.

N.12.9.4 Service Procedure

The service shall attempt to find historical records for the node specified by the Path parameter, and if successful, shall format a series of historical values into an array of strings according to the rules specified in Clauses N.8.10, N.10.1, and

N.11, starting at the date and time specified by the Start parameter, and proceeding in time increments of the Interval parameter, for the number of entries specified by the Count parameter. If an attribute identifier is specified by the Path parameter, it shall specify the Value attribute.

If there is a mismatch in the requested sample times and the actual sample times, the server shall resample the data by some means, such as interpolation, to find a value for the requested sample time. If the data is known to the server to not be available at the requested sample time, it shall return a WS_ERR_NO_DATA_AVAILABLE error for that sample time in the Results array.

If there is an error condition that prevents the retrieval or processing of the requested data, then the result of the service shall be an array of one element containing the error string. If the server can determine that the same error would be returned for each entry in the results array, then the result of the service may be an array of one element containing the error string.

The error conditions and responses are summarized in the following table:

Table N-25. Error Conditions for the getHistoryPeriodic Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains a locale specifier that is not currently supported.	WS_ERR_LOCALE_NOT_SUPPORTED
The Options parameter contains an unsupported option	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The path could not be parsed or contains an illegal character.	WS_ERR_PATH_SYNTAX
The node identified by the Path parameter does not exist.	WS_ERR_NODE_NOT_FOUND
An attribute other than Value is specified.	WS_ERR_ILLEGAL_ATTRIBUTE
The attribute specified in the Path parameter is not present in the specified node.	WS_ERR_ATTRIBUTE_NOT_FOUND
The Count parameter is 0.	WS_ERR_COUNT IS_ZERO
The Interval parameter is 0.	WS_ERR_INTERVAL_IS_ZERO
No data is available for a sample interval.	WS_ERR_NO_DATA_AVAILABLE
There is no history available for this node.	WS_ERR_NO_HISTORY
Communication with the device failed.	WS_ERR_COMMUNICATION_FAILED
Unable to return the requested value, for some other reason.	WS_ERR_OTHER

N.12.10 getDefaultLocale

This required service retrieves the locale that the server has configured for its default locale. The return value is a locale strings as defined in Clause N.11.4. The empty string ("") shall be returned if there is no default locale, in which case the canonical form shall be used for all values.

A typical programming language signature for this service is:

CString getDefaultLocale (CString options)

N.12.10.1 Structure

The structure of the getDefaultLocale service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-26. Structure of getDefaultLocale Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Result			M	M(=)

N.12.10.2 Argument

This parameter shall convey the parameters for the getDefaultLocale confirmed service request.

N.12.10.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.10.3 Result

This parameter, of type XML Schema string, shall contain the results of the service call. If the service succeeds, the result shall be a locale string as defined in Clause N.11.4 or an empty string. If the service fails, the result shall contain the error string. The format of error strings is defined by Clause N.13.

N.12.10.4 Service Procedure

The service shall return the locale string for the configured default locale. The service shall ignore the "locale" service option, if present. The empty string ("") shall be returned if there is no default locale.

The error conditions and responses are summarized in the following table:

Table N-27. Error Conditions for the getDefaultLocale Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains an unsupported option.	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE

N.12.11 getSupportedLocales

This required service can be used to retrieve the list of locales supported by the server. Each entry in the returned array is a locale string as defined in Clause N.11.4. If the server does not support multiple locales, then this service shall return only the default locale. If the server does not support localization, and only uses the canonical form, then an array with no entries shall be returned unless the noEmptyArrays service option is true, in which case the result array shall contain a single entry for the WS_ERR_EMPTY_ARRAY error condition.

A typical programming language signature for this service is:

CString[] getSupportedLocales (CString options)

N.12.11.1 Structure

The structure of the getSupportedLocales service primitives is shown in the following table. The terminology and symbology used in this table are explained in Clause 5.6.

Table N-28. Structure of getSupportedLocales Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Options	M	M(=)		
Result			M	M(=)

N.12.11.2 Argument

This parameter shall convey the parameters for the getSupportedLocales confirmed service request.

N.12.11.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.11.

N.12.11.3 Result

This parameter, of type array of XML Schema string, shall contain the results of the service call. If the service succeeds, the result shall be an array of valid result strings. The result array may be empty unless the noEmptyArrays service option is true, in which case the result array shall contain a single entry for the WS_ERR_EMPTY_ARRAY error condition. If the service fails, the array shall contain a single entry containing the error string. The format of error strings is defined by Clause N.13.

N.12.11.4 Service Procedure

The service shall collect all the locale strings that are in use in the server. If the server does not support multiple locales, then this service shall return only the default locale. The service shall ignore the "locale" service option, if present.

The error conditions and responses are summarized in the following table:

Table N-29. Error Conditions for the getSupportedLocales Service

Situation	Error
The service user could not be authenticated.	WS_ERR_NOT_AUTHENTICATED
The service user is not authorized to perform this function.	WS_ERR_NOT_AUTHORIZED
The Options parameter could not be parsed correctly or had illegal characters.	WS_ERR_OPTIONS_SYNTAX
The Options parameter contains an unsupported option.	WS_ERR_OPTION_NOT_SUPPORTED
The Options parameter contains an option value in an unsupported format.	WS_ERR_OPTION_VALUE_FORMAT
The Options parameter contains an option value that is out of range.	WS_ERR_OPTION_OUT_OF_RANGE
The requested array contains no data (the array size is 0) and the noEmptyArrays service option is true.	WS_ERR_EMPTY_ARRAY

If any error occurs, the result of the service shall be an array of one entry containing the error string.

N.13 Errors

For maximum interoperability with a wide range of clients, these Web services avoid returning complex (constructed) datatypes by returning both valid data and errors in the same result string.

The default error string encoding is "? error-number error-message". More specifically, the string shall be composed of: a question mark character, followed by a single space character, followed by a standardized error number defined in the following table, in decimal form, followed by a single space character, followed by an informative human-readable error message whose content is a local matter.

The default error encoding can be overridden by the client with the "errorString" service option (see Clause N.11.2 for examples). When the default format is overridden by the "errorString" service option, the string defined for the errorString option shall form the entire string generated for an error.

Table N-30. Error Numbers

Error Name	Error Number	Example Error Message
WS_ERR_OTHER	0	"Unspecified Error"
WS_ERR_NOT_AUTHENTICATED	1	"Not Authenticated"
WS_ERR_NOT_AUTHORIZED	2	"Not Authorized"
WS_ERR_OPTIONS_SYNTAX	3	"Bad Options Syntax"
WS_ERR_OPTION_NOT_SUPPORTED	4	"Option Not Supported"
WS_ERR_OPTION_VALUE_FORMAT	5	"Bad Option Value Format"
WS_ERR_OPTION_OUT_OF_RANGE	6	"Option Out of Range"
WS_ERR_LOCALE_NOT_SUPPORTED	7	"Locale Not Supported"
WS_ERR_PATH_SYNTAX	8	"Bad Path Syntax"
WS_ERR_NODE_NOT_FOUND	9	"Node Not Found"
WS_ERR_ATTRIBUTE_NOT_FOUND	10	"Attribute Not Found"
WS_ERR_ILLEGAL_ATTRIBUTE	11	"Illegal Attribute"
WS_ERR_VALUE_FORMAT	12	"Bad Value Format"
WS_ERR_VALUE_OUT_OF_RANGE	13	"Value Out of Range"
WS_ERR_INDEX_OUT_OF_RANGE	14	"Index Out of Range"
WS_ERR_NOT_WRITABLE	15	"Not Writable"
WS_ERR_WRITE_FAILED	16	"Write Failed"
WS_ERR_LIST_OF_PATHS_IS_EMPTY	17	"No Paths Provided "
WS_ERR_COUNT_IS_ZERO	18	"Requested Count is Zero"
WS_ERR_INTERVAL_IS_ZERO	19	"Requested Interval is Zero "
WS_ERR_NO_HISTORY	20	"No History"
WS_ERR_NO_DATA_AVAILABLE	21	"No Data Available"
WS_ERR_EMPTY_ARRAY	22	"Empty Array"
WS_ERR_NOT_AN_ARRAY	23	"Not an Array"
WS_ERR_COMMUNICATION_FAILED	24	"Communication with the Remote Device Failed"
WS_ERR_READBACK_FAILED	25	"The Readback Failed"

N.14 Extending BACnet/WS

The data model defined by this standard can be extended in the following ways:

1. Extended information that might be considered to be a property of a node may be modeled by adding children nodes with a NodeType of "Property". This allows for the extended property data to be arbitrarily complex.
2. Node classification can be extended by local application of the NodeSubtype attribute.
3. Any string value can be used for the localized value of the Units attribute. However, if the corresponding canonical value of the Units attribute cannot be expressed as defined in Clause N.8.11, then the canonical value of that attribute shall be "other".

[Add the following to **Clause 25** in the correct alphabetical order in Standard 135-2004, pp. 448-449.]

IETF RFC 2616 (1999), *Hypertext Transfer Protocol – HTTP/1.1*, Internet Engineering Task Force

IETF RFC 2617 (1999), *HTTP Authentication: Basic and Digest Access Authentication*, Internet Engineering Task Force

IETF RFC 2246 (1999), *The TLS Protocol Version 1.0*, Internet Engineering Task Force

NETSCAPE SSL3 DRAFT302 (1996), *The SSL Protocol Version 3.0*, Netscape Communications

IETF RFC 3066 (2001), *Tags for the Identification of Languages*, Internet Engineering Task Force

W3C (2000), *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web Consortium

W3C (2001), *XML Schema Part 0: Primer*, World Wide Web Consortium

W3C (2001), *XML Schema Part 1: Structures*, World Wide Web Consortium

W3C (2001), *XML Schema Part 2: Datatypes*, World Wide Web Consortium

W3C (2003), *Extensible Markup Language (XML) 1.0 (Second Edition)*, World Wide Web Consortium

WS-I (2004), *WS-I Basic Profile 1.0*, Web Services Interoperability Organization

[Add the following to **Clause 25, Sources for Reference Material**, in the correct alphabetical order in Standard 135-2004, p.449.]

Internet Engineering Task Force, www.ietf.org.

W3C: World Wide Web Consortium, www.w3.org.

WS-I: Web Services Interoperability Organization, www.ws-i.org.

Netscape Communications, www.netscape.com.

ANNEX H - COMBINING BACnet NETWORKS WITH NON-BACnet NETWORKS (NORMATIVE)

[Add new **Clause H.6**, p.562]

H.6 Using BACnet with the BACnet/WS Web Services Interface (Annex N)

This clause provides examples of the correspondence between BACnet/WS node attributes to specific properties of BACnet Objects. For some nodes and attributes, mapping might not be to a BACnet property but rather to a static value or to a function that transforms internal information to a BACnet datatype or concept.

H.6.1 Typical Mappings of BACnet/WS attributes to BACnet Object Properties

The "normalized attributes", as defined by Annex N, are designed to provide an interoperable model of selected data to a Web services client. The following clauses define the correspondence of those attributes with BACnet properties.

H.6.1.1 DisplayName

This attribute may correspond to the BACnet property Object_Name, except that DisplayName values do not need to be unique in the Web services data model.

H.6.1.2 Description

This attribute may correspond to the BACnet property Description.

H.6.1.3 Value and Related Attributes

The mappings for attributes related to the Value attribute, and its ValueType, vary according to BACnet object type, and may correspond as shown in the following table.

Table H-1. Value and Value Related Attribute Mappings to BACnet Object Properties

BACnet Object Type	Value	ValueType	Units	Maximum	Minimum	Resolution
Accumulator	Present_Value	"Integer"	Units	Max_Pres_Value		
Analog Input	Present_Value	"Real"	Units	Max_Pres_Value	Min_Pres_Value	Resolution
Analog Output	Present_Value	"Real"	Units	Max_Pres_Value	Min_Pres_Value	Resolution
Analog Value	Present_Value	"Real"	Units			
Averaging	(varies)	"Real"				
Binary Input	Present_Value	"Boolean"				
Binary Output	Present_Value	"Boolean"				
Binary Value	Present_Value	"Boolean"				
Calendar	Present_Value	"Boolean"				
Command	Present_Value	"Integer"				
Device	System_Status	"Multistate"				
Event Enrollment	Event_State	"Multistate"				
Life Safety Point	Present_Value	"Multistate"				
Life Safety Zone	Present_Value	"Multistate"				
Loop	Present_Value	"Real"	Output_ Units	Maximum_Output	Minimum_Output	
Multistate Input	Present_Value	"Multistate"				
Multistate Output	Present_Value	"Multistate"				
Multistate Value	Present_Value	"Multistate"				
Pulse Converter	Present_Value	"Real"	Units			
Schedule	Present_Value	(varies)				

H.6.1.4 Writable

This attribute may correspond to the PICS conformance statement declaration of writable for the BACnet property to which this node maps, but it may also vary depending on which user is making the Web services request or on other configuration or operational criteria.

H.6.1.5 InAlarm

This boolean attribute may correspond to the IN_ALARM flag in the BACnet property Status_Flags. If the IN_ALARM flag of that property is set to true, then InAlarm shall be true.

H.6.1.6 PossibleValues and WritableValues

The mapping for these attributes varies based on BACnet object type, and may be mapped according to the following table. The WritableValues attribute is always a subset of the PossibleValues attribute.

Table H-2. PossibleValues and WritableValues Attribute Mappings

BACnet Object Type	BACnet Property or Datatype Mapping
Binary Input	Active_Text, Inactive_Text properties
Binary Output	Active_Text, Inactive_Text properties
Binary Value	Active_Text, Inactive_Text properties
Command	Action_Text property
Device	BACnetDeviceStatus enumeration
Event Enrollment	BACnetEventState enumeration
Life Safety Point	BACnetLifeSafetyState enumeration
Life Safety Zone	BACnetLifeSafetyState enumeration
Multistate Input	State_Text property
Multistate Output	State_Text property
Multistate Value	State_Text property
Schedule	(varies)

H.6.1.7 Overridden

This boolean attribute may correspond to the OVERRIDDEN flag in the BACnet property StatusFlags. If the OVERRIDDEN flag of that property is set to true, then Overridden shall be true.

[Add a new entry to **History of Revisions**, p.598]

(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

Protocol		***Summary of Changes to the Standard***
Version	***Revision***	
…	…	**…**
1	5	**Addendum *c* to ANSI/ASHRAE 135-2004** Approved by the ASHRAE Standards Committee September 29, 2006 and by the ASHRAE Board of Directors September 29, 2006; and by the American National Standards Institute October 2, 2006. 1. Add BACnet/WS Web Services Interface.

POLICY STATEMENT DEFINING ASHRAE'S CONCERN FOR THE ENVIRONMENTAL IMPACT OF ITS ACTIVITIES

ASHRAE is concerned with the impact of its members' activities on both the indoor and outdoor environment. ASHRAE's members will strive to minimize any possible deleterious effect on the indoor and outdoor environment of the systems and components in their responsibility while maximizing the beneficial effects these systems provide, consistent with accepted standards and the practical state of the art.

ASHRAE's short-range goal is to ensure that the systems and components within its scope do not impact the indoor and outdoor environment to a greater extent than specified by the standards and guidelines as established by itself and other responsible bodies.

As an ongoing goal, ASHRAE will, through its Standards Committee and extensive technical committee structure, continue to generate up-to-date standards and guidelines where appropriate and adopt, recommend, and promote those new and revised standards developed by other responsible organizations.

Through its *Handbook*, appropriate chapters will contain up-to-date standards and design considerations as the material is systematically revised.

ASHRAE will take the lead with respect to dissemination of environmental information of its primary interest and will seek out and disseminate information from other responsible organizations that is pertinent, as guides to updating standards and guidelines.

The effects of the design and selection of equipment and systems will be considered within the scope of the system's intended use and expected misuse. The disposal of hazardous materials, if any, will also be considered.

ASHRAE's primary concern for environmental impact will be at the site where equipment within ASHRAE's scope operates. However, energy source selection and the possible environmental impact due to the energy source and energy transportation will be considered where possible. Recommendations concerning energy source selection should be made by its members.