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

Public Review Draft

ASHRAE® Standard

Proposed Addendum *c* to Standard 135-2004, *BACnet®—A Data Communication Protocol for Building Automation and Control Networks*

First Public Review (October 2004)
(Draft Shows Proposed Changes to Current Standard)

This draft has been recommended for public review by the responsible project committee. To submit a comment on this proposed addendum, use the comment form and instructions provided with this draft. The draft is subject to modification until it is approved for publication by the Board of Directors and ANSI. Until this time, the current edition of the standard (as modified by any published addenda on the ASHRAE web site) remains in effect. The current edition of any standard may be purchased from the ASHRAE Bookstore @ http://www/ashrae.org or by calling 404-636-8400 or 1-800-727-4723 (for orders in the U.S. or Canada).

This standard is under continuous maintenance. To propose a change to the current standard, use the change submittal form available on the ASHRAE web site @ http://www/ashrae.org.

The appearance of any technical data or editorial material in this public review document does not constitute endorsement, warranty, or guaranty by ASHRAE of any product, service, process, procedure, or design, and ASHRAE expressly disclaims such.

AMERICAN SOCIETY OF HEATING, REFRIGERATING AND AIR-CONDITIONING ENGINEERS, INC.
1791 Tullie Circle, NE · Atlanta GA 30329-2305

[This foreword, the table of contents, and the "rationale" on the following pages are not part of this standard. They are merely informative and do not contain requirements necessary for conformance to the standard. They have not been processed according to the ANSI requirements for a standard and may contain material that has not been subject to public review or a consensus process.]

FOREWORD

The purpose of this addendum is to present a proposed change for public review. These modifications are the result of change proposals made pursuant to the ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135. The proposed changes are summarized below.

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

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-2004 and Addenda is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout. Only this new and deleted text is open to comment as this time. All other material in this addendum is provided for context only and is not open for public review comment except as it relates to the proposed changes.

CONTENTS

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 the following entries to **Clause 25**, pp. 448-449, and place them in alphabetical order]

IETF RFC2616 (1999), *Hypertext Transfer Protocol – HTTP/1.18*, Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2616.

IETF RFC 3066 (2001), *Tags for the Identification of Languages*, Internet Engineering Task Force, http://www.faqs.org/rfcs/rfc3066.html.

W3C (2000), *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web Consortium, http://www.w3.org/TR/SOAP.

W3C (2001), *XML Schema Part 0: Primer*, World Wide Web Consortium, http://www.w3.org/XML/Schema#dev.

W3C (2001), *XML Schema Part 1: Structures*, World Wide Web Consortium, http://www.w3.org/XML/Schema#dev.

W3C (2001), *XML Schema Part 2: Datatypes*, World Wide Web Consortium, http://www.w3.org/XML/Schema#dev.

W3C (2003), *Extensible Markup Language (XML) 1.0 (Second Edition)*, World Wide Web Consortium, http://www.w3.org/TR/REC-xml.

WS-I (2004), *WS-I Basic Profile 1.0*, Web Services Interoperability Organization, www.ws-i.org/Profiles/BasicProfile-1.0.html.

[Add to the following entries to **Clause 25**, **Sources for Reference Material**, p.449, and place them in alphabetical order]

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.

[Insert 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 annex 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/1.18* (RFC2616) and encodes the data for transport using *Extensible Markup Language (XML) 1.0 (Second Edition)*, which uses the datatypes and canonical encodings defined by the World Wide Web Consortium XML Schema.

There are three distinct usages of datatype names in this annex. 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 used to store information in a BACnet/WS server are a local matter. In order to exchange that information using Web services, this annex 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", "Integer", "Multistate", "Boolean", "Real", "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.

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.

N.2 Paths

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 separated by forward slash characters.

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

Paths and attribute names are case sensitive.

The concatenated path form is: /identifier[/identifier]…[:attribute-name]

Examples: "/a" "/a/b" "/a/b/c:Description"

All identifiers must be of non-zero length. A path with no identifier ("/") refers to the root of the hierarchy, and "/:attribute-name" 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: / \ : ; | < > * ? " [ ] { }

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 annex 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 node type 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 be not 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." A reference node reflects most of the attributes of the node to which it refers (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.

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.

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

A BACnet/WS server may support multiple locales simultaneously, and several of the attributes of a node are accessible for different locales. 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 annex specifies the use of a transport authentication defined by other standards. Authentication may be provided by a simple HTTP username and password, or may be secured through SSL/TLS certificates or more advanced options such as WS-Security/SAML and others yet to be defined. To ensure a base level of interoperability, this annex requires that all servers be capable of supporting the HTTP "basic" authentication, which is supported by all major Web services toolkits and application servers.

Once a user is authenticated to a server, the authorization as to what that user can access and/or modify is according to authorization policies in the server. The configuration of these authorization policies is a local matter.

N.7 Sessions

The Web services defined by this annex are stateless. These are no sessions established 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 annex.

N.8.1 Primitive Attributes

The datatype of a primitive attribute in this annex is defined using its XML Schema datatype name, such as "boolean", "nonNegativeInteger", and "double". See Clause N.9 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 annex. Additionally, some enumerated attributes are localizable. In that case, the "canonical" set of values is defined by this annex, 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 annex is defined using its XML Schema datatype name, such as "boolean", "nonNegativeInteger", and "double". See Clause N.9 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 must 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.9 for more information on encoding for Web services. Attributes that are not listed as Localizable are never affected by the "locale" service option and are always encoded in their canonical form.

Table N-1. Attribute Summary

Attribute Name	Datatype	Array	Enumerated	Localizable	Presence
NodeType	string	No	Yes	No	Required
NodeSubtype	string	No	No	Yes	Optional
Attributes	string	Yes	No	No	Required
ValueType	string	No	Yes	No	Required
Value	(varies - see N.8.9)	No	No	Yes	Required if ValueType is not "None"
Children	string	Yes	No	No	Optional
Aliases	string	Yes	No	No	Required if there are reference nodes referring to this node (see Clause N.4)
Reference	string	No	No	No	Present if and only if the node is a reference node (see Clause N.4)
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
Description	string	No	No	Yes	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
MinimumLength	nonNegativeInteger	No	No	No	Optional
IsMultiLine	boolean	No	No	No	Optional
PossibleValues	string	Yes	No	Yes	Required if ValueType is "Multistate" or "Boolean"
WritableValues	string	Yes	No	Yes	Required if ValueType is "Multistate" or "Boolean" and Writable is true
DisplayName	string	No	No	Yes	Optional
Overridden	boolean	No	No	No	Optional
WritableLocales	string	Yes	No	No	Present if and only if ValueType is "String" and Writable is true
ValueAge	duration	No	No	Yes	Optional
HasHistory	boolean	No	No	No	Required if ValueType is not "None"

N.8.5 NodeType

This required attribute indicates the general classification of a node. 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", "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 "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. 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", "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 strings values, usually 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 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.9 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
"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	duration
"Time"	time	time	time	duration
"DateTime"	dateTime	dateTime	dateTime	duration
"Duration"	duration	duration	duration	duration

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.10.3.

N.8.11 Units

This optional attribute defines the engineering units for the Value attribute of the node. If the ValueType 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 following strings:

{"meters-per-second-per-second", "square-meters", "square-centimeters", "square-feet", "square-inches", "milliamperes", "amperes", "amperes-per-meter", "amperes-per-square-meter", "ampere-square-meters", "farads", "henrys", "ohms", "ohm-meters", "milliohms", "kilohms", "megohms", "siemens", "siemens-per-meter", "teslas", "volts", "millivolts", "kilovolts", "megavolts", "volt-amperes", "kilovolt-amperes", "megavolt-amperes", "volt-amperes-reactive", "kilovolt-amperes-reactive", "megavolt-amperes-reactive", "volts-per-degree-Kelvin", "volts-per-meter", "degrees-phase", "power-factor", "webers", "joules", "kilojoules", "kilojoules-per-kilogram", "megajoules", "watt-hours", "kilowatt-hours", "megawatt-hours", "btus", "kilo-btus", "mega-btus", "therms", "ton-hours", "joules-per-kilogram-dry-air", "kilojoules-per-kilogram-dry-air", "megajoules-per-kilogram-dry-air", "btus-per-pound-dry-air", "btus-per-pound", "joules-per-degree-Kelvin", "kilojoules-per-degree-Kelvin", "megajoules-per-degree-Kelvin", "joules-per-kilogram-degree-Kelvin", "newtons", "cycles-per-hour", "cycles-per-minute", "hertz", "kilohertz", "megahertz", "per-hour", "grams-of-water-per-kilogram-dry-air", "percent-relative-humidity", "millimeters", "centimeters", "meters", "inches", "feet", "candelas", "candelas-per-square-meter", "watts-per-square-foot", "watts-per-square-meter", "lumens", "luxes", "foot-candles", "kilograms", "pounds-mass", "tons", "grams-per-second", "grams-per-minute", "kilograms-per-second", "kilograms-per-minute", "kilograms-per-hour", "pounds-mass-per-second", "pounds-mass-per-minute", "pounds-mass-per-hour", "tons-per-hour", "milliwatts", "watts", "kilowatts", "megawatts", "btus-per-hour", "kilo-btus-per-hour", "horsepower", "tons-refrigeration", "pascals", "hectopascals", "kilopascals", "millibars", "bars", "pounds-force-per-square-inch", "centimeters-of-water", "inches-of-water", "millimeters-of-mercury", "centimeters-of-mercury", "inches-of-mercury", "degrees-Celsius", "degrees-Kelvin", "degrees-Kelvin-per-hour", "degrees-Kelvin-per-minute", "degrees-Fahrenheit", "degree-days-Celsius", "degree-days-Fahrenheit", "delta-degrees-Fahrenheit", "delta-degrees-Kelvin", "years", "months", "weeks", "days", "hours", "minutes", "seconds", "hundredths-seconds", "milliseconds", "newton-meters", "millimeters-per-second", "millimeters-per-minute", "meters-per-second", "meters-per-minute", "meters-per-hour", "kilometers-per-hour", "feet-per-second", "feet-per-minute", "miles-per-hour", "cubic-feet", "cubic-meters", "imperial-gallons", "liters", "us-gallons", "cubic-feet-per-second", "cubic-feet-per-minute", "cubic-meters-per-second", "cubic-meters-per-minute", "cubic-meters-per-hour", "imperial-gallons-per-minute", "liters-per-second", "liters-per-minute", "liters-per-hour", "us-gallons-per-minute", "degrees-angular", "degrees-Celsius-per-hour", "degrees-Celsius-per-minute", "degrees-Fahrenheit-per-hour", "degrees-Fahrenheit-per-minute", "joule-seconds", "kilograms-per-cubic-meter", "kilowatt-hours-per-square-meter", "kilowatt-hours-per-square-foot", "megajoules-per-square-meter", "megajoules-per-square-foot", "no-units", "newton-seconds", "newtons-per-meter", "parts-per-million", "parts-per-billion", "percent", "percent-obscuration-per-foot", "percent-obscuration-per-meter", "percent-per-second" , "per-year", "per-month", "per-week", "per-day", "per-hour", "per-minute", "per-second", "psi-per-degree-Fahrenheit", "radians", "radians-per-second", "revolutions-per-minute", "square-meters-per-Newton", "watts-per-meter-per-degree-Kelvin", "watts-per-square-meter-degree-Kelvin", "other"}

In the case where the units of the node's value do not map to one of the canonical strings defined here, the localized value may represent the units in a locale specific format, but the canonical value shall be "other".

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 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.

N.8.17 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.8.18 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.8.19 IsMultiLine

This optional attribute indicates that the value of the Value attribute, when the ValueType attribute is equal to "String", is intended to have multiple lines of text.

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 duration 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 path to a child node.

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 node that is not a reference 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 WritableLocales

This optional attribute is an array that contains the collection of writable locales for this node. This attribute is present if and only if ValueType attribute equals "String".

N.9 Encodings

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

N.9.1 Canonical Form

This annex 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 defines encoding rules for these datatypes. A value encoded according to those rules is referred to in this annex as the "canonical form" of that value. All attributes not indicated as "Localizable" in Clause N.8.4 shall always be encoded in their canonical form.

N.9.2 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 annex 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 duration, 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 annex, 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 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.10 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 may 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 annex. 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-3. Service Options

Option Name	Datatype	Default if Not Specified
"readback"	boolean	false
"errorString"	string	(see Clause N.12)
"locale"	string	varies based on server configuration
"writeSingleLocale"	boolean	false
"canonical"	boolean	false
"precision"	nonNegativeInteger	6

N.10.1 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.10.2 errorString

This option specifies the string to be returned for errors rather than the default format defined by Clause N.12.

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.10.3 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=<tags>", where <tags> 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".

This annex defines a means for servers to support multiple locales. Support for multiple locales is optional. A server that supports multiple locales designates one of them to be the "default locale." The default locale is used whenever the "locale" option is not specified. An error of WS_ERR_LOCALE_NOT_SUPPORTED shall be returned if a locale is specified that the server does not support.

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.10.5.

N.10.4 writeSingleLocale

This attribute 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 may not be aware of which locales are in use, and setting only one may 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 specified and no "locale" option is specified, then string values are set only in the default locale.

N.10.5 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.9.1, 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-4. Locale and Canonical Options

Attribute Name	Effect of "locale"	Effect of "canonical"
Value, when ValueType is "String"	For reading, server may return different values for different locales. For writing, locale is ignored (all locales are written) unless the "writeSingleLocale" option is true.	Ignored.
Value, when ValueType is "Multistate"	Server may return and accept different values for different locales. These values will be one of the values returned for the "PossibleValues" attribute for the same locale.	Ignored.
Value, when ValueType is "Real", "Integer", "Datetime", "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.9.1.
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.9.1.
Description	May return different values for different locales.	Ignored.

N.10.6 precision

This option specifies the number of digits after the decimal point for the floating point value of any requested attribute. For example, "precision=2" makes "123.45673" into "123.46". This applies to fractional seconds in time-related values as well.

This option does not affect the return value or values from the setValue or setValues service calls that are invoked without the "readback" option set to true since, in those cases, the return value is equal to the set value without processing.

N.11 Services

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

N.11.1 getValue Service

The simple getValue 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.11.1.1 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-5. Structure of getValue Service Primitives

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

N.11.1.2 Argument

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

N.11.1.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.1.2.2 Path

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

N.11.1.3 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.9.2. The result shall be either a valid value or an error string. The format of error strings is defined by Clause N.12.

N.11.1.4 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.9.1, and N.10. If an attribute identifier is not specified by the Path parameter, the Value attribute is assumed.

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

Table N-6. Error Conditions for the getValue Service

Situation	Error
The authentication of the service user could not be established.	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

N.11.2 getValues Service

This 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.11.2.1 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-7. Structure of getValues Service Primitives

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

N.11.2.2 Argument

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

N.11.2.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.2.2.2 Paths

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

N.11.2.3 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.9.2. The format of error strings is defined by Clause N.12.

N.11.2.4 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 entered into a corresponding entry in the return array.

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

Table N-8. Error Conditions for the getValues Service

Situation	Error
The authentication of the service user could not be established.	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_MISSING_ PARAMETER
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

N.11.3 getArray Service

This 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. This service may 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.

A typical programming language signature for this service would be:

CString[] getArray(CString options, CString path)

N.11.3.1 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-9. Structure of getArray Service Primitives

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

N.11.3.2 Argument

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

N.11.3.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.3.2.2 Paths

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

N.11.3.3 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. 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.12.

N.11.3.4 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.9.1, and N.10.

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

Table N-10. Error Conditions for the getArray Service

Situation	Error
The authentication of the service user could not be established.	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).	WS_ERR_EMPTY_ARRAY
The attribute specified in the Path parameter is not an array attribute.	WS_ERR_NOT_AN_ARRAY

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

N.11.3.5 getArrayRange Service

This service can be used to retrieve only a portion of an array attribute such as Children or PossibleValues as an array of strings. If this service is used for a primitive attribute such as Value, then the primitive attribute shall be treated as an array with a single entry.

A typical programming language signature for this service would be:

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

N.11.3.6 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-11. 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.11.3.7 Argument

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

N.11.3.7.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.3.7.2 Paths

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

N.11.3.7.3 Index

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

N.11.3.7.4 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.11.3.8 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. 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.12.

N.11.3.9 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.9.1, and N.10, starting at the index specified by the Index parameter and proceeding for the number of entries specified by the Count parameter. If the specified attribute is not an array attribute, then it shall be treated as an array attribute with one entry for the purposes of this service. 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-12. Error Conditions for the getArrayRange Service

Situation	Error
The authentication of the service user could not be established.	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 indexes for the specified attribute.	WS_ERR_INDEX_OUT_OF_RANGE
The count parameter is zero.	WS_ERR_PARAMETER_OUT_OF_RANGE
The requested array contains no data (the array size is zero).	WS_ERR_EMPTY _ARRAY
The attribute specified in the Path parameter is not an array attribute.	WS_ERR_NOT_AN_ARRAY

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

N.11.4 getArraySize Service

This 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.

A typical programming language signature for this service is:

CString getArraySize(CString options, CString path)

N.11.4.1 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-13. Structure of getArraySize Service Primitives

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

N.11.4.2 Argument

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

N.11.4.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.4.2.2 Paths

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

N.11.4.3 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.9.2. If the service fails, the result shall contain the error string. The format of error strings is defined by Clause N.12.

N.11.4.4 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-14. Error Conditions for the getArraySize Service

Situation	Error
The authentication of the service user could not be established.	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

N.11.5 setValue Service

The simple setValue 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.

A typical programming language signature for this service is:

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

N.11.5.1 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-15. 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.11.5.2 Argument

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

N.11.5.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.5.2.2 Path

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

N.11.5.2.3 Value

This parameter, of type XML Schema string, shall contain a new value for the path. This parameter is polymorphically encoded, as defined in Clause N.9.2, and is in the same format as that which would be returned by the getValue service for the same path and service options.

N.11.5.3 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.9.2. The format of error strings is defined by Clause N.12.

N.11.5.4 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.9.1, and N.10. If an attribute identifier is not specified by the Path parameter, the Value attribute shall be assumed.

If the server supports multiple locales for the specified node, and the ValueType attribute equals "String", then the new value shall be set 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 service options.

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 must 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, and the result of that operation shall be used as the result of this service.

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-16. Error Conditions for the getValue Service

Situation	Error
The authentication of the service user could not be established.	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 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.	The error returned from the getValue operation.

N.11.6 setValues Service

This service is similar to the setValues 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.

A typical programming language signature for this service is:

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

N.11.6.1 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-17. 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.11.6.2 Argument

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

N.11.6.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.6.2.2 Paths

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

N.11.6.2.3 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.9.2, and shall have the same format as that which would be returned by the getValue service for the same path and service options.

N.11.6.3 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.9.2. The format of error strings is defined by Clause N.12.

N.11.6.4 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.

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

Table N-18. Error Conditions for the getValues Service

Situation	Error
The authentication of the service user could not be established.	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_MISSING_ PARAMETER
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.	The error returned from the getValue operation.

N.11.7 getHistoryPeriodic

This 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 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 an error for that 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, CDuration interval, CUnsigned count)

N.11.7.1 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-19. 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(=)		
Result			M	M(=)

N.11.7.2 Argument

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

N.11.7.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.7.2.2 Paths

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

N.11.7.2.3 Start

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

N.11.7.2.4 Interval

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

N.11.7.2.5 Count

This parameter, of type XML Schema nonNegativeInteger, shall contain the number of values to return.

N.11.7.3 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.9.2. 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.12.

N.11.7.4 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.9.1, and N.10, 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 an error for that sample time.

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

Table N-20. Error Conditions for the getPeriodicHistory Service

Situation	Error
The authentication of the service user could not be established.	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
An attribute other than Value is specified.	WS_ERR_ILLEGAL_ATTRIBUTE
The index parameter or the index parameter plus the count parameter is outside the range of indexes for the specified attribute.	WS_ERR_INDEX_OUT_OF_RANGE
The Count or Interval parameter is 0.	WS_ERR_PARAMETER_OUT_OF_RANGE
No data is available for a single sample interval.	WS_ERR_NO_DATA _AVAILABLE
There is no history available for this node.	WS_ERR_NO_HISTORY

N.11.8 getDefaultLocale

This service retrieves the locale that the server has configured for its default locale. The return values are the locale strings as defined in Clause N.10.3.

A typical programming language signature for this service is:

CString getDefaultLocale (CString options)

N.11.8.1 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-21. Structure of getDefaultLocale Service Primitives

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

N.11.8.2 Argument

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

N.11.8.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.8.3 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.10.3. If the service fails, the result shall contain the error string. The format of error strings is defined by Clause N.12.

N.11.8.4 Service Procedure

The service shall return the locale string for the configured default locale.

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

Table N-22. Error Conditions for the getDefaultLocale Service

Situation	Error
The authentication of the service user could not be established.	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.11.9 getSupportedLocales

This 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.10.3. If the server does not support multiple locales, then this service shall return only the default locale.

A typical programming language signature for this service is:

CString[] getSupportedLocales (CString options)

N.11.9.1 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-23. Structure of getSupportedLocales Service Primitives

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

N.11.9.2 Argument

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

N.11.9.2.1 Options

This parameter, of type XML Schema string, shall contain a string of service options as defined in Clause N.10.

N.11.9.3 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. 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.12.

N.11.9.4 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 error conditions and responses are summarized in the following table:

Table N-24. Error Conditions for the getSupportedLocales Service

Situation	Error
The authentication of the service user could not be established.	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

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

N.12 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 Table N-25, 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.10.2 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-25. Error Numbers

Error Name	Error Number	Example Error Message
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_MISSING_ PARAMETER	17	Missing Parameter
WS_ERR_PARAMETER_OUT_OF_RANGE	18	Parameter out of Range
WS_ERR_NOT_WRITABLE	19	Not Writable
WS_ERR _WRITE_FAILED	20	Write Failed
WS_ERR_NO_HISTORY	21	No History
WS_ERR_NO_DATA _AVAILABLE	22	No Data Available
WS_ERR_EMPTY_ARRAY	23	Empty Array
WS_ERR_NOT_AN_ARRAY	24	Not an Array

N.13 Extending BACnet/WS

The data model defined by this annex 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. The Units enumeration may be extended by configuring the Units attribute to the canonical value of "other", and setting a localized value of the Units attribute to the new units string.

[The following annex title is included here only for generating the table of contents of this document and makes no change to the standard.]

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 M)

This subclause provides examples of the correspondence between BACnet/WS node attributes to specific properties of BACnet Objects. For some nodes and attributes, mapping may not be to any 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 subclauses 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 StatusFlags. 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 Writable 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	BACnetDeviceState 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.