OPC 30030

OPC UA for BACnet

Release 2.00.1

2023-05-17

OPC UA Companion Specification

Standard Type:	Industrial Communications	Comments:	
Title:	OPC UA for BACnet	Date:	2023-05-17
Version:	Release 2.00.1	Software:	MS-Word
Editors:		Source:	OPC 30030 - UA Companion Specification for BACnet 2.00.1.docx
Owner:		Status:	Release

CONTENTS

FIGURES

TABLES

BACNET INTEREST GROUP EUROPE / OPC FOUNDATION

AGREEMENT OF USE

COPYRIGHT RESTRICTIONS

- This document is provided "as is" by the OPC Foundation and the BIG-EU.
- Right of use for the Mapping Document is restricted to the Mapping Document.
- Right of use for the Mapping Document will be granted without cost.
- The Mapping Document may be distributed through computer systems, printed or copied as long as the content remains unchanged and the document is not modified.
- OPC Foundation and BACnet Interest Group do not guarantee usability for any purpose and shall not be made liable for any case using the content of this document.
- The user of the document agrees to indemnify OPC Foundation and BACnet Interest Group Europe and their officers, directors and agents harmless from all demands, claims, actions, losses, damages (including damages from personal injuries), costs and expenses (including attorneys' fees) which are in any way related to activities associated with its use of content from the Mapping Document.
- The document shall not be used in conjunction with company advertising, shall not be sold or licensed to any party.
- The intellectual property and copyright is solely owned by the OPC Foundation and BACnet Interest Group Europe.
- This document includes content from (ANSI/ASHRAE Standard 135-2012 A Data Communication Protocol for Building Automation and Control Networks ISSN 1041-2336) © 2012 ASHRAE, www.ashrae.org. Used by permission. ASHRAE retains its copyright to content from ANSI/ASHRAE Standard 135-2012 when such content is used within this document.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OPC or BACnet specifications may require use of an invention covered by patent rights. OPC Foundation or BACnet Interest Group Europe shall not be responsible for identifying patents for which a license may be required by any OPC or BACnet specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OPC or BACnet specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

WARRANTY AND LIABILITY DISCLAIMERS

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OPC FOUDATION NOR BACnet INTEREST GROUP EUROPE MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OPC FOUNDATION NOR BACnet INTEREST GROUP EUROPE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you.

RESTRICTED RIGHTS LEGEND

This Specification is provided with Restricted Rights. Use, duplication or disclosure by the U.S. government is subject to restrictions as set forth in (a) this Agreement pursuant to DFARs 227.7202-3(a); (b) subparagraph (c)(1)(i) of the Rights in Technical Data and Computer Software clause at DFARs 252.227-7013; or (c) the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 subdivision (c)(1) and (2), as applicable. Contractor / manufacturer are the OPC Foundation, 16101 N. 82nd Street, Suite 3B, Scottsdale, AZ, 85260-1830

COMPLIANCE

The combination of BACnet Interest Group Europe and OPC Foundation shall at all times be the sole entities that may authorize developers, suppliers and sellers of hardware and software to use certification marks, trademarks or other special designations to indicate compliance with these materials as specified within this document. Products developed using this specification may claim compliance or conformance with this specification if and only if the software satisfactorily meets the certification requirements set by BACnet Interest Group Europe or the OPC

Foundation. Products that do not meet these requirements may claim only that the product was based on this specification and must not claim compliance or conformance with this specification.

TRADEMARKS

Most computer and software brand names have trademarks or registered trademarks. The individual trademarks have not been listed here.

GENERAL PROVISIONS

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of Germany.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, this specification.

Revision 2.00.1 Highlights

The following table includes the Mantis issues resolved with this revision.

Mantis ID	Scope	Summary	Resolution
8659	Errata	Spelling Mistakes	Fixed spelling mistakes in Table 144 programm -> program averaring -> averaging unasigned -> unassigned altert-enrollment -> alert-enrollment
8934	Errata	Mandatory InstanceDeclaration Object_Identifier.	Removed InstanceDeclaration Object_Identifier in NodeSet-File from EventAlgorithm of BACnetEventReportingType. No effects to this document.
8466	Clarification	WriteProperty Method on the BACnetDeviceType Object	B.5 was referencing to a Method not defined in the specification. Removed that part of the section.
8973	Errata	Fix AllowSubTypes in structured DataTypes	Some structured DataTypes did not use "AllowSubtypes" although abstract DataTypes where used. Tables and NodeSet need to be updated for DataTypes: BACnetEventParameterChangeOfBitstring BACnetEventParameterChangeOfValue BACnetEventFaultParameterExtended BACnetTimeValueValue BACnetEventParameterExtendedParameters BACnetMessageClass BACnetPriorityValue

OPC UA FOR BACNET

1 Scope

This specification was created by a joint working group of the OPC Foundation and BACnet Interest Group Europe. It defines an OPC UA Information Model to represent the BACnet architectural models.

Typical use cases include but are not limited to SCADA connectivity, enterprise integration of building automation networks and data and interfacing between industry and building automation.

OPC Foundation

The OPC Foundation defines standards for online data exchange between automation systems. They address access to current data (OPC DA), alarms and events (OPC A&E) and historical data (OPC HDA). Those standards are successfully applied in industrial automation.

The new OPC Unified Architecture (OPC UA) unifies the existing standards and brings them to state-of-the-art technology using service-oriented architecture (SOA). Platform-independent technology allows the deployment of OPC UA beyond current OPC applications only running on Windows-based PC systems. OPC UA can also run on embedded systems as well as Linux / UNIX based enterprise systems. The provided information can be generically modelled and therefore arbitrary information models can be provided using OPC UA.

BACnet Interest Group Europe

The BACnet Interest Group Europe (BIG-EU) was founded in 1998 as an association under German law and meanwhile represents BACnet activities all over Europe.

BIG-EU has more than 115 members: vendors, project designer, system integrators and end-users of BACnet systems.

The working group marketing (WG-M) organizes events like forums, trade-shows, etc. while the working group technique supports technical activities like specifications or enhancements of the BACnet standard.

BIG-EU closely collaborates with the ASHRAE/SSPC-135 and BACnet International with the BACnet Testing Laboratories working group BTL-WG.

The BACnet Academy offers BACnet training courses all over Europe.

The BACnet Europe Journal issued twice a year by the BACnet Interest Group Europe is a popular magazine about European BACnet activities.

2 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments and errata) applies.

ANSI/ASHRAE Standard 135-2012 A Data Communication Protocol for Building Automation and Control Networks

ISO16484-5:2012 Building automation and control systems (BACS) – Part 5: Data communication

ANSI/ASHRAE Standard 135.1-2011 Method of Test for Conformance to BACnet

ISO16484-6:2009 Building automation and control systems (BACS) – Part 6: Data communication conformance testing

BTL Implementation Guidelines v34 Document designed for implementers of BACnet. This document enumerates many common mistakes made by first time implementers of BACnet. (available from www.bacnetinternational.org)

Addendum 135-2012al Specify Best Practices for Gateway Design (available from www.bacnet.org/Addenda/index.html)

OPC 10000-1, *OPC Unified Architecture - Part 1: Overview and Concepts*
http://www.opcfoundation.org/UA/Part1/

OPC 10000-2, *OPC Unified Architecture - Part 2: Security Model*
http://www.opcfoundation.org/UA/Part2/

OPC 10000-3, *OPC Unified Architecture - Part 3: Address Space Model*
http://www.opcfoundation.org/UA/Part3/

OPC 10000-4, *OPC Unified Architecture - Part 4: Services*
http://www.opcfoundation.org/UA/Part4/

OPC 10000-5, *OPC Unified Architecture - Part 5: Information Model*
http://www.opcfoundation.org/UA/Part5/

OPC 10000-6, *OPC Unified Architecture - Part 6: Mappings*
http://www.opcfoundation.org/UA/Part6/

OPC 10000-7, *OPC Unified Architecture - Part 7: Profiles*
http://www.opcfoundation.org/UA/Part7/

OPC 10000-8, *OPC Unified Architecture - Part 8: Data Access*
http://www.opcfoundation.org/UA/Part8/

OPC 10000-9, *OPC Unified Architecture - Part 9: Alarms and Conditions*
http://www.opcfoundation.org/UA/Part9/

3 Terms, definitions, and conventions

3.1 Use of terms

Defined terms of OPC UA specifications, types and their components defined in OPC UA specifications and in this specification are highlighted with italic in this document.

BACnet related terms and names are always used together with BACnet.

3.2 Terms used from BACnet standard

3.2.1
Command Prioritization

In building automation systems, multiple entities may control and manipulate an object. There may be a need to arbitrate between a schedule program, manual operation by the facility manager, and from the service technician. For so-called commandable properties (like the present value of output object types), BACnet uses a 16-level prioritization schema where 1 is the highest priority (Manual Life Safety) and 16 is the lowest. The value in the highest priority slot takes precedence over all others. A default value is taken from the property “Relinquish_Default” in case all slots in the priority array are uninitialized (NULL).

3.2.2
Change of Value (COV)

Change-of-Value is used to inform client processes about changes of values spontaneously without the need for polling. For analog data a hysteresis (COV-increment) is provided as a special property to allow reducing the traffic and only notify relevant changes.

Note 1 to entry: The mapping to OPC UA concepts is described in Annex B.

3.3 OPC UA for BACnet Information Model terms

3.3.1 BACnetUaMapper

The software that implements the mapping between BACnet and OPC UA defined in this specification.

3.4 Abbreviations and symbols

A&E	Alarms & Events
ANSI	American National Standards Institute
API	Application Program Interface
ARCNET	Attached Resource Computer NETwork
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
ASN.1	Abstract Syntax Notation One (ISO 8824)
BACnet	Building Automation and Control networks
COV	Change of Value
DA	Data Access
HDA	Historical Data Access
HMI	Human-Machine Interface
IEC	International Electrotechnical Commission
IP	Internet Protocol - RFC 791
ISO	International Organization for Standardization
LAN	Local Area Network
MES	Manufacturing Execution System
NaN	"Not a Number", a unique binary pattern representing an invalid number (see ANSI/IEEE 754-1985)
NAT	Network Address Translation - RFC 2663
PICS	Protocol Implementation Conformance Statement
UA	Unified Architecture
UTC	Universal Time Coordinated
XML	Extensible Markup Language

3.5 Conventions used in this document

3.5.1 Conventions for Node descriptions

Node definitions are specified using tables (See Table 1)

Table 1 – Type Definition Table

Attribute	**Value**				
Attribute name	Attribute value. If it is an optional Attribute that is not set "--" will be used.				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
ReferenceType name	*NodeClass* of the Target*Node*.	*BrowseName* of the target *Node*. If the *Reference* is to be instantiated by the server, then the value of the target Node's BrowseName is "--".	*Attributes* of the referenced *Node*, only applicable for *Variables* and *Objects*.		Referenced *Other* of the referenced *Object*.
Notes – Notes referencing footnotes of the table content.					

Attributes are defined by providing the *Attribute* name and a value, or a description of the value.

References are defined by providing the *ReferenceType* name, the *BrowseName* of the *TargetNode* and its *NodeClass*.

- If the *TargetNode* is a component of the *Node* being defined in the table the *Attributes* of the composed *Node* are defined in the same row of the table. That implies that the referenced *Node* has a *HasModelParent Reference* with the *Node* defined in the Table as *TargetNode* (see OPC 10000-3 for the definition of *ModelParents*).

- The *DataType* is only specified for *Variables*; "[<number>]" indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the *ArrayDimensions* is set as identified by <number> values. If no <number> is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all the *ArrayDimensions* can be omitted. If no brackets are provided, it identifies a scalar *DataType* and the *ValueRank* is set to the corresponding value (see OPC 10000-3). In addition, *ArrayDimensions* is set to null or is omitted. If it can be Any or ScalarOrOneDimension, the value is put into "{<value>}", so either "{Any}" or "{ScalarOrOneDimension}" and the *ValueRank* is set to the corresponding value (see OPC 10000-3) and the *ArrayDimensions* is set to null or is omitted. In Table 2 examples are given.

Table 2 – Examples of DataTypes

Notation	**Data-Type**	**Value-Rank**	**Array-Dimensions**	**Description**
Int32	Int32	-1	omitted or NULL	A scalar Int32
Int32[]	Int32	1	omitted or {0}	Single-dimensional array of Int32 with an unknown size
Int32[][]	Int32	2	omitted or {0,0}	Two-dimensional array of Int32 with unknown sizes for both dimensions
Int32[3][]	Int32	2	{3,0}	Two-dimensional array of Int32 with a size of 3 for the first dimension and an unknown size for the second dimension
Int32[5][3]	Int32	2	{5,3}	Two-dimensional array of Int32 with a size of 5 for the first dimension and a size of 3 for the second dimension
Int32{Any}	Int32	-2	omitted or NULL	An Int32 where it is unknown if it is scalar or array with any number of dimensions
Int32{ScalarOrOneDimension}	Int32	-3	omitted or NULL	An Int32 where it is either a single-dimensional array or a scalar

- The TypeDefinition is specified for *Objects* and *Variables*.

- The TypeDefinition column specifies a *NodeId* of a *TypeDefinitionNode*, i.e. the specified *Node* points with a *HasTypeDefinition Reference* to the corresponding *TypeDefinitionNode*. The symbolic name of the *NodeId* is used in the table.

- The *Other* of the referenced component is provided by specifying the symbolic name of the rule in the *Other* column. In the *AddressSpace*, the *Node* shall use a *HasOther Reference* to point to the corresponding *Other Object*.

If the *NodeId* of a *DataType* is provided, the symbolic name of the *Node* representing the *DataType* shall be used.

Nodes of all other *NodeClasses* cannot be defined in the same table; therefore only the used *ReferenceType*, their *NodeClass* and their *BrowseName* are specified. A reference to another of this document points to their definition.

If no components are provided, the DataType, TypeDefinition and Other columns may be omitted and only a Comment column is introduced to point to the *Node* definition.

Components of *Nodes* can be complex, i.e. containing components by themselves. The TypeDefinition, NodeClass, DataType and Other can be derived from the type definitions, and the symbolic name can be created as defined in 3.5.2.1. Therefore those containing components are not explicitly specified; they are implicitly specified by the type definitions.

3.5.2 NodeIds and BrowseNames

3.5.2.1 NodeIds

The *NodeIds* of all *Nodes* described in this document are only symbolic names. Annex A defines the actual *NodeIds*.

The symbolic name of each *Node* defined in this document is its *BrowseName*, or, when it is part of another *Node*, the *BrowseName* of the other *Node*, a ".", and the *BrowseName* of itself. In this case "part of" means that the whole has a *0:HasProperty* or *HasComponent Reference* to its part. Since all *Nodes* not being part of another *Node* have a unique name in this document, the symbolic name is unique.

The namespace for this specification is defined in Annex A. The NamespaceIndex for all *NodeIds* defined in this specification is server specific and depends on the position of the namespace URI in the server namespace table.

Note: This specification does not only define concrete *Nodes*, but also requires that some Nodes have to be generated, for example one for each device type available in the frame application. The *NodeIds* of those *Nodes* are server-specific, including the Namespace. But the NamespaceIndex of those *Nodes* cannot be the NamespaceIndex used for the Nodes defined by this specification, because they are not defined by this specification but generated by the Server.

3.5.2.2 BrowseNames

The text part of the *BrowseNames* for all *Nodes* defined in this specification is specified in the tables defining the *Nodes*. The NamespaceIndex for all *BrowseNames* defined in this specification is server specific and depends on the position of the namespace URI defined in this specification in the server namespace table.

If the BrowseName is not defined by this specification, a namespace index prefix like '0:EngineeringUnits' is added to the BrowseName. This is typically necessary if a Property of

another specification is overwritten or used in the OPC UA types defined in this specification. Table 260 provides a list of namespaces used in this specification.

3.5.3 Common Attributes

3.5.3.1 General

For all *Nodes* specified in this specification, the *Attributes* named in Table 3 shall be set as specified in the table.

Table 3 – Common Node Attributes

Attribute	Value
DisplayName	The *DisplayName* is a *LocalizedText*. Each server shall provide the *DisplayName* identical to the *BrowseName* of the *Node* for the LocaleId “en”. Whether the server provides translated names for other LocaleIds is vendor specific.
Description	Optionally a vendor specific description is provided
NodeClass	Shall reflect the *NodeClass* of the *Node*
NodeId	The *NodeId* is described by *BrowseNames* as defined in 3.5.2.1 and defined in Annex A.
WriteMask	Optionally the *WriteMask Attribute* can be provided. If the *WriteMask Attribute* is provided, it shall set all *Attributes* to not writeable that are not said to be vendor-specific. For example, the *Description Attribute* may be set to writeable since a Server may provide a server-specific description for the *Node*. The *NodeId* shall not be writeable, because it is defined for each *Node* in this specification.
UserWriteMask	Optionally the *UserWriteMask Attribute* can be provided. The same rules as for the *WriteMask Attribute* apply.

3.5.3.2 Objects

For all *Objects* specified in this specification, the *Attributes* named in Table 4 shall be set as specified in the table.

Table 4 – Common Object Attributes

Attribute	Value
EventNotifier	Whether the *Node* can be used to subscribe to *Events* or not is vendor specific

3.5.3.3 Variables

For all *Variables* specified in this specification, the *Attributes* named in Table 5 shall be set as specified in the table.

Table 5 – Common Variable Attributes

Attribute	Value
MinimumSamplingInterval	Optionally, a vendor-specific minimum sampling interval is provided
AccessLevel	The access level for *Variables* used for type definitions is vendor-specific, for all other *Variables* defined in this part, the access level shall allow a current read; other settings are vendor specific.
UserAccessLevel	The value for the *UserAccessLevel Attribute* is vendor-specific. It is assumed that all *Variables* can be accessed by at least one user.
Value	For *Variables* used as *InstanceDeclarations,* the value is vendor-specific; otherwise it shall represent the value described in the text.
ArrayDimensions	If the *ValueRank* does not identify an array of a specific dimension (i.e. *ValueRank* <= 0) the *ArrayDimensions* can either be set to null or the *Attribute* is missing. This behaviour is vendor-specific. If the *ValueRank* specifies an array of a specific dimension (i.e. *ValueRank* > 0) then the *ArrayDimensions Attribute* shall be specified in the table defining the *Variable*.

3.5.3.4 VariableTypes

For all *VariableTypes* specified in this specification, the *Attributes* named in Table 6 shall be set as specified in the table.

Table 6 – Common VariableType Attributes

Attributes	Value
Value	Optionally a vendor-specific default value can be provided
ArrayDimensions	If the *ValueRank* does not identify an array of a specific dimension (i.e. *ValueRank* <= 0) the *ArrayDimensions* can either be set to null or the *Attribute* is missing. This behaviour is vendor-specific. If the *ValueRank* specifies an array of a specific dimension (i.e. *ValueRank* > 0) then the *ArrayDimensions Attribute* shall be specified in the table defining the *VariableType*.

4 General information to BACnet and OPC UA

4.1.1 Introduction to BACnet

4.1.1.1 General

BACnet (Building Automation and Control networks) is a data-communication protocol specifically designed for building automation applications.

1987 the ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers) founded the committee 135 to create an open and neutral communication protocol for building automation. First published in 1995 the BACnet standard is under continuous maintenance by SSPC 135. A public review process before the formal approval assures acceptance by the industry or interested parties.

While starting as a national American standard BACnet has become a world-wide standard and was accepted by ISO as the standards ISO 16484-5 and ISO 16484-6.

The general architecture of BACnet describes three main parts, the BACnet data link layers, the BACnet objects and the BACnet application services.

4.1.1.2 BACnet Data-Link-Layers

BACnet supports a total of 9 different network media (data-link-layer).

Those are:

- BACnet Ethernet (ISO 8802-3 aka Layer-2 Ethernet, rarely used)
- ARCnet (Network or EIA-485, rarely used)
- LonTalk (any media supported by LonTalk, rarely used)
- MS/TP (Master/Slave Token Passing) based on EIA-485 (serial networks, commonly used)
- PTP (Point-to-Point) based on EIA-232 (serial connection, rarely used)
- BACnet/IP (based upon IPv4 and UDP communication, commonly used)
- BACnet/IPv6 (based upon IPv6 and UDP communication, rarely used)
- ZIGBEE (wireless mesh networks, rarely used)
- BACnet/SecureConnect (based upon TCP / websocket based communication, rarely used but expected to be commonly used in the future)

4.1.1.3 BACnet Objects

BACnet models the data of building automation components as objects. Those include Analog, Binary and Multi-State Input, Output and Value Objects, primitive data, the representation of devices, counters, calendar and scheduling, trend-logging, alarming, life-safety and access control objects as well as objects to represent a building structure (groups, structured views).

Objects consist of properties to represent the data associated to the specific information. This includes text information like the name and description, the technical address of the data, the

present value, units, alarm-limits as well as specific building automation functionality like command priorities, elapsed active time or change-of-state counting. Due to the de-central approach of BACnet a client can request all information from a device representing data as a server.

4.1.1.4 BACnet Application Services

The BACnet application services allow access to the objects and provide functions to interact between BACnet applications in a network. Another set of network services on layer 3 of the ISO/OSI model provide routing functions between the different data-link-layers as well as exchanging information about network security.

Application services include object access like reading, writing and change-of-value. Other services provide file-transfer, alarm information, remote device and network management and virtual terminal functions.

To assure interoperability between applications BACnet specifies a PICS (Protocol Implementation Conformance Statement) document. A PICS is a self-declaration by the vendor of a BACnet device and describes which parts of BACnet are supported by the implementation.

4.1.1.5 BACnet Device Profiles

BACnet devices may support one of the 8 device profiles specified for better interoperability.

The profiles are:

- B-AWS: Advanced Operator Workstation
- B-OWS: Operator Workstation
- B-OD: Operator Display
- B-BC: Building Controller
- B-AAC: Advanced Application Controller
- B-ASC: Application specific controller
- B-SS: Smart Sensor
- B-SA: Smart Actuator

4.1.2 Introduction to OPC Unified Architecture

4.1.2.1 General

The main use case for OPC standards is the online data exchange between devices and HMI or SCADA systems using Data Access functionality. In this use case the device data is provided by an OPC server and is consumed by an OPC client integrated into the HMI or SCADA system. OPC DA provides functionality to browse through a hierarchical namespaces containing data items and to read, write and to monitor these items for data changes. The classic OPC standards are based on Microsoft COM/DCOM technology for the communication between software components from different vendors. Therefore classic OPC server and clients are restricted to Windows PC based automation systems.

OPC UA incorporates all features of classic OPC standards like OPC DA, A&E and HDA but defines platform independent communication mechanisms and generic, extensible and object-oriented modelling capabilities for the information a system wants to expose.

The OPC UA network communication part defines different mechanisms optimized for different use cases. The first version of OPC UA is defining an optimized binary TCP protocol for high performance intranet communication as well as a mapping to accepted internet standards like Web Services. The abstract communication model does not depend on a specific protocol mapping and allows adding new protocols in the future. Features like security, access control and reliability are directly built into the transport mechanisms. Based on the platform independence of the protocols, OPC UA servers and clients can be directly integrated into devices and controllers.

The OPC UA *Information Model* provides a standard way for *Servers* to expose *Objects* to *Clients*. *Objects* in OPC UA terms are composed of other *Objects*, *Variables* and *Methods*. OPC UA also allows relationships to other *Objects* to be expressed.

The set of *Objects* and related information that an OPC UA *Server* makes available to *Clients* is referred to as its *AddressSpace*. The elements of the OPC UA *Object* Model are represented in the *AddressSpace* as a set of *Nodes* described by *Attributes* and interconnected by *References*. OPC UA defines eight classes of *Nodes* to represent *AddressSpace* components. The classes are *Object*, *Variable*, *Method*, *ObjectType, DataType, ReferenceType* and *View*. Each *NodeClass* has a defined set of *Attributes*.

This specification makes use of three essential OPC UA *NodeClasses*: *Objects, Methods* and *Variables*.

Objects are used to represent components of a system. An *Object* is associated to a corresponding *ObjectType* that provides definitions for that *Object*.

Methods are used to represent commands or services of a system.

Variables are used to represent values. Two categories of *Variables* are defined, *Properties* and *DataVariables*.

Properties are *Server*-defined characteristics of *Objects*, *DataVariables* and other *Nodes*. *Properties* are not allowed to have *Properties* defined for them. An example for *Properties* of *Objects* is the *Object_Identifier Property* of a *BACnetObjectType*.

DataVariables represent the contents of an *Object*. *DataVariables* may have component *DataVariables*. This is typically used by *Servers* to expose individual elements of arrays and structures. This specification uses *DataVariables* to represent data like the Present_Value of a *BACnetAnalogInput Object*.

4.1.2.2 Graphical Notation

OPC UA defines a graphical notation for an OPC UA *AddressSpace*. It defines graphical symbols for all *NodeClasses* and how different types of *References* between *Nodes* can be visualized. Figure 1 shows the symbols for the six *NodeClasses* used in this specification. *NodeClasses* representing types always have a shadow.

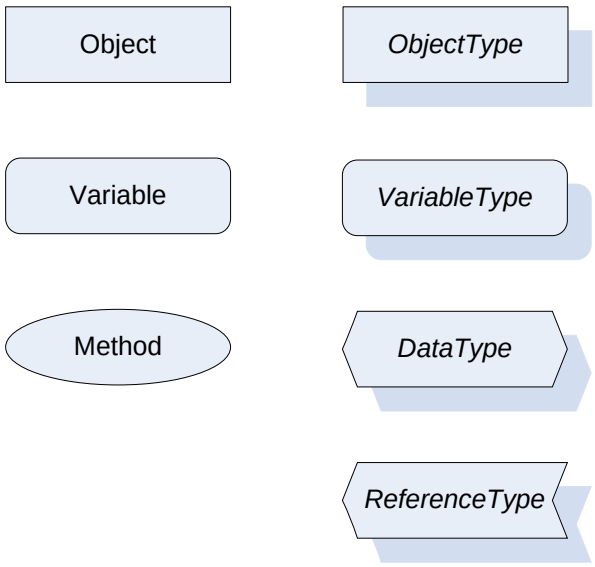


Figure 1 – OPC UA *Graphical Notation for NodeClasses*

Figure 2 shows the symbols for the *ReferenceTypes* used in this specification. The *Reference* symbol is normally pointing from the source *Node* to the target *Node*. The only exception is the *HasSubType Reference*. The most important *References* like *HasComponent*, *0:HasProperty*, HasTypeDef*i*nition and *HasSubType* have special symbols avoiding the name of the *Reference*. For other *ReferenceTypes* or derived *ReferenceTypes* the name of the *ReferenceType* is used together with the symbol.

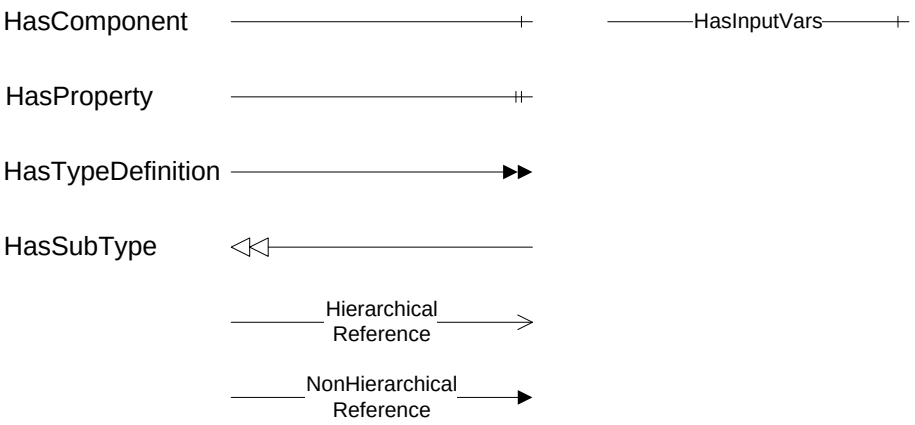


Figure 2 – OPC UA *Graphical Notation for References*

Figure 3 shows a typical example for the use of the graphical notation. Object_A and Object_B are instances of the ObjectType_Y indicated by the HasTypeDefinition *References*. The ObjectType_Y is derived from ObjectType_X indicated by the HasSubType *Reference*. The Object_A has the components Variable_1, Variable_2 and Method_1.

To describe the components of an *Object* on the *ObjectType* the same *NodeClasses* and *References* are used on the *Object* and on the *ObjectType* like for ObjectType_Y in the example. The instance *Nodes* used to describe an *ObjectType* are instance declaration *Nodes*.

To provide more detailed information for a *Node*, a subset or all *Attributes* and their values can be added to a graphical symbol.

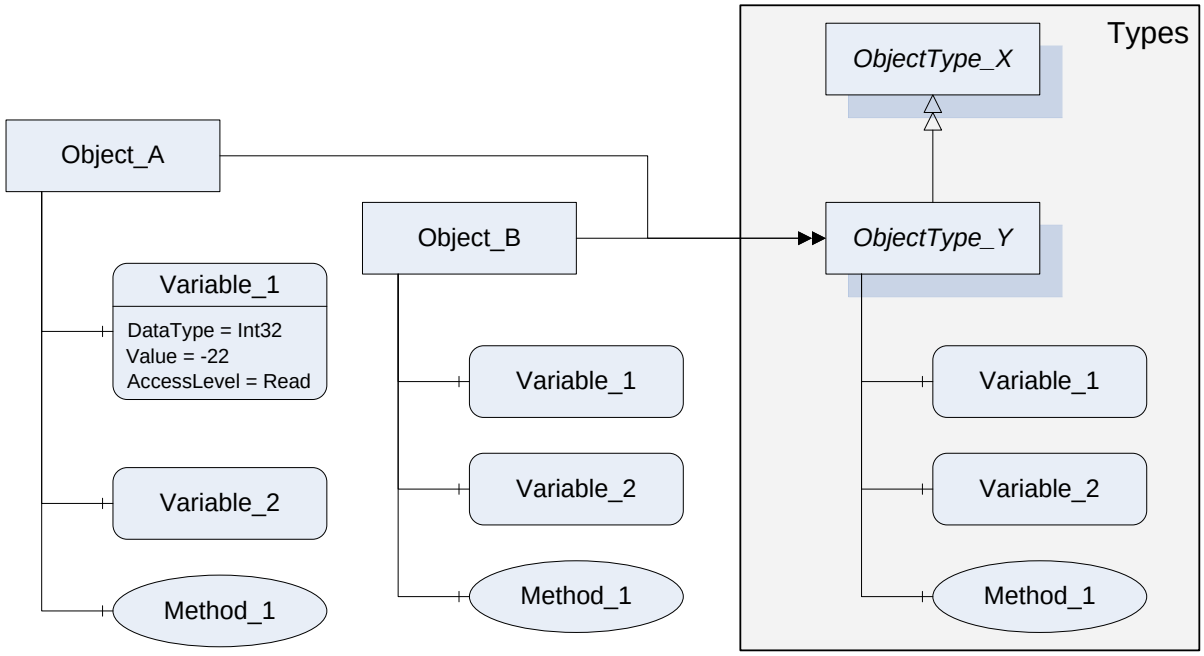


Figure 3 – OPC UA *Graphical Notation Example*

4.1.3 Use Cases

The following use cases illustrate the usage of the information model. Not all necessary *Objects* must be realized within a concrete OPC UA Server. These use cases cover a system that is BACnet client and OPC UA server. The use cases for the other direction (OPC UA client and BACnet server) are not covered in the current version and may be added in a future version.

- Observation

  Observation comprises reading and monitoring data such as present value, trend log and events of BACnet objects available in an existing BACnet network from a BACnet OPC UA Server.

  The representation of BACnet internetworks, devices their objects and properties is consistent across BACnet OPC UA server products.

  The BACnet OPC UA server may restrict the exposed information from a BACnet internetwork through configuration and/or user autorization.

  Example 1: Enterprise integration (monitoring energy consumption, comparing trends)

  Example 2: Multi domain integration (building automation with industrial automation)

- Operation

  Operation inherits the functionality of observation and extends it.

  Operation comprises writing data such as set points, selecting mode of operation or acknowledge alarms through the BACnet OPC UA server.

  Example 1: Enterprise integration (writing set points)

  Example 2: Multi domain integration (building automation with industrial automation) (writing operation mode)

- Engineering (Configuration / Maintenance)

  Engineering inherits the functionality of operation and extends it.

  Engineering comprises of configuring BACnet objects through a BACnet OPC UA *Server* like configuring modes of operation, enabling/disabling alarms and configuring schedules.

  Example1: The properties of BACnet devices may be configured by OPC UA clients during the installation or engineering phase.

  Example2: If room assignments or names change, the object representation may need to be updated accordingly. The OPC UA mapping model may be used to access and modify this information.

The following Figure 4 shows the use case diagram.

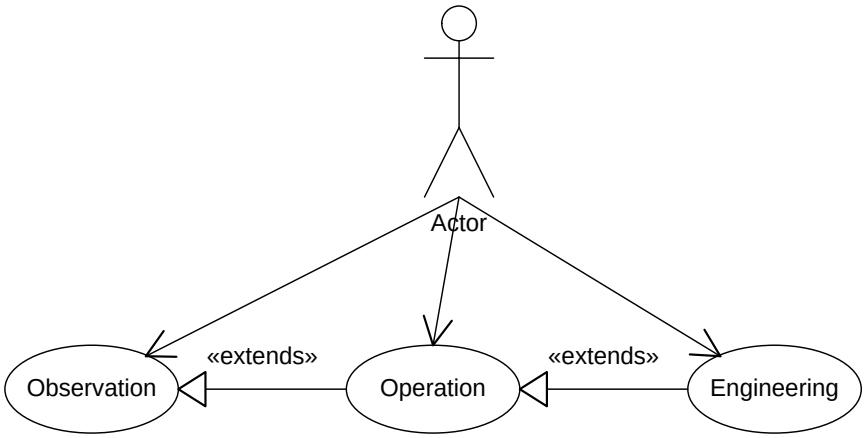


Figure 4 – Use case diagram

5 BACnet OPC UA Model Overview

5.1 Modelling concepts

BACnet defines a list of object types where all common properties are repeated for each type. Type hierarchies and inheritance are not used in BACnet.

Since OPC UA supports type hierarchies, inheritance and aggregation, these concepts are used to avoid duplicated definitions in the OPC UA representation of BACnet. The example of a BACnet analog input object type is used in this overview chapter to describe how the different concepts are used in this mapping specification.

The following three concepts are used to reduce duplicated definitions in this specification

- Type hierarchies and inheritance is used to map common BACnet properties on a base type or different levels of a type hierarchy.

- BACnet properties that match existing OPC UA *Attributes* or *Properties* are mapped to existing OPC UA concepts like BACnet Object_Name to OPC UA *Attribute BrowseName* or the BACnet properties Units, Min_Pres_Value and Max_Pres_Value to OPC UA *Properties* of the *AnalogItemType.*

- Aggregation is used if BACnet properties can be logically grouped or a group of BACnet properties is used in different OPC UA *ObjectTypes* across the type hierarchy.

Figure 5 shows an example for type hierarchies and inheritance used to map the BACnet Analog Input object type to OPC UA. The BACnet properties are shown in the same order as defined in the BACnet standard.

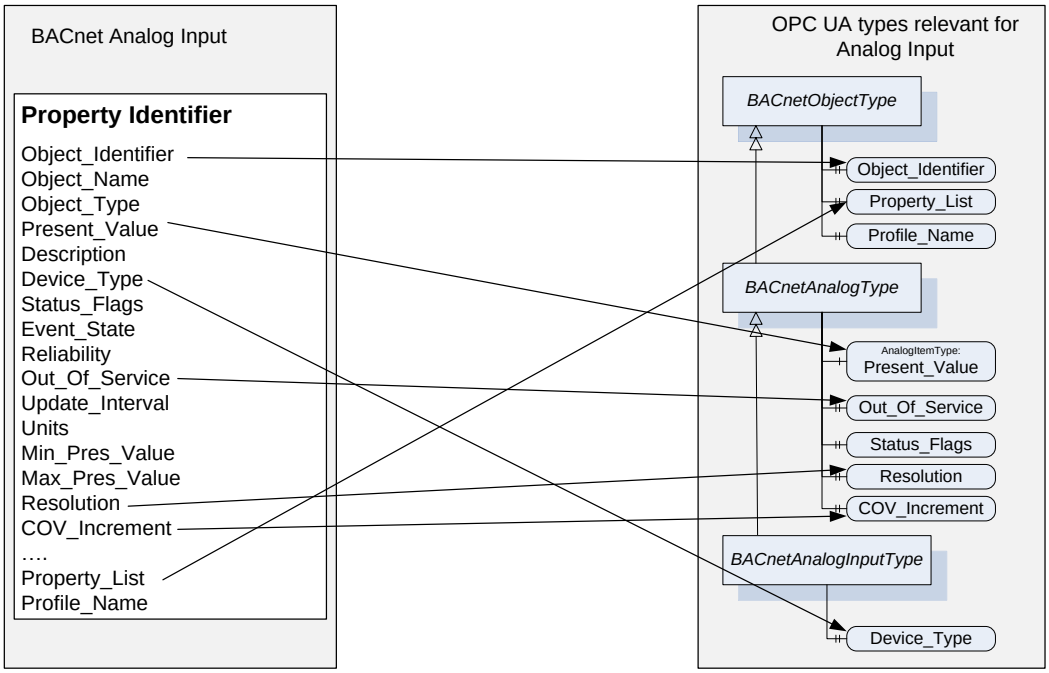


Figure 5 – Mapping with inheritance and type hierarchies

The BACnet properties defined for all BACnet object types are mapped to the *BACnetObjectType*. This abstract OPC UA ObjectType is used as root for the BACnet object type hierarchy.

The BACnet properties defined for all BACnet analog object types are mapped to the abstract *BACnetAnalogType*. The remaining property is mapped to the concrete OPC UA *ObjectType BACnetAnalogInputType*.

Figure 6 shows an example for mapping BACnet properties to OPC UA built-in concepts like *TypeDefinition*, *Attributes* or standard OPC UA *Properties*.

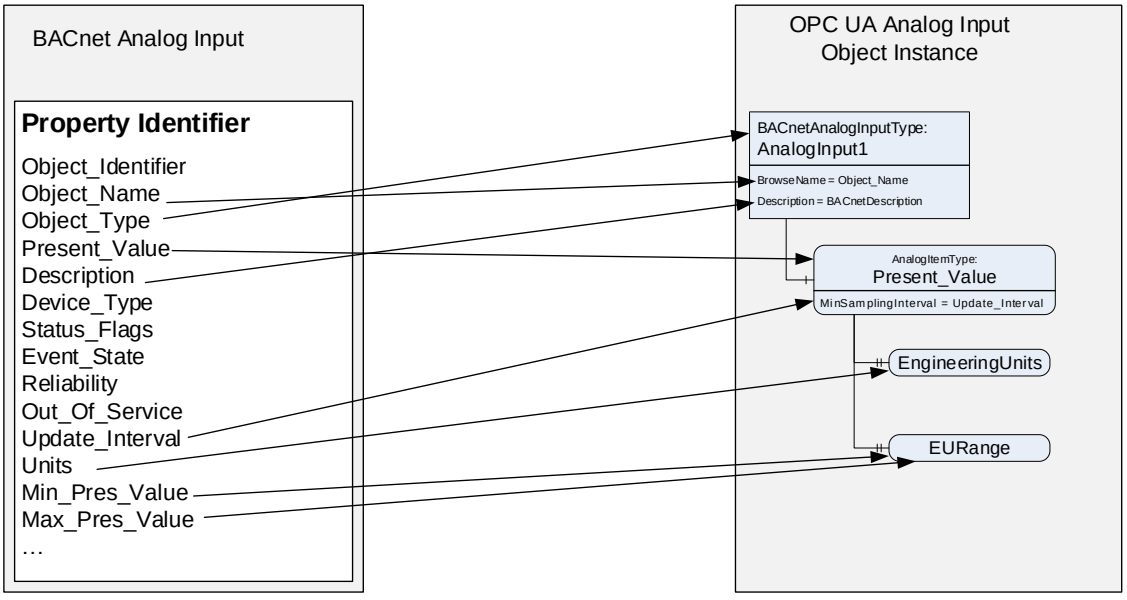


Figure 6 – Mapping to OPC UA Attributes and Properties

The BACnet property Object_Type is mapped to the *TypeDefinition* of the OPC UA *Object* instance.

The BACnet properties Object_Name, Description and Update_Interval are mapped to the OPC UA *Attributes BrowseName*, *Description* and *MinSamplingInterval*.

The BACnet properties Units, Min_Pres_Value and Max_Pres_Value are mapped the OPC UA *Properties EngineeringUnits* and *EURange* of the OPC UA *VariableType AnalogItemType*.

Figure 7 shows an example how groups of BACnet properties are integrated through aggregation.

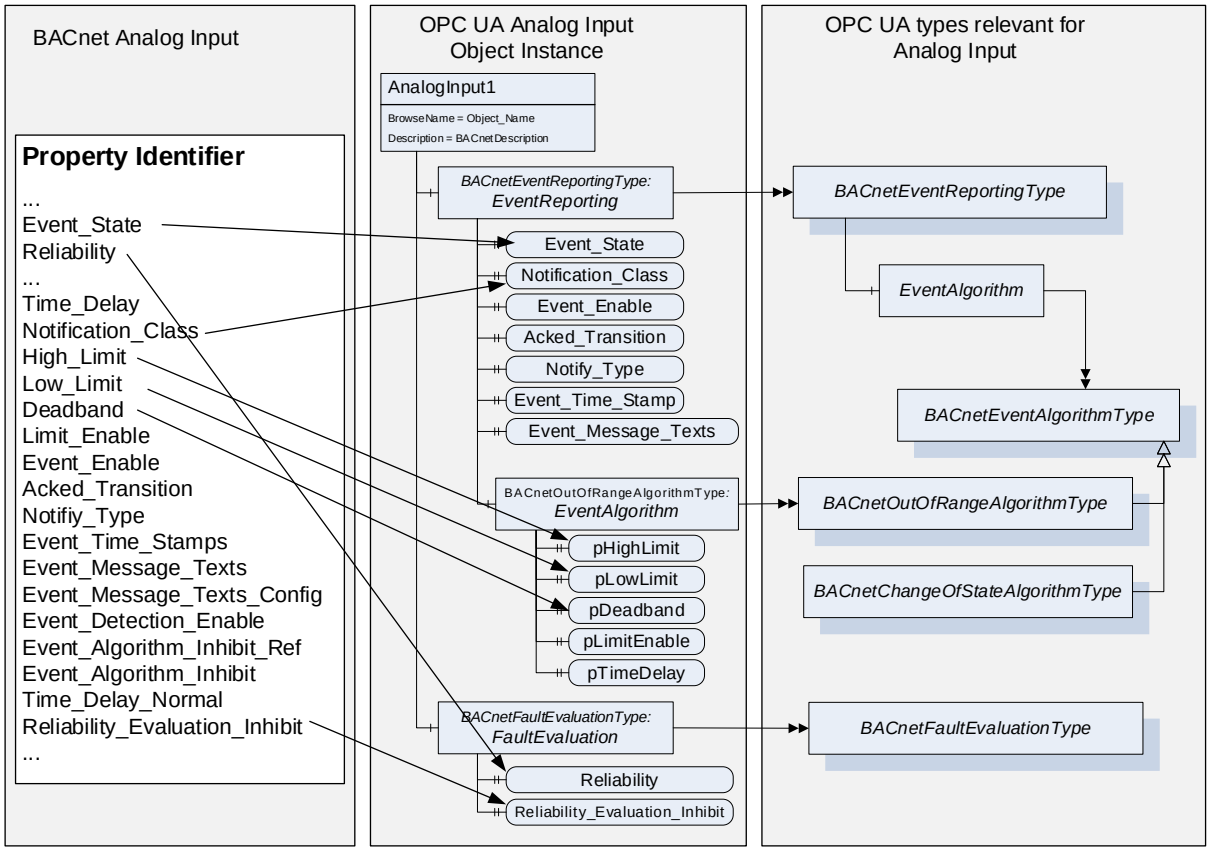


Figure 7 – Mapping aggregation of BACnet property groups

The BACnet properties used to configure event reporting like Event_State, Notification_Class, Event_Enabled and Notify_Type are grouped in the OPC UA *ObjectType BACnetEventReportingType*. An instance of this type is then aggregates as OPC UA *Object EventReporting* in the OPC UA *Object* AnalogInput1.

The BACnet properties used for the event algorithm in the event reporting like High_Limit, Low_Limit or Deadband are aggregated in the OPC UA *Object EventAlgorithm*. Since different event algorithms can be used in event reporting, they build their own event algorithm type hierarchy with the base *ObjectType BACnetEventAlgorithmType* and the derived types like *BACnetOutOfRangeAlgorithmType* that is used in the *BACnetAnalogInputType*.

The BACnet properties used to configure fault evaluation like Reliability and Reliability_Evaluation_Inhibit are grouped in the OPC UA *ObjectType BACnetFaultEvaluationType*. An instance of this type is then aggregates as OPC UA *Object FaultEvaluation* in the OPC UA *Object* AnalogInput1.

5.2 Model Overview

Figure 8 depicts the main *ObjectTypes* of the OPC UA for BACnet information model and their relationship. The drawing is not intended to be complete.

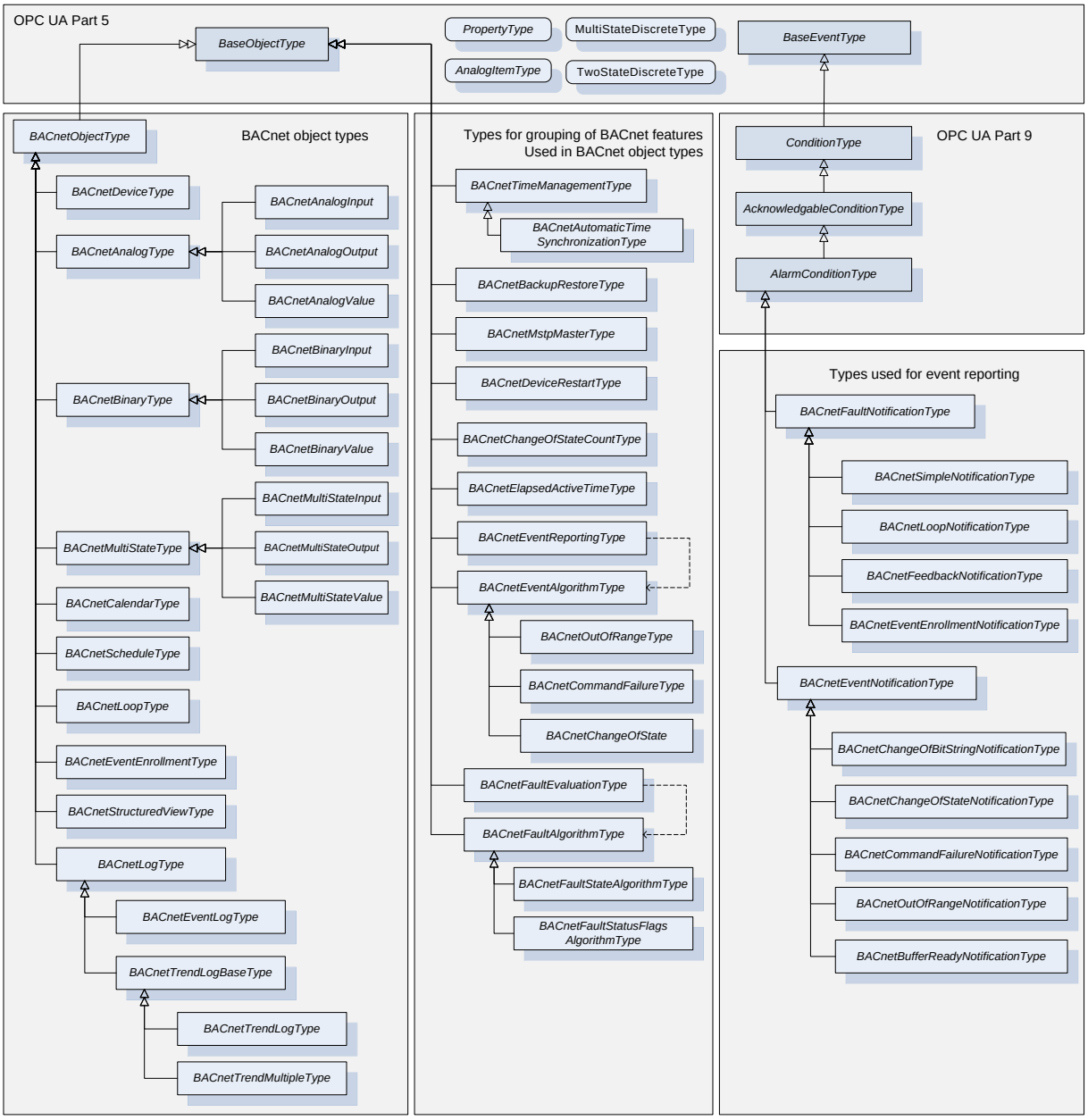


Figure 8 – *BACnet OPC UA Model* Overview

The boxes in this drawing show the *ObjectTypes* used in this specification as well as some elements from other specifications. The upper grey box shows the OPC UA core *ObjectType* from which the OPC UA for BACnet information model *ObjectTypes* are derived and some *VariableTypes* used in the *BACnet ObjectTypes*.

The left grey box in the second level shows the main *ObjectTypes* that this specification introduces. They represent corresponding BACnet object types. A type hierarchy is used whenever identical components are used in different BACnet object types.

The right grey box in the second level shows the *ObjectTypes* used for grouping of BACnet features. These groupings are used in the OPC UA for BACnet information model *ObjectTypes* shown in the left grey box.

Figure 9 provides an example for the mapping of a BACnet Analog Input object type to an OPC UA *ObjectType*.

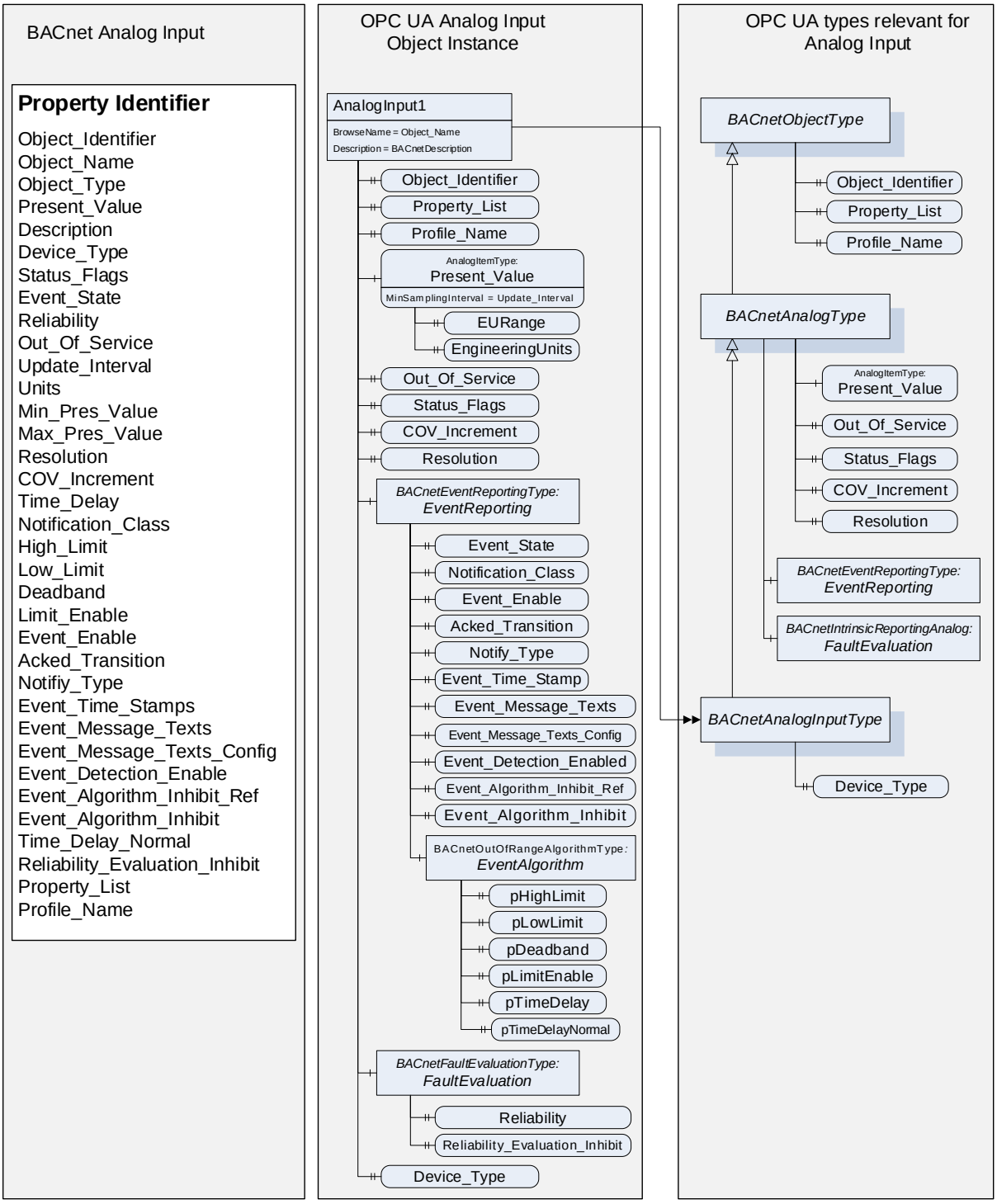


Figure 9 – *BACnet mapping example*

The left grey box shows the list of BACnet properties of the BACnet Analog Input object type.

The middle grey box shows an instance of an OPC UA *BACnetAnalogInput ObjectType*.

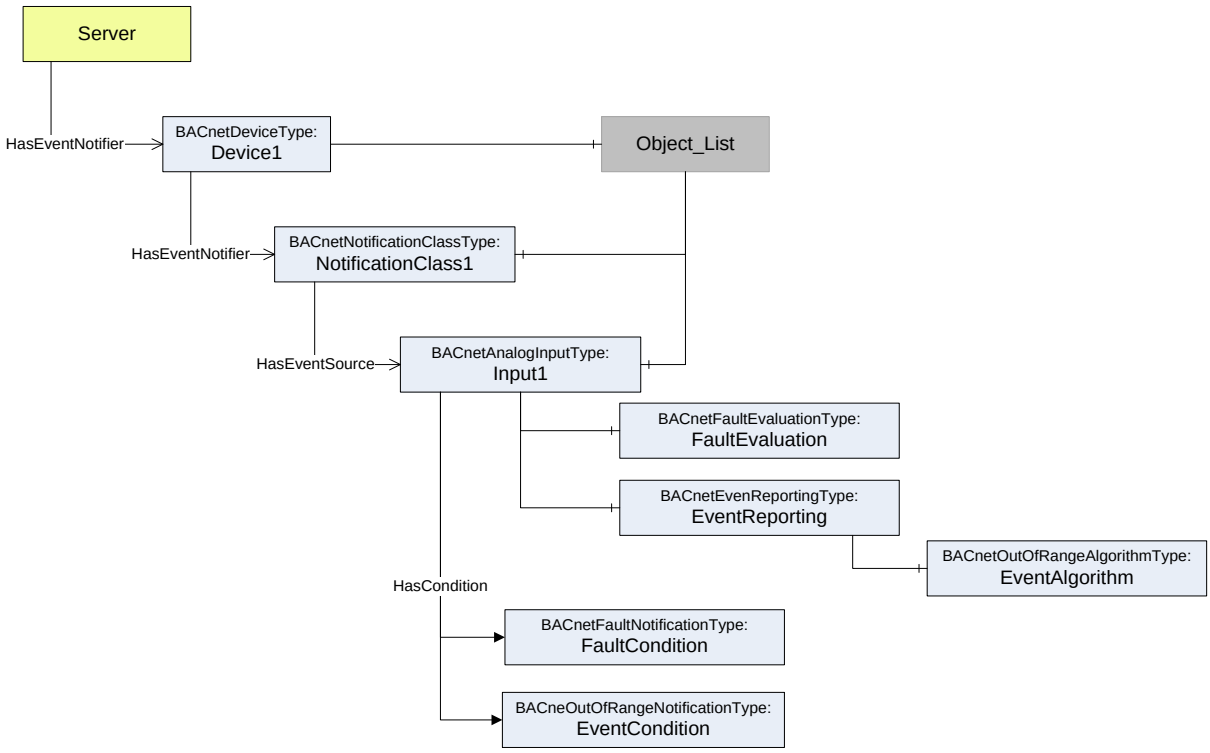
The right grey box shows the OPC UA *ObjectTypes* used to represent a BACnet Analog Input object type.

Most of the BACnet properties are mapped to OPC UA *Properties* using the BACnet property name as OPC UA *BrowseName*. They are either *Properties* of the *Object* directly or *Properties* of the *EventReporting Object*.

The following BACnet properties are mapped to existing OPC UA information.

- Object_Name is mapped to the *BrowseName* of the OPC UA *Object*
- Object_Type is mapped to the type definition of the OPC UA *Object*
- Description is mapped to the OPC UA *Attribute Description* of the OPC UA *Object*
- Present_Value, Units, Min_Pres_Value and Max_Pres_Value are mapped to an OPC UA *AnalogItemType Variable* with the name Present_Value
- Update is mapped to the OPC UA *Attribute MinimumSamplingInterval* of the OPC UA *Variable* Present_Value

5.3 Event and alarm handling



5.4 Character Set handling

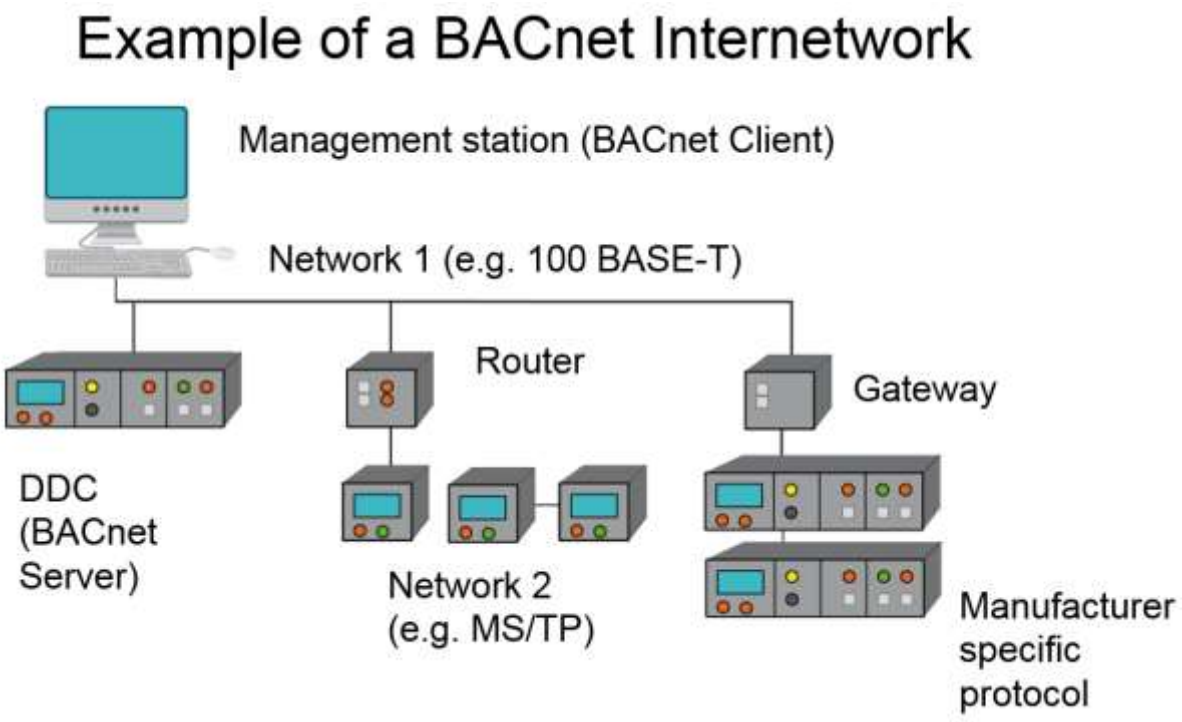
All character string information provided through this information model is solely ISO 10646 (UTF-8). A gateway application may convert character string information into other character sets according to the BACnet device capabilities.

6 OPC UA ObjectTypes used for structuring the address space

6.1 BACnetInternetworkType

6.1.1 General

This OPC UA *ObjectType* represents a *BACnet Internetwork*. A BACnet Internetwork consists of one or more BACnet Devices and may be setup using different BACnet Data-Link-Layers connected through BACnet router devices.



6.1.2 ObjectType definition

The *BACnetInternetworkType* is formally defined in Table 7.

Table 7 – *BACnetInternetworkType* Definition

Attribute	**Value**				
BrowseName	BACnet*Internetwork*Type				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasComponent	Object	<BACnetDeviceName>		BACnetDeviceType	OP
0:HasComponent	Method	TranslateBACnetIds			M
0:HasComponent	Method	NetworkScan			O
0:HasComponent	Method	AddDeviceByAddress			O
0:HasComponent	Method	AddDeviceById			O
0:HasComponent	Method	GetDeviceIdList			M
0:HasComponent	Method	TimeSynchronization			O

The *BACnetObjectType* is an abstract type and cannot be used directly.

6.1.3 ObjectType Description

6.1.3.1 Objects <BACnetDeviceName>

List of BACnet devices contained in the internetwork. The *BACnetUaMapper* may limit the list to the devices configured in the *BACnetUaMapper* or configured through the *Methods NetworkScan*, *AddDevice* and *RemoveDevice*.

6.1.3.2 Method TranslateBACnetIds

TranslateBACnetIds allows an OPC UA *Client* to request the NodeIds of OPC UA *Objects* representing BACnet objects by passing in the BACnet object identifier for the device and the object. This enables a *Client* to translate the BACnet address into an OPC UA *NodeId*.

Signature

```
TranslateBACnetIds(
   [in]  BACnetDeviceObjectPropertyReference[]   BACnetIds
   [out] NodeId[]                                OpcUaIds
   );
```

Argument	Description
BACnetIds	List of BACnet references to BACnet devices, objects and properties. The BACnetDeviceObjectPropertyReference DataType is defined in 10.5.9.
OpcUaIds	List of NodeIds of the OPC UA *Objects* representing the requested BACnet object in the BACnet device.

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

6.1.3.3 Method NetworkScan

NetworkScan allows an OPC UA *Client* to update the BACnet device list by using the BACnet Who-Is and I-Am services.

Signature

```
NetworkScan (
   [in]  0:UInt32                          WaitTimeInSeconds
   [in]  0:Boolean                         ApplyRange
   [in]  0:UInt32                          DeviceRangeLow
   [in]  0:UInt32                          DeviceRangeHigh
   [out] BaseDataType []                   DeviceAddressBindings
   [out] 0:UInt32 []                       MaxAPDULengthAccepted
   [out] BACnetSegmentation []             SegmentationSupported
   [out] 0:UInt16 []                       VendorIdentifier
   );
```

Argument	Description
WaitTimeInSeconds	Wait time between sending Who-Is and returning the received I-Am results.
ApplyRange	Flag indicating if the DeviceRangeLow and DeviceRangeHigh parameters are applied.
DeviceRangeLow	This parameter is an unsigned integer in the range 0 - 4194303. In conjunction with the 'Device Instance Range High Limit' parameter, it defines the devices that are qualified to respond with an I-Am service request. If the 'Device Instance Range Low Limit' parameter is present, then the 'Device Instance Range High Limit' parameter shall also be present, and only those devices whose Device Object_Identifier instance number falls within the range 'Device Instance Range Low Limit' ≤ Device Object_Identifier Instance Number ≤ 'Device Instance Range High Limit' shall be qualified to respond. The value of the 'Device Instance Range Low Limit' shall be less than or equal to the value of the 'Device Instance Range High Limit'. If the 'Device Instance Range Low Limit' and 'Device Instance Range High Limit' parameters are omitted, then all devices that receive this message are qualified to respond with an I-Am service request.
DeviceRangeHigh	This parameter is an unsigned integer in the range 0 - 4194303. In conjunction with the 'Device Instance Range Low Limit' parameter, it defines the devices that are qualified to respond with an I-Am service request. If the 'Device Instance Range High Limit' parameter is present, then the 'Device Instance Range Low Limit' parameter shall also be present, and only those devices whose Device Object_Identifier instance number falls within the range 'Device Instance Range Low Limit' ≤ Device Object_Identifier Instance Number ≤ 'Device Instance Range High Limit' shall be qualified to respond. The value of the 'Device Instance Range High Limit' shall be greater than or equal to the value of the 'Device Instance Range Low Limit'. If the 'Device Instance Range Low Limit' and 'Device Instance Range High Limit' parameters are omitted, then all devices that receive this message are qualified to respond with an I-Am service request.
DeviceAddressBindings	The format is server-specific. In each value of the array should be the device identifier and the BACnet MAC address provided by the I-Am service.
MaxAPDULengthAccepted	This element, of type Unsigned, is the maximum number of octets that may be contained in a single, indivisible application layer protocol data unit. The value of this property shall be greater than or equal to 50. The value of this property is also constrained by the underlying data link technology. See Clauses 6 through 11. If the value of this property is not encodable in the 'Max APDU Length Accepted' parameter of a ConfirmedRequest-PDU, then the value encoded shall be the highest encodable value less than the value of this property. In such cases, a responding device may ignore the encoded value in favor of the value of this property, if it is known.
SegmentationSupported	This element, of type BACnetSegmentation, indicates whether the BACnet Device supports segmentation of messages and, if so, whether it supports segmented transmission, reception, or both: {SEGMENTED_BOTH, SEGMENTED_TRANSMIT, SEGMENTED_RECEIVE, NO_SEGMENTATION}
VendorIdentifier	This element, of type Unsigned16, is a unique vendor identification code, assigned by ASHRAE, which is used to distinguish proprietary extensions to the protocol.

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

6.1.3.4 Method AddDeviceByAddress

AddDeviceByAddress allows adding a BACnet device with a known address to *BACnetInternetworkType Object*. The necessary information can be retrieved from the *Method NetworkScan*. The *BACnetUaMapper* may limit access to this method to administrative users.

Signature

```
AddDeviceByAddress (
   [in]  BaseDataType     Address
   );
```

Argument	Description
Address	BACnet device identifier and mac address as returned by the NetworkScan.

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

6.1.3.5 Method GetDeviceIdList

GetDeviceIdList provides a list of known devices. The necessary information can be updated using the *Method NetworkScan*. The *BACnetUaMapper* may limit access to this method to administrative users.

Signature

```
GetDeviceIdList (
   [out] BACnetObjectIdentifier[]     BACnetDeviceIds,
   [out] NodeId[]                     OpcUaObjectIds
   );
```

Argument	Description
BACnetDeviceIds	An array of BACnetObjectIdentifiers known to the BACnetUaMapper
OpcUaObjectIds	An array of NodeIds containing the internal identification of the corresponding BACnetObjectIdentifier. Array shall have the same size as BACnetDeviceIds

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

6.1.3.6 Method AddDeviceById

AddDeviceById allows adding a BACnet device with a known internal Id to *BACnetInternetworkType Object*. The necessary information can be retrieved from the *Method GetDeviceIdList*. The *BACnetUaMapper* may limit access to this method to administrative users.

Signature

```
AddDeviceById (
   [in]  NodeId      DeviceObject
   );
```

Argument	Description
DeviceObject	The internal Id of the BACnet device to be added. Can be retrieved by calling GetDeviceIdList.

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

6.1.3.7 Method TimeSynchronization

This Method is used to update the time of the BACnet devices in the network. See B.3 for more details. This method is only provided if the *BACnetUaMapper* is a BACnet time master.

Signature

```
TimeSynchronization (
   [in]  UtcTime     Time
   );
```

Argument	Description
Time	The UTC time used to update the time of the BACnet devices.

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

7 OPC UA ObjectTypes representing BACnet object types

7.1 BACnetObjectType

7.1.1 General

This *ObjectType* defines the super type for all OPC UA *ObjectTypes* representing *BACnet* object types in an OPC UA *Address Space*. It introduces two properties common for all BACnet object types. Figure 10 shows an overview for the *BACnetObjectType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 8.

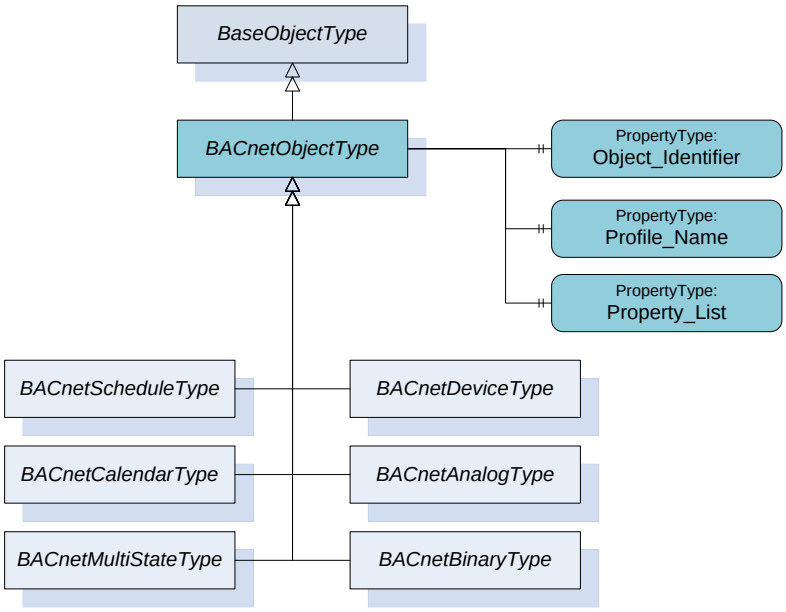


Figure 10 – *BACnetObjectType overview*

7.1.2 ObjectType definition

The *BACnetObjectType* is formally defined in Table 8.

Table 8 – *BACnetObjectType* Definition

Attribute	Value				
BrowseName	BACnetObjectType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Object_Identifier	BACnetObjectIdentifier	0:PropertyType	M
0:HasProperty	Variable	Profile_Name	0:String	0:PropertyType	O

The *BACnetObjectType* is an abstract type and cannot be used directly.

7.1.3 ObjectType Description

7.1.3.1 Standard OPC UA Object Attributes

The following BACnet properties are mapped to existing OPC UA *Object Attributes*.

- Object_Name is mapped to the *BrowseName* of the OPC UA *Object*
- Object_Type is mapped to the type definition of the OPC UA *Object*
- Description is mapped to the OPC UA *Attribute Description* of the OPC UA *Object*

7.1.3.2 Variable Object_Identifier

This OPC UA Property represents the BACnet property Object_Identifier.

It is the unique address within the BACnet device. It consists of a BACnet object type and instance number. The *BACnetObjectIdentifier DataType* is defined in 10.2.1.

7.1.3.3 Variable Profile_Name

This OPC UA Property represents the BACnet property Profile_Name.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified by BACnet.

7.2 BACnetObjectTypeUnknown

7.2.1 General

This *ObjectType* is used to represent object types not covered by this specification (standard or proprietary object types).

7.2.2 ObjectType definition

The *BACnetObjectTypeUnknown* is formally defined in Table 9.

Table 9 – BACnetObjectTypeUnknown Definition

Attribute	**Value**				
BrowseName	BACnetObjectTypeUnknown				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1					
0:HasComponent	Variable	Object_Type	BACnetObjectTypeEnum	0:BaseDataVariableType	M

7.2.3 ObjectType Description

7.2.3.1 Variable Object_Type

This OPC UA *Property*, of *DataType BACnetObjectTypeEnum*, represents the BACnet property Object_Type. The *BACnetObjectTypeEnum DataType* is defined in 10.4.21.

The property is an enumeration that describes the BACnet object type of the type not covered by this specification. This information is contained in the OPC UA type definition NodeId for the BACnet object types mapped in this specification.

7.3 BACnetDeviceType

7.3.1 General

This OPC UA *ObjectType* represents a *BACnet Device* object type. There is exactly one *BACnetDeviceType* in each *BACnet Device.* The *Object_Identifier* property of the *BACnetDeviceType* identifies the device and is unique throughout the BACnet internetwork. This *ObjectType* exposes standard BACnet services through OPC UA *methods.* These methods may be invoked to manage the configuration of the *BACnet Device*.

Figure 11 shows an overview for the *BACnetDeviceType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 10.

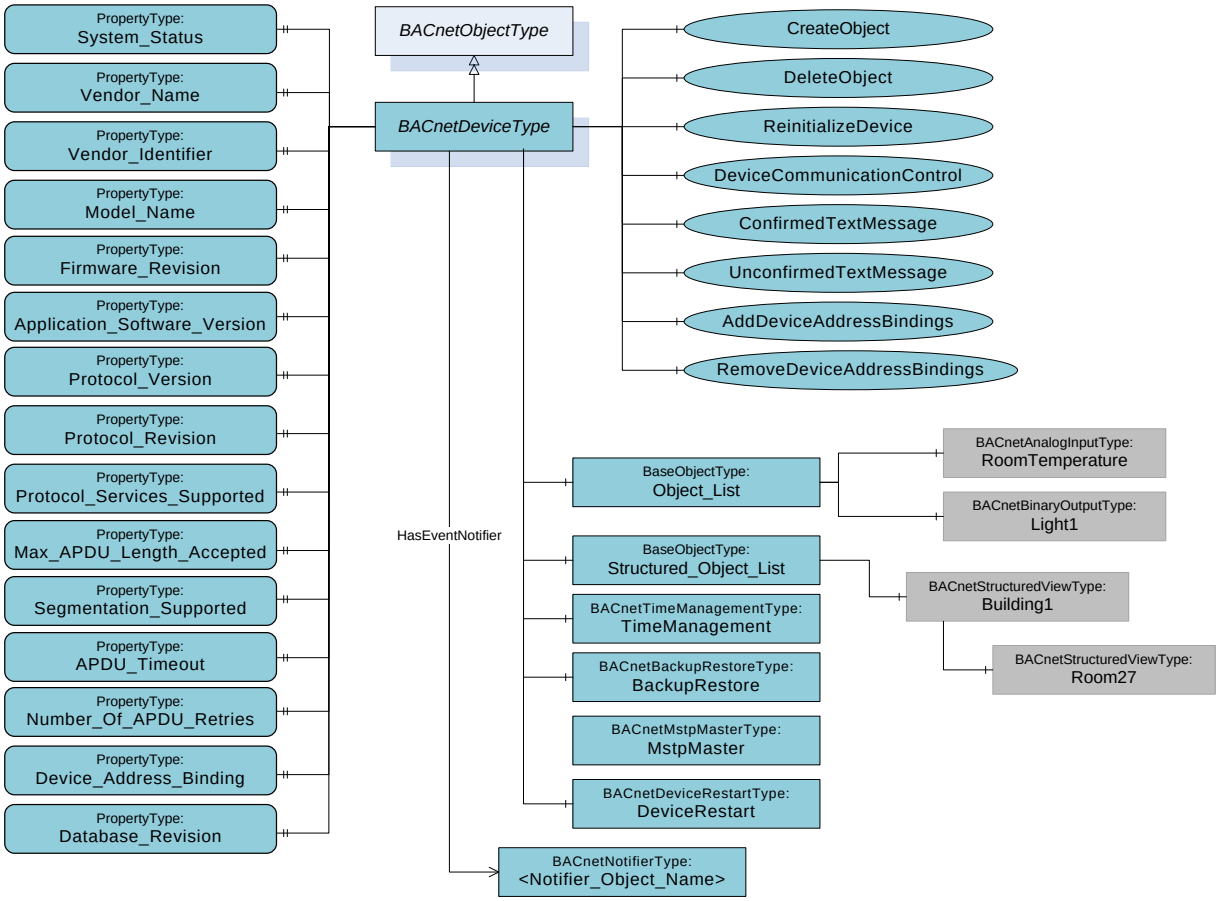


Figure 11 – *BACnetDeviceType overview*

7.3.2 ObjectType definition

The *BACnetDeviceType* is formally defined in Table 10.

Table 10 – BACnetDeviceType Definition

Attribute		Value			
BrowseName		BACnetDeviceType			
IsAbstract		False			
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1					
0:HasComponent	Object	Object_List		BaseObjectType	M
0:HasComponent	Object	Structured_Object_List		BaseObjectType	O
0:HasComponent	Object	TimeManagement		BACnetTimeManagementType	O
0:HasComponent	Object	BackupRestore		BACnetBackupRestoreType	O
0:HasComponent	Object	MstpMaster		BACnetMstpMasterType	O
0:HasComponent	Object	DeviceRestart		BACnetDeviceRestartType	O
0:HasNotifier	Object	<Notifier_Object_Name>		BACnetNotifierType	OP
0:HasProperty	Variable	System_Status	BACnetDeviceStatus	0:PropertyType	M
0:HasProperty	Variable	Vendor_Name	0:String	0:PropertyType	M
0:HasProperty	Variable	Vendor_Identifier	0:UInt16	0:PropertyType	M
0:HasProperty	Variable	Model_Name	0:String	0:PropertyType	M
0:HasProperty	Variable	Serial_Number	0:String	0:PropertyType	M
0:HasProperty	Variable	Firmware_Revision	0:String	0:PropertyType	M
0:HasProperty	Variable	Application_Software_Version	0:String	0:PropertyType	M
0:HasProperty	Variable	Location	0:String	0:PropertyType	O
0:HasProperty	Variable	Protocol_Version	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Protocol_Revision	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Protocol_Services_Supported	BACnetServicesSupportedBits	0:PropertyType	M
0:HasProperty	Variable	Protocol_Object_Types_Supported	BACnetObjectTypeSupportedBits	0:PropertyType	M
0:HasProperty	Variable	Max_APDU_Length_Accepted	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Segmentation_Supported	BACnetSegmentation	0:PropertyType	M
0:HasProperty	Variable	Max_Segments_Accepted	0:UInteger	0:PropertyType	O
0:HasProperty	Variable	APDU_Segment_Timeout	0:UInteger	0:PropertyType	O
0:HasProperty	Variable	APDU_Timeout	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Number_Of_APDU_Retries	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Device_Address_Binding	BACnetAddressBinding[]	0:PropertyType	M
0:HasProperty	Variable	Database_Revision	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Active_COV_Subscriptions	BACnetCOVSubscription[]	0:PropertyType	O
0:HasComponent	Method	CreateObject			O
0:HasComponent	Method	DeleteObject			O
0:HasComponent	Method	ReinitializeDevice			O
0:HasComponent	Method	DeviceCommunicationControl			O
0:HasComponent	Method	TextMessage			O
0:HasComponent	Method	AddDeviceAddressBindings			O
0:HasComponent	Method	RemoveDeviceAddressBindings			O
0:GeneratesEvent	ObjectType	BACnetNotificationType			

The *BACnetDeviceType ObjectType* is a concrete type and can be used directly.

The components of the *BACnetDeviceType* have additional subcomponents which are defined in Table 11.

Table 11 – BACnetDeviceType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
Object_List	0:HasProperty	Variable	Object_List	BACnetObjectIdentifier []	0:PropertyType	M
Object_List	0:HasComponent	Object	<BACnetObjectName>		BACnetObjectType	OP
Structured_Object_List	0:HasProperty	Variable	Structured_Object_List	BACnetObjectIdentifier []	0:PropertyType	M
Structured_Object_List	0:HasComponent	Object	<BACnetStructuredView Name>		BACnetStructuredViewType	OP

7.3.3 ObjectType Description

7.3.3.1 EventNotifier Attribute

If the BACnet device object contains BACnet notification class objects, the *SubscribeToEvents* flag is set in the *EventNotifier* OPC UA *Attribute*.

7.3.3.2 Object Object_List

Object_List gathers the references to BACnet objects of the BACnet device provided through the BACnet property Object_List.

The OPC UA *Property Object_List* on the *Object*_List object, of *DataType BACnetObjectIdentifier*[], represents the BACnet property Object_List. The *BACnetObjectIdentifier* DataType is defined in 10.2.1. It contains a list of BACnet objects within the device.

The list of *BACnetObjectTypes* on the *Object*_List object contains the list of BACnet objects within the device. The list may be limited through configuration of the *BACnetUaMapper*.

7.3.3.3 Object Structured_Object_List

Structured_Object_List gathers the references to all BACnet structured view objects of the BACnet device provided through the BACnet property Structured_Object_List.

The OPC UA *Property Structured_Object_List* on the Structured_Object_List object, of *DataType BACnetObjectIdentifier*[], represents the BACnet property Structured_Object_List. The *BACnetObjectIdentifier* DataType is defined in 10.2.1. It contains a list of BACnet structured view objects within the device.

The list of *BACnetStructuredViewTypes* on the Structured_Object_List object contains the list of BACnet structured view objects within the device. The list may be limited through configuration of the *BACnetUaMapper*.

7.3.3.4 Object TimeManagement

This optional OPC UA *Object*, of type *BACnetTimeManagementType*, provides localized time and date information to consumers of the device object.

If an automatic time synchronization master is available, the special subtype *BACnetAutomaticTimeSynchronizationMasterType* is used.

7.3.3.5 Object BackupRestore

This optional OPC UA *Object*, of type *BACnetBackupRestoreType*, provides backup and restore status to consumers of the device object. This status provides information about the last restore time, backup failures, preparation and completion times, and so on.

7.3.3.6 Object MstpMaster

This optional OPC UA *Object*, of type *BACnetMstpMasterType*, describes paramters that are relevant if the device objects functions as a master on an MS/TP network.

7.3.3.7 Object DeviceRestart

This optional OPC UA *Object*, of type *BACnetDeviceRestartType*, provides information related to restarting the device. The last restart time and the set of devices that are notified when the device is restarted are provided by this object.

7.3.3.8 Notifier Objects

OPC UA *Objects* that have the *EventNotifier Attribute* set to *SubscribeToEvents* are so called event notifiers. They can be used by OPC UA *Clients* to subscribe for *Events* from the OPC UA *Server*. The OPC UA *ReferenceType HasNotifier* is used to build an event notifier tree starting from the *Server Object*.

BACnetNotifierType Objects like instances of *BACnetNotificationClassTypes* are used together with instances of the *BACnetDeviceType* to represent the OPC UA event notifier hierarchy starting from the Server object.

Figure 12 shows an example of BACnet objects referencing each other with BACnetObjectIdentifiers and the representation with OPC UA *Objects* and *References*. The example contains a BACnet Device object, one Analog Input object that creates alarms and therefore references a Notification Class object.

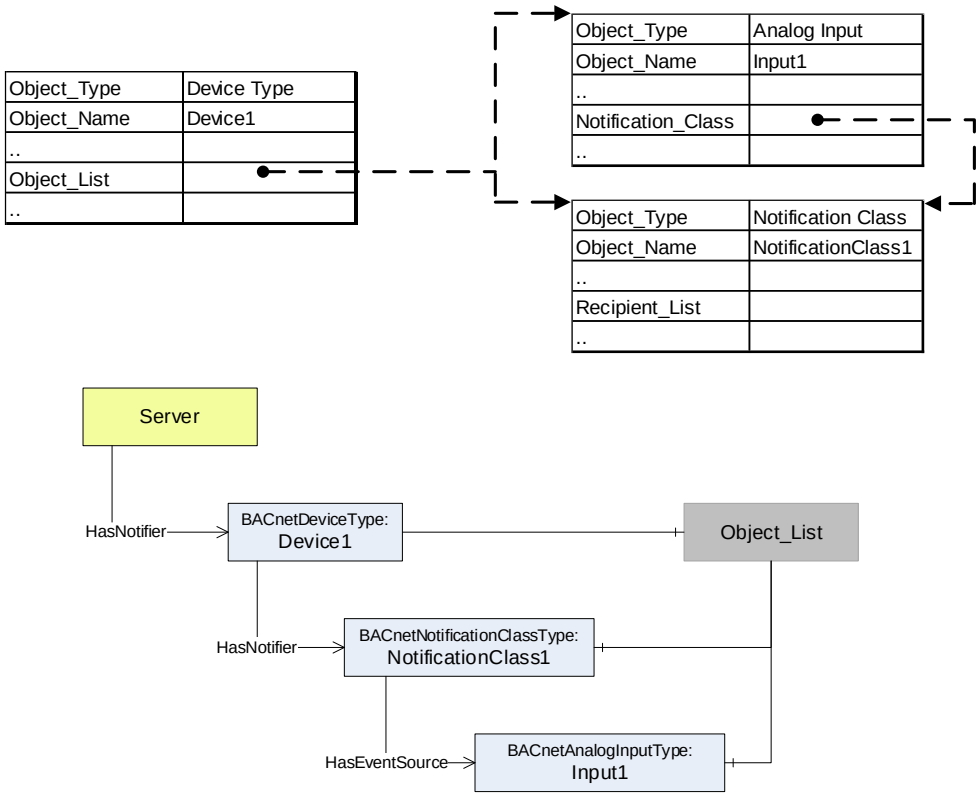


Figure 12 – Event notifiers in BACnet and OPC UA

7.3.3.9 Variable System_Status

This OPC UA *Property*, of *DataType BACnetDeviceStatus*, represents the BACnet property System_Status. The *BACnetDeviceStatus DataType* is defined in 10.4.9.

The property is an enumeration that describes the current operating state of the device. Some example states are “operational”, “non-operational”, “backup in progress”, and so forth.

7.3.3.10 Variable Vendor_Name

This OPC UA *Property*, of *DataType String*, represents the BACnet property Vendor_Name.

This property identifies the manufacturer of the BACnet device.

7.3.3.11 Variable Vendor_Identifier

This OPC UA *Property*, of *DataType UInt16*, represents the BACnet property Vendor_Identifier.

This property represents a unique identification code, assigned by ASHRAE. The code is used to identify proprietary extensions to the protocol.

7.3.3.12 Variable Model_Name

This OPC UA *Property*, of *DataType String*, represents the BACnet property Model_Name.

This property identifies the model name of the BACnet device.

7.3.3.13 Variable Serial_Number

This OPC UA *Property*, of *DataType String*, represents the BACnet property Serial_Number.

This property identifies the serial number of the BACnet device.

7.3.3.14 Variable Firmware_Revision

This OPC UA *Property*, of *DataType String*, represents the BACnet property Firmware_Revision.

This property identifies the firmware installed in the BACnet device.

7.3.3.15 Variable Application_Software_Version

This OPC UA *Property*, of *DataType String*, represents the BACnet property Application_Software_Version.

This property identifies the application software version installed in the BACnet device.

7.3.3.16 Variable Location

This optional OPC UA *Property*, of *DataType String*, represents the BACnet property Location.

This property identifies the physical location of the BACnet device.

7.3.3.17 Variable Protocol_Version

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Protocol_Version.

This property identifies the major version of the BACnet protocol supported by this device.

7.3.3.18 Variable Protocol_Revision

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Protocol_Revision.

7.3.3.19 Variable Protocol_Services_Supported

This OPC UA *Property*, of *DataType BACnetServicesSupportedBits*, represents the BACnet property Protocol_Services_Supported. The *BACnetServicesSupportedBits DataType* is defined in 10.3.6.

The BACnetServicesSupportedBits identifies the standardized protocol services that are executed by the implementation of the protocol on this device. See the BACnet specification for minimum supported service requirements.

7.3.3.20 Variable Protocol_Object_Types_Supported

This OPC UA *Property*, of *DataType BACnetObjectTypeSupportedBits*, represents the BACnet property Protocol_Object_Types_Supported. The *BACnetObjectTypeSupportedBits DataType* is defined in 10.3.5.

This property identifies the standardized BACnet object types (like analog inputs, binary outputs, calendars, and so on) that can be represented by this device's implementation of the BACnet protocol. The minimum set of supported objects shall be at least Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, and Binary Value.

7.3.3.21 Variable Max_APDU_Length_Accepted

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Max_APDU_Length_Accepted.

This property describes the maxium number of octets that may be packaged into a single application layer protocol data unit. The data type of this variable is an unsigned integer and its value shall be greater than or equal to 50.

7.3.3.22 Variable Segmentation_Supported

This OPC UA *Property*, of *DataType BACnetSegmentation*, represents the BACnet property Segmentation_Supported. The *BACnetSegmentation DataType* is defined in 10.4.30.

The BACnetSegmentation is an enumeration that indicates whether the BACnet device supports message segmentation and, if so, whether it supports segmented transmission, reception, or both.

7.3.3.23 Variable Max_Segments_Accepted

This optional OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Max_Segments_Accepted.

The Max_Segments_Accepted property indicates the maximum number of segments of an APDU that the device will accept.

7.3.3.24 Variable APDU_Segment_Timeout

This optional OPC UA *Property*, of *DataType UInteger*, represents the BACnet property APDU_Segment_Timeout.

The APDU_Segment_Timeout property indicates the amount of time (in milliseconds) between retransmission of an APDU segment. See the BACnet specification for recommended default values and best practices.

7.3.3.25 Variable APDU_Timeout

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property APDU_Timeout.

The APDU_Timeout property indicates the amount of time (in milliseconds) between retransmissions of an APDU that requires acknowledgement, but for which no acknowledgement has been received. See the BACnet specification for recommended default values and best practices.

7.3.3.26 Variable Number_Of_APDU_Retries

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Number_Of_APDU_Retries.

The Number_Of_APDU_Retries property indicates the maximum number of times that an ADPU is retransmitted. See the BACnet specification for recommended default values and best practices.

7.3.3.27 Variable Device_Address_Binding

This OPC UA *Property*, containing an array of *DataType BACnetAddressBinding*, represents the BACnet property Device_Address_Binding. The *BACnetAddressBinding DataType* is defined in 10.5.3.

The BACnetAddressBinding data type identify the actual device address that will be used when the remote device must be accessed using a BACnet service request. See the BACnet specification for rules related to zero network-number addresses and empty lists.

7.3.3.28 Variable Database_Revision

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Database_Revision.

The Database_Revision property describes a logical revision number for the device database. The revision is incremented when an object is created, an object is deleted, an object's name is changed, an object's Object_Identifier property is changed, or a restore is performed. See the BACnet specification for exceptions.

7.3.3.29 Variable Active_COV_Subscriptions

This OPC UA *Property*, containing an array of *DataType BACnetCOVSubscription*, represents the BACnet property Active_COV_Subscriptions. The *BACnetCOVSubscription DataType* is defined in 10.5.4.

The BACnetCOVSubscription value provides a network-visible indication of those COV subscriptions that are active at any given time. When a COV subscription is created using the BACnet COV subscription services, a ew entry is added to the Active_COV_Subscriptions list. The entry is removed when the subscription is terminated.

7.3.3.30 Method CreateObject

This method represents the BACnet service CreateObject. It is used to create a new instance of an object. The BACnet properties of standard objects created with this Method may be initialized in two ways: initial values may be provided as part of the *CreateObject Method* call or values may be written to the newly created object using OPC UA *Write*.

Signature

```
CreateObject (
   [in]  BACnetObjectIdentifier      ObjectSpecifier
   [in]  0:KeyValuePair[]            ListOfInitialValues
   );
```

Argument	Description
ObjectSpecifier	Provides the information about the BACnet object type to be created or the BACnet object identifier if the object to be created. The instance information shall be undefined if only the object type is passed in. The *BACnetObjectIdentifier DataType* is defined in 10.2.1.
ListOfInitialValues	A list of initial values that shall be used to initialize the values of the specified properties of the newly created object.

Method Result Codes

ResultCode	Description
BadOutOfMemory	This status is returned for the BACnet error code NO_SPACE_FOR_OBJECT
BadNotSupported	This status is returned for the BACnet error code DYNAMIC_CREATION_NOT_SUPPORTED
BadTypeDefinitionInvalid	This status is returned for the BACnet error code UNSUPPORTED_OBJECT_TYPE
BadNodeIdInvalid	This status is returned for the BACnet error code OBJECT_IDENTIFIER_ALREADY_EXISTS
BadTypeMismatch	This status is returned for the BACnet error code INVALID_DATATYPE
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNodeIdInvalid	This status is returned for the BACnet error code UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error code CHARACTER_SET_NOT_SUPPORTED
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadTypeMismatch	This status is returned for the BACnet error code DATATYPE_NOT_SUPPORTED

7.3.3.31 Method DeleteObject

This method represents the BACnet service DeleteObject. It can be used to delete objects that may be created and deleted dynamically.

Signature

```
DeleteObject (
   [in]  BACnetObjectIdentifier      ObjectIdentifier
   );
```

Argument	Description
ObjectIdentifier	Specifies the BACnet object to be deleted. The *BACnetObjectIdentifier DataType* is defined in 10.2.1.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error code UNKNOWN_OBJECT
BadUserAccessDenied	This status is returned for the BACnet error code OBJECT_DELETION_NOT_PERMITTED

7.3.3.32 Method ReinitializeDevice

This method represents the BACnet service ReinitializeDevice. It is used to instruct a remote device to reboot itself (cold start), reset itself to some predefined initial state (warm start), or to control the backup or restore procedure. Resetting or rebooting a device is primarily initiated by a human operator for diagnostic purposes.
Remark: Since the method contains the Password, this method should only be available if the connection is encrypted.

Signature

```
ReinitializeDevice (
   [in]  BACnetReinitializedStateofDevice ReinitializedStateofDevice
   [in]  0:String                          Password
   );
```

Argument	Description
ReinitializedStateofDevice	This parameter allows the caller to specify the desired state of the device after its reinitialization. The *BACnetReinitializedStateofDevice DataType* is defined in 10.4.27.
Password	password parameter.

Method Result Codes

ResultCode	Description
BadInvalidState	This status is returned for the BACnet error code CONFIGURATION_IN_PROGRESS
BadUserAccessDenied	This status is returned for the BACnet error code PASSWORD_FAILURE
BadNoCommunication	This status is returned for the BACnet error code COMMUNICATION_DISABLED

7.3.3.33 Method DeviceCommunicationControl

This method represents the BACnet service DeviceCommunicationControl. It is used to instruct a remote device to stop initiating communication and optionally stop responding to communication for a specified duration of time.
Remark: Since the method contains the Password, this method should only be available if the connection is encrypted.

Signature

```
DeviceCommunicationControl (
   [in]  0:UInt16                          TimeDurationInMinutes
   [in]  BACnetDeviceCommunicationEnabled  EnableDisable
   [in]  0:String                          Password
   );
```

Argument	Description
TimeDurationInMinutes	This optional parameter of DataType 0:UInt16 indicates the number of minutes that the remote device shall stop communication. If the parameter is not specified, 0 shall be passed in as value.
EnableDisable	This parameter is an enumeration that may take on the values ENABLE, DISABLE, or DISABLE_INITIATION. It is used to indicate whether the device should enable all, disable initiation, or disable all communications on the network interface. The *BACnetDeviceCommunicationEnabled DataType* is defined in 10.4.8.
Password	password parameter.

Method Result Codes

ResultCode	Description
BadUserAccessDenied	This status is returned for the BACnet error code PASSWORD_FAILURE
BadNotSupported	This status is returned for the BACnet error code O_FUNCTIONALITY_NOT_SUPPORTED

7.3.3.34 Method TextMessage

This method represents the BACnet services ConfirmedTextMessage and UnconfirmedTextMessage.
The UnconfirmedTextMessage service is used by a client BACnet-user to send a text message to one or more BACnet devices. This service may be broadcast, multicast, or addressed to a single recipient. This service may be used in cases where confirmation that the text message was received is not required. Messages may be prioritized into normal or urgent categories. In addition, a given text message may optionally be classified by a numeric class code or class identification string. This classification may be used by receiving BACnet devices to determine how to handle the text message. For example, the message class might indicate a particular output device on which to print text or a set of actions to take when the text message is received. In any case, the interpretation of the class is a local matter.
The ConfirmedTextMessage service is used by a client BACnet-user to send a text message to another BACnet device. This service is not a broadcast or multicast service. This service may be used in cases when confirmation that the text message was received is required. The confirmation does not guarantee that a human operator has seen the message. Messages may be prioritized into normal or urgent categories. In addition, a given text message may be optionally classified by a numeric class code or class identification string. This classification may be used by the receiving BACnet device to determine how to handle the text message. For example, the message class might indicate a particular output device on which to print text or a set of actions to take when the text is received. In any case, the interpretation of the class is a local matter.

Signature

```
TextMessage (
   [in]  0:Boolean                  SendUnconfirmed
   [in]  BACnetObjectIdentifier     TextMessageSourceDevice
   [in]  BACnetMessageClass         MessageClass
   [in]  BACnetMessagePriority      MessagePriority
   [in]  0:String                   Message
   );
```

Argument	Description
SendUnconfirmed	Flag indicating if the BACnet service UnconfirmedTextMessage (TRUE) or ConfirmedTextMessage (FALSE) is used.
TextMessageSourceDevice	This parameter, of type BACnetObjectIdentifier, shall convey the value of the Object_Identifier property of the Device object of the device that initiated this text message.
MessageClass	This parameter, if present, shall indicate the classification of the received message. The datatype of this parameter shall be a choice of Unsigned or CharacterString. The interpretation of the meaning of any particular value for this parameter shall be a local matter.
MessagePriority	This parameter, of type ENUMERATED, shall indicate the priority for message handling: {NORMAL, URGENT}
Message	This parameter, of type CharacterString, shall be used to convey the text message.

Method Result Codes

ResultCode	Description
BadUserAccessDenied	This status is returned for the BACnet error code PASSWORD_FAILURE
BadNotSupported	This status is returned for the BACnet error code O_FUNCTIONALITY_NOT_SUPPORTED

7.3.3.35 Method AddDeviceAddressBindings

This Method adds entries to the BACnet property Device_Address_Binding.

Signature

```
AddDeviceAddressBindings (
   [in]  BACnetAddressBinding []          AddressBindings
   [out] 0:UInt32                         FirstFailedElementNumber
   );
```

Argument	Description
AddressBindings	Array of address bindings to add to the entries in the BACnet property Device_Address_Binding. The BACnetAddressBinding DataType is defined in 10.5.3.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the AddressBindings. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE and DATATYPE_NOT_SUPPORTED
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadOutOfMemory	This status is returned for the BACnet error code NO_SPACE_TO_ADD_LIST_ELEMENT

7.3.3.36 Method RemoveDeviceAddressBindings

This Method removes entries from the BACnet property Device_Address_Binding.

Signature

```
RemoveDeviceAddressBindings (
   [in]  BACnetAddressBinding[]           AddressBindings
   [out] 0:UInt32                         FirstFailedElementNumber
   );
```

Argument	Description
AddressBindings	Array of address bindings to remove from the entries in the BACnet property Device_Address_Binding. The BACnetAddressBinding DataType is defined in 10.5.3.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the AddressBindings. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadNotFound	This status is returned for the BACnet error code LIST_ELEMENT_NOT_FOUND

7.4 BACnetAnalogType

7.4.1 General

This OPC UA *ObjectType* defines the super type for all OPC UA *ObjectTypes* that represent *BACnet Analog* object types in an OPC UA *Address Space*.

Figure 13 shows an overview for the *BACnetAnalogType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 12.

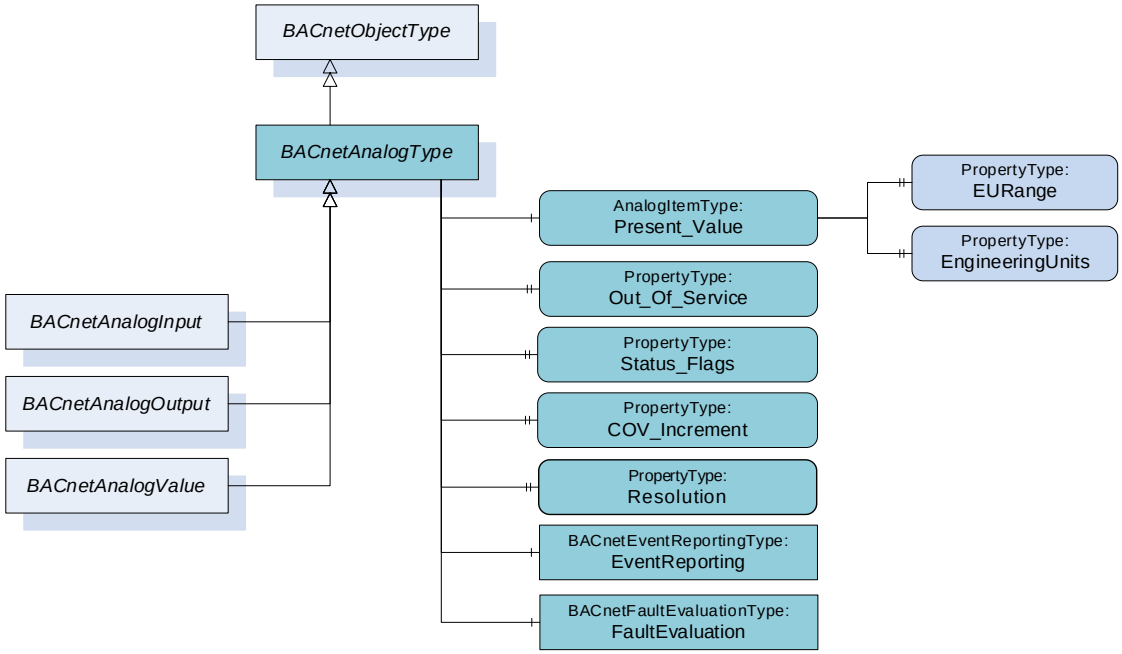


Figure 13 – *BACnetAnalogType overview*

7.4.2 ObjectType definition

The *BACnetAnalogType* is formally defined in Table 12.

Table 12 – *BACnetAnalogType* Definition

Attribute	**Value**				
BrowseName	BACnetAnalogType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1					
0:HasComponent	Variable	Present_Value	0:Float	0:AnalogUnitType	M
0:HasProperty	Variable	Out_Of_Service	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Status_Flags	BACnetStatusFlags	0:PropertyType	M
0:HasProperty	Variable	Resolution	0:Float	0:PropertyType	O
0:HasProperty	Variable	COV_Increment	0:Float	0:PropertyType	O
0:HasComponent	Object	EventReporting		BACnetEventReportingType	O
0:HasComponent	Object	FaultEvaluation		BACnetFaultEvaluationType	O

The *BACnetAnalogType* is an abstract type and cannot be used directly.

The components of the *BACnetAnalogType* have additional subcomponents which are defined in Table 13.

Table 13 – BACnetAnalogType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetOutOfRange AlgorithmType	M

7.4.3 ObjectType Description

7.4.3.1 Variable Present_Value

This OPC UA *Variable*, of *DataType Float*, indicates the current value, in engineering units, of the input being measured. The *Present_Value Variable* shall be writable when *Out_Of_Service* is true.

This OPC UA *AnalogItemType Variable* represents the BACnet properties Present_Value, Min_Pres_Value, Max_Pres_Value, Units and Update_Interval. The OPC UA *VariableType AnalogItemType* is defined in OPC 10000-8. It defines the OPC UA *Properties EURange*, *EngineeringUnits* and *InstrumentRange*.

The following list provides the BACnet property and the mapping to the corresponding data member in the OPC UA AnalogItem.

- Present_Value represented by *Value Attribute* of the *Variable*.
- Min_Pres_Value represented by *Low* part of *EURange Property* of the *Variable*. If the optional BACnet property Min_Pres_Value is not present, the *EURange Property* should not be provided.
- Max_Pres_Value represented by the *High* part of *EURange Property* of the *Variable*. If the optional BACnet property Max_Pres_Value is not present, the *EURange Property* should not be provided.
- Units represented by the *EngineeringUnits Property* of the *Variable*. The mapping of BACnet units to OPC UA units is defined in 11.
- Update_Interval represented by *MinSamplingInterval Attribute* of the *Variable*.

7.4.3.2 Variable Out_Of_Service

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Out_Of_Service.

It is an indication of whether or not the physical input that the object represents is not in service. While *Out_Of_Service* is True, *Present_Value* and *Reliability* may be changed to any value as a means of simulation and/or testing.

7.4.3.3 Variable Status_Flags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

It represents four *Boolean* flags that represent the general health of a *BACnetAnalogType*. The flags are IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE.

7.4.3.4 Variable COV_Increment

This OPC UA *Property*, of type *Float*, represents the BACnet property COV_Increment.

It shall specify the minimum change in the BACnet property Present_Value that will cause a Change of Value notification to be issued to BACnet clients that have subscribed to Change of Value events.

7.4.3.5 Variable Resolution

This OPC UA *Property*, of *DataType Float*, represents the BACnet property Resolution.

It indicates the smallest recognizable change in Present_Value in engineering units.

7.4.3.6 Object EventReporting

The *EventReporting Object* contains status and configuration information for the event reporting of *BACnetAnalogTypes* and their subtypes. The *BACnetEventReportingType* is defined in 8.8. The *Object* is optional and is not present if event generation is not activated for the BACnet object.

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetOutOfRangeAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetAnalogType* or one of its subtypes. The *BACnetOutOfRangeAlgorithmType* is defined in 8.9.3.

On the *EventAlgorithm* instance of the *BACnetOutOfRangeAlgorithmType* the BACnet property High_Limit is mapped to pHighLimit, the BACnet property Low_Limit is mapped to LowLimit, the BACnet property Deadband is mapped to pDeadband and the BACnet property Limit_Enable is mapped to LimitEnable.

7.4.3.7 Object FaultEvaluation

The *FaultEvaluation Object* contains status and configuration information for the fault evaluation of the *BACnetAnalogType*. The *BACnetFaultEvaluationType* is defined in 8.23. The *Object* is optional and is not present if fault evaluation is not activated for the BACnet object.

7.5 BACnetAnalogInputType

7.5.1 General

This OPC UA *ObjectType* represents the *BACnet* object type *Analog Input*. An analog input converts a continuously variable input signal into a discrete value that can be processed by a computer system.

Figure 14 shows an overview for the *BACnetAnalogInputType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 14.

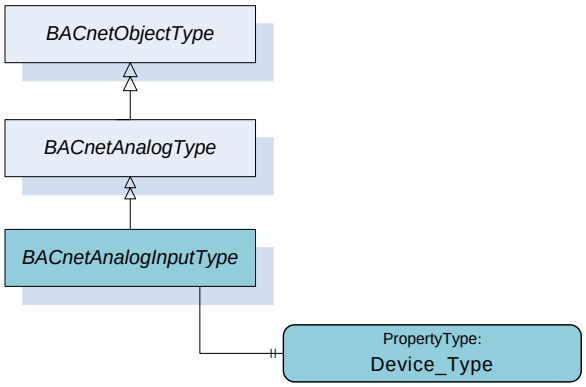


Figure 14 – *BACnetAnalogInputType overview*

7.5.2 ObjectType definition

The *BACnetAnalogInputType ObjectType* is formally defined in Table 14.

Table 14 – *BACnetAnalogInputType* Definition

Attribute	**Value**				
BrowseName	BACnetAnalogInputType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetAnalogType*					
0:HasProperty	Variable	Device_Type	0:String	0:PropertyType	O

The *BACnetAnalogInputType ObjectType* is a concrete type and can be used directly.

7.5.3 ObjectType Description

7.5.3.1 Variable Device_Type

This OPC UA *Property*, of *DataType String*, represents the BACnet property Device_Type.

It is a text description of the physical device connected to the analog input.

7.6 BACnetAnalogOutputType

7.6.1 General

This OPC UA *ObjectType* represents the *BACnet* object type *Analog Output*. An analog output converts a discrete value into a continuously variable output signal.

Figure 15 shows an overview for the *BACnetAnalogOutput* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 15.

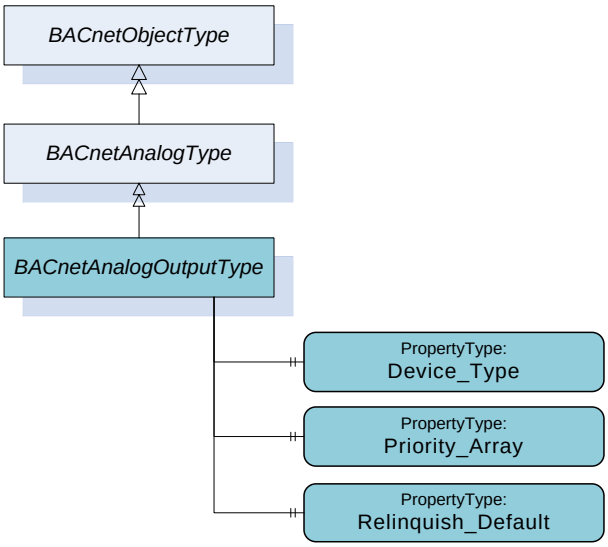


Figure 15 – *BACnetAnalogOutputType overview*

7.6.2 ObjectType definition

The *BACnetAnalogOutput ObjectType* is formally defined in Table 15.

Table 15 – *BACnetAnalogOutputType* Definition

Attribute	**Value**				
BrowseName	BACnetAnalogOutputType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetAnalogType*					
0:HasProperty	Variable	Device_Type	0:String	0:PropertyType	O
0:HasProperty	Variable	Priority_Array	BACnetPriorityValue [16]	0:PropertyType	M
0:HasProperty	Variable	Relinquish_Default	0:Float	0:PropertyType	M

The *BACnetAnalogOutput ObjectType* is a concrete type and can be used directly.

7.6.3 ObjectType Description

7.6.3.1 Variable Device_Type

This OPC UA *Property*, of *DataType String*, represents the BACnet property Device_Type.

It is a text description of the physical device connected to the analog output.

7.6.3.2 Variable Priority_Array

This OPC UA Property, of *DataType BACnetPriorityValue* [16], represents the BACnet property Priority_Array. The *BACnetPriorityValue DataType* is defined in 10.6.4.

It is an array that contains prioritized values that are in effect for this object. See 3.2.1 for details on command prioritization.

7.6.3.3 Variable Relinquish_Default

This OPC UA *Property*, of *DataType Float*, represents the BACnet property Relinquish_Default.

It is the default value to be used for the Present_Value when all command priority values in the Priority_Array have a NULL value. See 3.2.1 for details on command prioritization.

7.7 BACnetAnalogValueType

7.7.1 General

This OPC UA *ObjectType* represents the *BACnet* object type *Analog Value*. This is a control system parameter that resides in the memory of the *BACnet Device*.

Figure 16 shows an overview for the *BACnetAnalogValueType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 16.

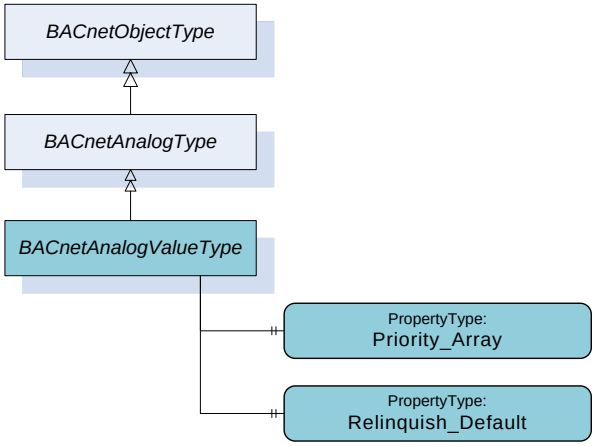


Figure 16 – *BACnetAnalogValueType overview*

7.7.2 ObjectType definition

The *BACnetAnalogValueType ObjectType* is formally defined in Table 16.

Table 16 – *BACnetAnalogValueType* Definition

Attribute	**Value**				
BrowseName	BACnetAnalogValueType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetAnalogType*					
0:HasProperty	Variable	Priority_Array	BACnetPriorityValue [16]	0:PropertyType	O
0:HasProperty	Variable	Relinquish_Default	0:Float	0:PropertyType	O

The *BACnetAnalogValueType ObjectType* is a concrete type and can be used directly.

7.7.3 ObjectType Description

7.7.3.1 Variable Priority_Array

This OPC UA Property, of *DataType BACnetPriorityValue* [16], represents the BACnet property Priority_Array. The *BACnetPriorityValue DataType* is defined in 10.6.4.

It is an array that contains prioritized commands that are in effect for this object. See 3.2.1 for details on command prioritization.

7.7.3.2 Variable Relinquish_Default

This OPC UA *Property*, of *DataType 0:Float*, represents the BACnet property Relinquish_Default.

It is the default value to be used for the Present_Value when all command priority values in the Priority_Array have a NULL value. See 3.2.1 for details on command prioritization.

7.8 BACnetBinaryType

7.8.1 General

This OPC UA *ObjectType* defines the super type for all OPC UA *ObjectTypes* that represent *BACnet Binary* object types in an OPC UA *Address Space*.

Figure 17 shows an overview for the *BACnetBinaryType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 17.

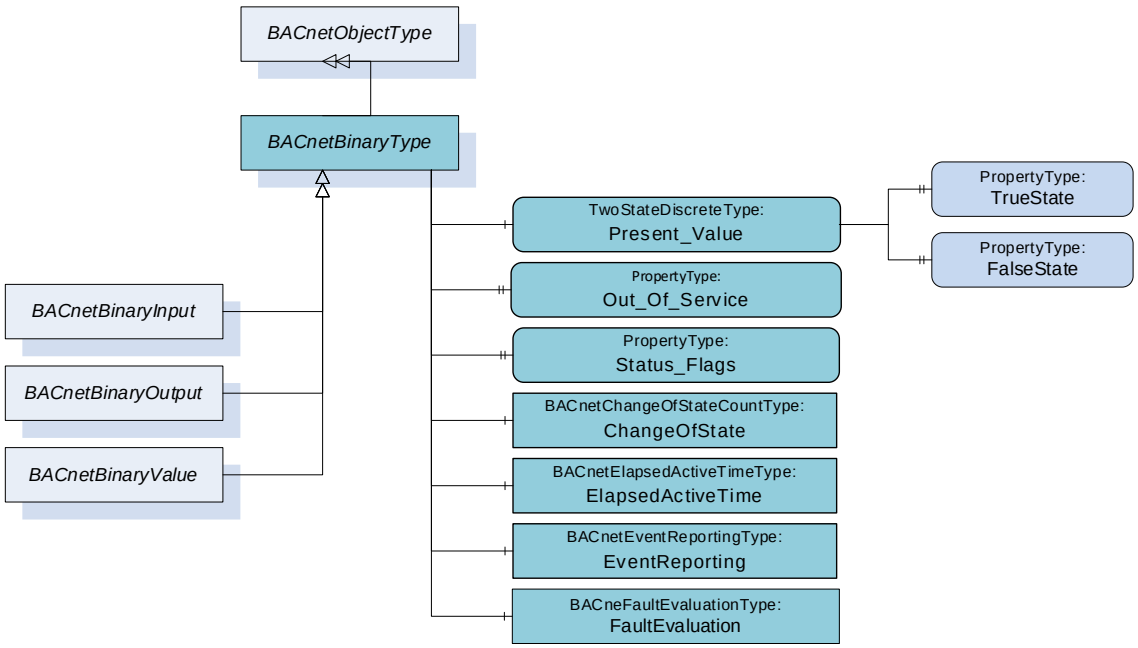


Figure 17 – *BACnetBinaryType overview*

7.8.2 ObjectType definition

The *BACnetBinaryType ObjectType* is formally defined in Table 17.

Table 17 – *BACnetBinaryType* Definition

Attribute	Value				
BrowseName	BACnetBinaryType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasComponent	Variable	Present_Value	0:Boolean	TwoStateDiscreteType	M
0:HasProperty	Variable	Out_Of_Service	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Status_Flags	BACnetStatusFlags	0:PropertyType	M
0:HasComponent	Object	ChangeOfState		BACnetChangeOfStateCountType	O
0:HasComponent	Object	ElapsedActiveTime		BACnetElapsedActiveTimeType	O
0:HasComponent	Object	EventReporting		BACnetEventReportingType	O
0:HasComponent	Object	FaultEvaluation		BACnetFaultEvaluationType	O

The *BACnetBinaryType ObjectType* is an abstract type and cannot be used directly.

7.8.3 ObjectType Description

7.8.3.1 Variable Present_Value

This OPC UA *Variable*, of *DataType 0:Boolean*, represents the logical state of the BACnet Binary Input object. The logical state of the Input shall be either Inactive or Active.

This OPC UA *TwoStateDiscreteType Variable* represents the BACnet properties Present_Value, Inactive_Text and Active_Text. The OPC UA *VariableType TwoStateDiscreteType* is defined in OPC 10000-8. It defines the OPC UA *Properties TrueState* and *FalseState*.

The following list provides the BACnet property and the mapping to the corresponding data member in the OPC UA *TwoStateDiscrete* item.

- Present_Value represented by *Value Attribute* of the *Variable*.
- Active_Text represented by the *TrueState Property* of the *Variable*.
- Inactive_Text represented by the *FalseState Property* of the *Variable*.

7.8.3.2 Variable Out_Of_Service

This OPC UA *Property*, of *DataType 0:Boolean*, represents the BACnet property Out_Of_Service.

It is an indication of whether or not the physical input that the object represents is not in service. While the Out_Of_Service property is True, the Present_Value and Reliability properties may be changed to any value as a means of simulation and/or testing.

7.8.3.3 Variable Status_Flags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

It represents four 0:Boolean flags that represent the general health of an analog input. The flags are IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE.

7.8.3.4 Object ChangeOfState

The *ChangeOfState Object* contains BACnet properties related to counting the change of state of *BACnetBinaryType Objects*. The *BACnetChangeOfStateCountType* is defined in 8.1. The *Object* is optional and is not present if change of state counting is not activated for the BACnet object.

7.8.3.5 Object ElapsedActiveTime

The *ElapsedActiveTime Object* contains BACnet properties related to the elapsed active time of *BACnetBinaryType Objects*. The *BACnetElapsedActiveTimeType* is defined in 8.7. The *Object* is optional and is not present if this feature is not available for the BACnet object.

7.8.3.6 Object EventReporting

The *EventReporting Object* contains status and configuration information for the event reporting of *BACnetBinaryType Objects*. The *BACnetEventReportingType* is defined in 8.8. The *Object* is optional and is not present if event generation is not activated for the BACnet object.

7.8.3.7 Object FaultEvaluation

The *FaultEvaluation Object* contains status and configuration information for the fault evaluation of the *BACnetBinaryType*. The *BACnetFaultEvaluationType* is defined in 8.23. The *Object* is optional and is not present if fault evaluation is not activated for the BACnet object.

7.9 BACnetBinaryInputType

7.9.1 General

This OPC UA *ObjectType* represents the *BACnet* object type *Binary Input*. A binary input converts one or more discrete input signals into a value that can be processed by a computer system.

Figure 18 shows an overview for the *BACnetBinaryInputType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 18.

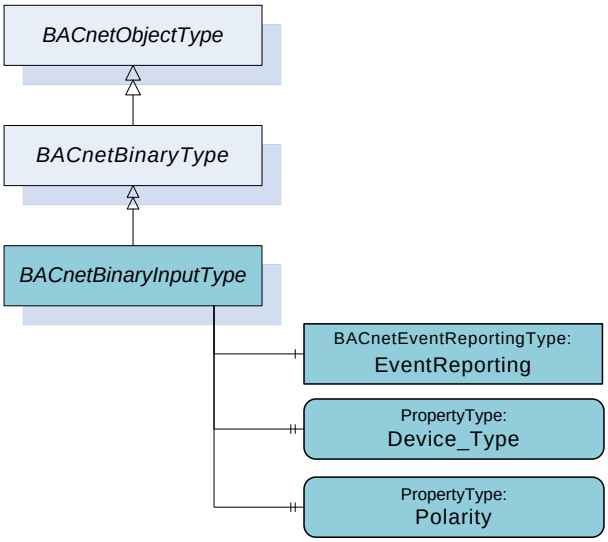


Figure 18 – *BACnetBinaryInputType overview*

7.9.2 ObjectType definition

The *BACnetBinaryInputType ObjectType* is formally defined in Table 18.

Table 18 – *BACnetBinaryInputType* Definition

Attribute	Value				
BrowseName	BACnetBinaryInputType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetBinaryType* defined in 7.8.					
0:HasComponent	Object	EventReporting		BACnetEventReportingType	O
0:HasProperty	Variable	Device_Type	0:String	0:PropertyType	O
0:HasProperty	Variable	Polarity	BACnetPolarity	0:PropertyType	M

The *BACnetBinaryInputType ObjectType* is a concrete type and can be used directly.

The components of the *BACnetBinaryInputType* have additional subcomponents which are defined in Table 19.

Table 19 – BACnetBinaryInputType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetChangeOfStateAlgorithmType	M
EventReporting EventAlgorithm	0:HasProperty	Variable	AlarmValues	0:Boolean	0:PropertyType	M

7.9.3 ObjectType Description

7.9.3.1 EventReporting Object override

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetChangeOfStateAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetBinaryInputType*. The *BACnetChangeOfStateAlgorithmType* is defined in 8.10.

On the EventAlgorithm instance of the *BACnetChangeOfStateAlgorithmType*, the *DataType* of the *Alarm_Values Property* is changed to 0:Boolean.

The BACnet property Alarm_Value of the BACnet object type Binary Input is mapped to the *AlarmValues Property* of the *BACnetChangeOfStateAlgorithmType*.

7.9.3.2 Variable Device_Type

This OPC UA *Property*, of *DataType String*, represents the BACnet property Device_Type.

It is a text description of the physical device connected to the binary input.

7.9.3.3 Variable Polarity

This OPC UA *Property*, of *DataType* BACnetPolarity, represents the BACnet property Polarity. The *BACnetPolarity DataType* is defined in 10.4.22.

It indicates the relationship between the physical state of the Input and the logical state represented by the Present_Value property.

7.10 BACnetBinaryOutputType

7.10.1 General

This OPC UA *ObjectType* represents the *BACnet* object type *Binary Output*. A binary output converts a value into one or more discrete output signals.

Figure 19 shows an overview for the *BACnetBinaryOutputType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 20.

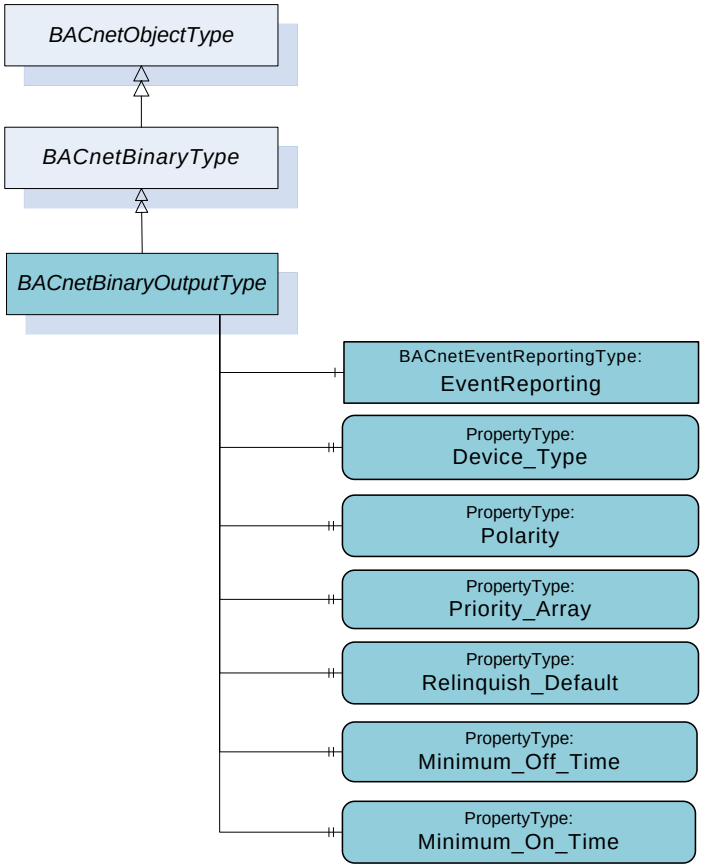


Figure 19 – *BACnetBinaryOutputType overview*

7.10.2 ObjectType definition

The *BACnetBinaryOutputType ObjectType* is formally defined in Table 20.

Table 20 – *BACnetBinaryOutputType* Definition

Attribute	Value				
BrowseName	BACnetBinaryOutputType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetBinaryType* defined in 7.8.					
0:HasComponent	Object	EventReporting		BACnetEvent ReportingType	O
0:HasProperty	Variable	Device_Type	0:String	0:PropertyType	O
0:HasProperty	Variable	Polarity	BACnetPolarity	0:PropertyType	M
0:HasProperty	Variable	Priority_Array	BACnetPriorityValue [16]	0:PropertyType	M
0:HasProperty	Variable	Relinquish_Default	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Minimum_Off_Time	0:UInt32	0:PropertyType	O
0:HasProperty	Variable	Minimum_On_Time	0:UInt32	0:PropertyType	O
0:HasProperty	Variable	Feedback_Value	0:Boolean	0:PropertyType	O

The *BACnetBinaryOutputType ObjectType* is a concrete type and can be used directly.

The components of the *BACnetBinaryOutputType* have additional subcomponents which are defined in Table 21.

Table 21 – BACnetBinaryOutputType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetCommandFailureAlgorithmType	M

7.10.3 ObjectType Description

7.10.3.1 EventReporting Object override

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetCommandFailureAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetBinaryOutputType*. The *BACnetCommandFailureAlgorithmType* is defined in 8.10.

7.10.3.2 Variable Device_Type

This OPC UA *Property*, of *DataType String*, represents the BACnet property Device_Type.

It is a text description of the physical device connected to the binary output.

7.10.3.3 Variable Polarity

This OPC UA *Property*, of *DataType* BACnetPolarity, represents the BACnet property Polarity. The *BACnetPolarity DataType* is defined in 10.4.22.

It indicates the relationship between the physical state of the Ouput and the logical state represented by the Present_Value property.

7.10.3.4 Variable Priority_Array

This OPC UA Property, of *DataType* BACnetPriorityValue [16], represents the BACnet property Priority_Array. The *BACnetPriorityValue DataType* is defined in 10.6.4.

It is an array that contains prioritized commands that are in effect for this object. See 3.2.1 for details on command prioritization.

7.10.3.5 Variable Relinquish_Default

This OPC UA *Property*, of *DataType 0:Boolean*, represents the BACnet property Relinquish_Default.

It is the default value to be used for the Present_Value when all command priority values in the Priority_Array have a NULL value. See 3.2.1 for details on command prioritization.

7.10.3.6 Variable Minimum_Off_Time

This OPC UA Property, of *DataType UInt32*, represents the BACnet property Minimum_Off_Time.

It represents the minimum number of seconds that the Present_Value shall remain in the Inactive state after a write to the Present_Value property causes that property to assume the Inactive state.

7.10.3.7 Variable Minimum_On_Time

This OPC UA Property, of *DataType UInt32*, represents the BACnet property Minimum_On_Time.

It represents the minimum number of seconds that the Present_Value shall remain in the Active state after a write to the Present_Value property causes the property to assume the Active state.

7.10.3.8 Variable Feedback_Value

This OPC UA *Property*, of type *0:Boolean*, is an indication of the actual value of the entity controlled by Present_Value.

7.11 BACnetBinaryValueType

7.11.1 General

This OPC UA *ObjectType* represents a *BACnet Binary Value* object type. A binary value is a control system parameter that resides in the memory of the BACnet Device and may be one of only two states.

Figure 20 shows an overview for the *BACnetBinaryValueType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 22.

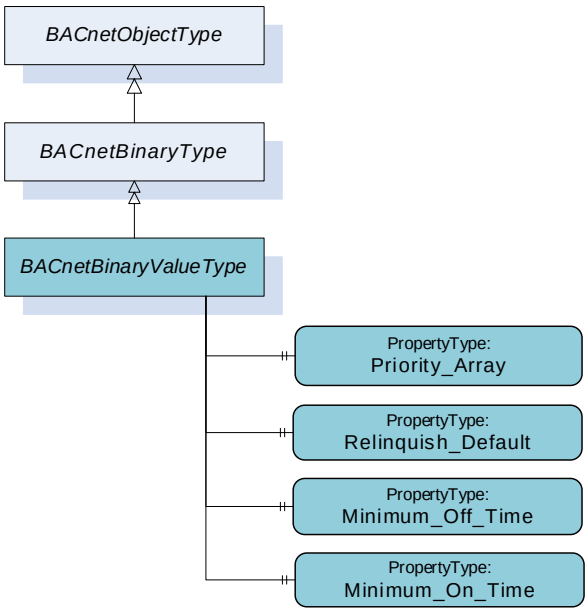


Figure 20 – *BACnetBinaryValueType overview*

7.11.2 ObjectType definition

The *BACnetBinaryValueType ObjectType* is formally defined in Table 22.

Table 22 – *BACnetBinaryValueType* Definition

Attribute	**Value**				
BrowseName	BACnetBinaryValueType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetBinaryType* defined in 7.8.					
0:HasComponent	Object	EventReporting		BACnetEvent ReportingType	O
0:HasProperty	Variable	Priority_Array	BACnetPriorityValue [16]	0:PropertyType	O
0:HasProperty	Variable	Relinquish_Default	0:Boolean	0:PropertyType	O
0:HasProperty	Variable	Minimum_Off_Time	0:UInt32	0:PropertyType	O
0:HasProperty	Variable	Minimum_On_Time	0:UInt32	0:PropertyType	O

The *BACnetBinaryValueType ObjectType* is a concrete type and can be used directly.

The components of the *BACnetBinaryValueType* have additional subcomponents which are defined in Table 23.

Table 23 – BACnetBinaryValueType Additional Subcomponents

BrowsePath	**References**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetChangeOfStateAlgorithmType	M
EventReporting / EventAlgorithm	0:HasProperty	Variable	AlarmValues	0:Boolean	0:PropertyType	M

7.11.3 ObjectType Description

7.11.3.1 EventReporting Object override

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetChangeOfStateAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetBinaryValueType*. The *BACnetChangeOfStateAlgorithmType* is defined in 8.10.

On the EventAlgorithm instance of the *BACnetChangeOfStateAlgorithmType*, the *DataType* of the *Alarm_Values Property* is changed to 0:Boolean.

The BACnet property Alarm_Value of the BACnet object type Binary Value is mapped to the *AlarmValues Property* of the *BACnetChangeOfStateAlgorithmType*.

7.11.3.2 Variable Priority_Array

This OPC UA Property, of *DataType BACnetPriorityValue* [16], represents the BACnet property Priority_Array. The *BACnetPriorityValue DataType* is defined in 10.6.4.

It is an array that contains prioritized commands that are in effect for this object. See 3.2.1 for details on command prioritization.

7.11.3.3 Variable Relinquish_Default

This OPC UA *Property*, of *DataType 0:Boolean*, represents the BACnet property Relinquish_Default.

It is the default value to be used for the Present_Value when all command priority values in the Priority_Array have a NULL value. See 3.2.1 for details on command prioritization.

7.11.3.4 Variable Minimum_Off_Time

This OPC UA Property, of *DataType UInt32*, represents the BACnet property Minimum_Off_Time.

It represents the minimum number of seconds that the Present_Value shall remain in the Inactive state after a write to the Present_Value property causes that property to assume the Inactive state.

7.11.3.5 Variable Minimum_On_Time

This OPC UA Property, of *DataType UInt32*, represents the BACnet property Minimum_On_Time.

It represents the minimum number of seconds that the Present_Value shall remain in the Active state after a write to the Present_Value property causes the property to assume the Active state.

7.12 BACnetMultiStateType

7.12.1 General

This OPC UA *ObjectType* defines the super type for all OPC UA *ObjectTypes* that represent *BACnet MultiState* object types in an OPC UA *Address Space*.

Figure 21 shows an overview for the *BACnetMultiStateType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 24.

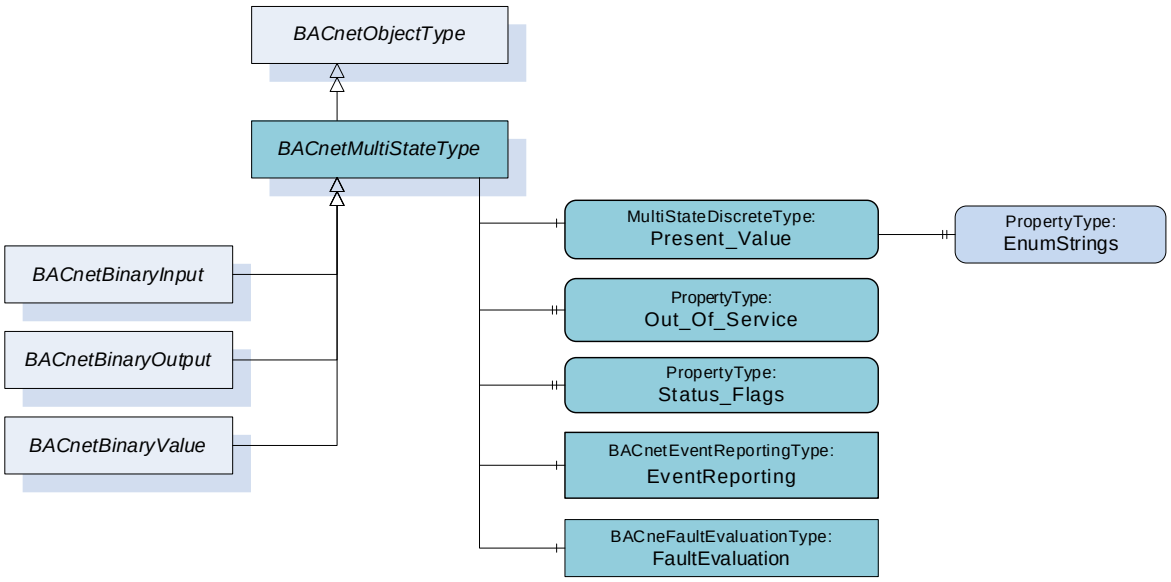


Figure 21 – *BACnetMultiStateType overview*

7.12.2 ObjectType definition

The *BACnetMultiStateType ObjectType* is formally defined in Table 24.

Table 24 – *BACnetMultiStateType* Definition

Attribute	Value				
BrowseName	BACnetMultiStateType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasComponent	Variable	Present_Value	0:UInteger	MultiStateDiscreteType	M
0:HasProperty	Variable	Out_Of_Service	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Status_Flags	BACnetStatusFlags	0:PropertyType	M
0:HasComponent	Object	EventReporting		BACnetEventReportingType	O
0:HasComponent	Object	FaultEvaluation		BACnetFaultEvaluationType	O

The *BACnetMultiStateType ObjectType* is an abstract type and cannot be used directly.

7.12.3 ObjectType Description

7.12.3.1 Variable Present_Value

This OPC UA Property, of *DataType 0:UInteger*, reflects the logical state of the input.

This OPC UA *MultiStateDiscreteType Variable* represents the BACnet properties Present_Value, State_Text and Number_Of_States. The OPC UA VariableType *MultiStateDiscreteType* is defined in OPC 10000-8. It defines the OPC UA *Property EnumStrings*.

The following list provides the BACnet property and the mapping to the corresponding data member in the OPC UA multistate discrete item.

- Present_Value represented by *Value Attribute* of the *Variable*.
- State_Text and Number_Of_States represented by the EnumStrings Property of the Variable. If the optional BACnet property State_Text is not present, the EnumStrings

property is filled with State_1 to State_N strings where the N and the number of strings are defined by Number_Of_States.

7.12.3.2 Variable Out_Of_Service

This OPC UA Property, of *DataType 0:Boolean*, represents the BACnet property Out_Of_Service.

It is an indication of whether or not the physical input that the object represents is not in service. While the Out_Of_Service property is True, the Present_Value and Reliability properties may be changed to any value as a means of simulation and/or testing.

7.12.3.3 Variable Status_Flags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

It represents four *0:Boolean* flags that represent the general health of a *BACnetMultiStateType*. The flags are IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE.

7.12.3.4 Object EventReporting

The *EventReporting Object* contains status and configuration information for the event reporting of *BACnetMultiStateType Objects*. The *BACnetEventReportingType* is defined in 0. The *Object* is optional and is not present if event generation is not activated for the BACnet object.

7.12.3.5 Object FaultEvaluation

The *FaultEvaluation Object* contains status and configuration information for the fault evaluation of the *BACnetMultiStateType*. The *BACnetFaultEvaluationType* is defined in 8.23. The *Object* is optional and is not present if fault evaluation is not activated for the BACnet object.

7.13 BACnetMultiStateInputType

7.13.1 General

This OPC UA *ObjectType* represents the *BACnet* object type *MultiState Input*. A multi-state input includes a Present_Value variable whose value represents a state set by a locally defined algorithm.

Figure 22 shows an overview for the *BACnetMultiStateInputType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 25.

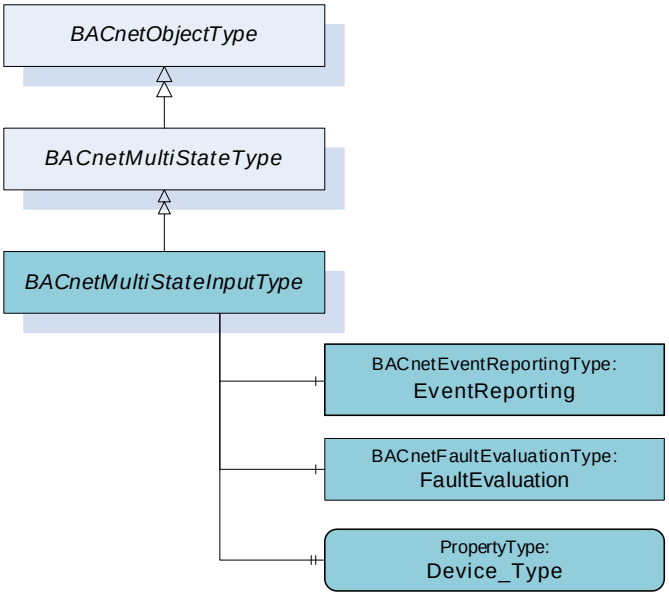


Figure 22 – *BACnetMultiStateInputType overview*

7.13.2 ObjectType definition

The *BACnetMultiStateInputType ObjectType* is formally defined in Table 25.

Table 25 – *BACnetMultiStateInputType* Definition

Attribute	**Value**				
BrowseName	BACnetMultiStateInputType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetMultiStateType* defined in 7.12.					
0:HasComponent	Object	EventReporting		BACnetEventReportingType	O
0:HasComponent	Object	FaultEvaluation		BACnetFaultEvaluationType	O
0:HasProperty	Variable	Device_Type	0:String	0:PropertyType	O

The *BACnetMultiStateInputType ObjectType* is a concrete type and can be used directly.

The components of the *BACnetMultiStateInputType* have additional subcomponents which are defined in Table 26.

Table 26 – BACnetMultiStateInputType Additional Subcomponents

BrowsePath	**References**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetChangeOfStateAlgorithmType	M
EventReporting EventAlgorithm	0:HasProperty	Variable	AlarmValues	0:UInteger [ ]	0:PropertyType	M
FaultEvaluation	0:HasComponent	Object	FaultAlgorithm		BACnetFaultStateAlgorithmType	O
FaultEvaluation FaultAlgorithm	0:HasProperty	Variable	FaultValues	0:UInteger [ ]	0:PropertyType	M

7.13.3 ObjectType Description

7.13.3.1 EventReporting Object override

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetChangeOfStateAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetMultiStateInputType*. The *BACnetChangeOfStateAlgorithmType* is defined in 8.10.

On the *EventAlgorithm* instance of the *BACnetChangeOfStateAlgorithmType*, the *DataType* of the *Alarm_Values Property* is changed to *0:UInteger* array.

The BACnet property Alarm_Values of the BACnet object type Multi State Input is mapped to the *AlarmValues Property* of the *BACnetChangeOfStateAlgorithmType*.

7.13.3.2 FaultEvaluation Object override

The instance declaration *Object FaultEvaluation* overrides definitions of the *BACnetFaultEvaluationType*.

The *FaultAlgorithm* component is changed to *TypeDefinition BACnetFaultStateAlgorithmType* if used in the *BACnetMultiStateInputType*. The *BACnetFaultStateAlgorithmType* is defined in 8.25.

On the FaultAlgorithm instance of the *BACnetFaultStateAlgorithmType*, *DataType* of the *Fault_Values Property* is changed to 0:UInteger array.

7.13.3.3 Variable Device_Type

This OPC UA *Property*, of *DataType String*, represents the BACnet property Device_Type.

It provides a text description of the physical device connected to the multi-state input.

7.14 BACnetMultiStateOutputType

7.14.1 General

This OPC UA *ObjectType* represents the *BACnet* object type *MultiState Output*. A multi-state output includes a Present_Value variable whose value represents the desired state of a physical output. The states and algorithms used to generate them are locally defined.

Figure 23 shows an overview for the *BACnetMultiStateOutputType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 27.

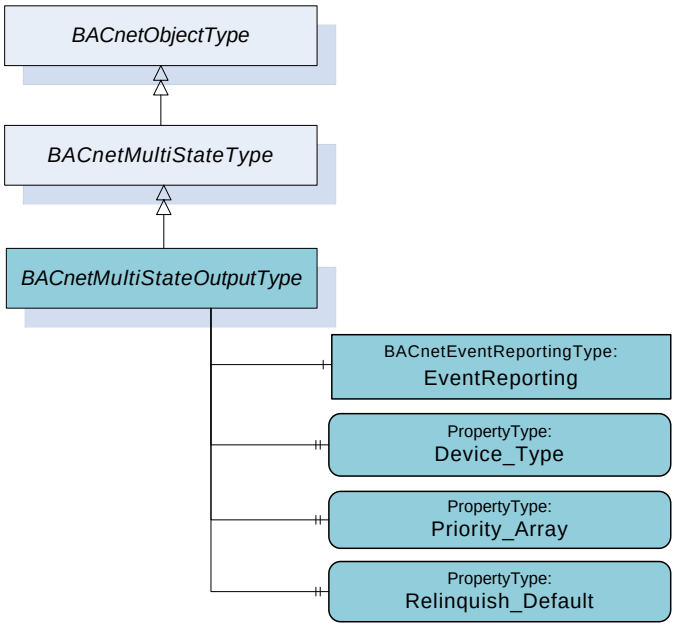


Figure 23 – *BACnetMultiStateOutputType overview*

7.14.2 ObjectType definition

The *BACnetMultiStateOutputType ObjectType* is formally defined in Table 27.

Table 27 – *BACnetMultiStateOutputType* Definition

Attribute	**Value**				
BrowseName	BACnetMultiStateOutputType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetMultiStateType* defined in 7.12.					
0:HasComponent	Object	EventReporting		BACnetEvent ReportingType	O
0:HasProperty	Variable	Device_Type	String	0:PropertyType	O
0:HasProperty	Variable	Priority_Array	BACnetPriorityValue [16]	0:PropertyType	M
0:HasProperty	Variable	Relinquish_Default	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Feedback_Value	0:UInteger	0:PropertyType	O

The *BACnetMultiStateOutputType ObjectType* is a concrete type and can be used directly.

The components of the *BACnetMultiStateOutputType* have additional subcomponents which are defined in Table 28.

Table 28 – BACnetMultiStateOutputType Additional Subcomponents

BrowsePath	**References**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetCommandFailureAlgorithmType	M

7.14.3 ObjectType Description

7.14.3.1 EventReporting Object override

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetCommandFailureAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetMultiStateOutputType*. The *BACnetCommandFailureAlgorithmType* is defined in 8.10.

7.14.3.2 Variable Device_Type

This OPC UA *Property*, of *DataType String*, represents the BACnet property Device_Type.

It is a text description of the physical device connected to the multi-state output.

7.14.3.3 Variable Priority_Array

This OPC UA Property, of *DataType* BACnetPriorityValue [16], represents the BACnet property Priority_Array. The *BACnetPriorityValue DataType* is defined in 10.6.4.

It is an array that contains prioritized commands that are in effect for this object. See 3.2.1 for details on command prioritization.

7.14.3.4 Variable Relinquish_Default

This OPC UA *Property*, of *DataType 0:UInteger*, represents the BACnet property Relinquish_Default.

It is the default value to be used for the Present_Value when all command priority values in the Priority_Array have a NULL value. See 3.2.1 for details on command prioritization.

7.14.3.5 Variable Feedback_Value

This OPC UA *Property*, of type *0:UInteger*, is an indication of the actual value of the entity controlled by Present_Value.

7.15 BACnetMultiStateValueType

7.15.1 General

This OPC UA *ObjectType* represents a *BACnet Multi-state Value* object type. A multi-state value is a control system parameter that resides in the memory of the BACnet Device and includes a Present_Value variable whose value represents the object state. The states and algorithms used to generate them are locally defined.

Figure 24 shows an overview for the *BACnetMultiStateValueType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 29.

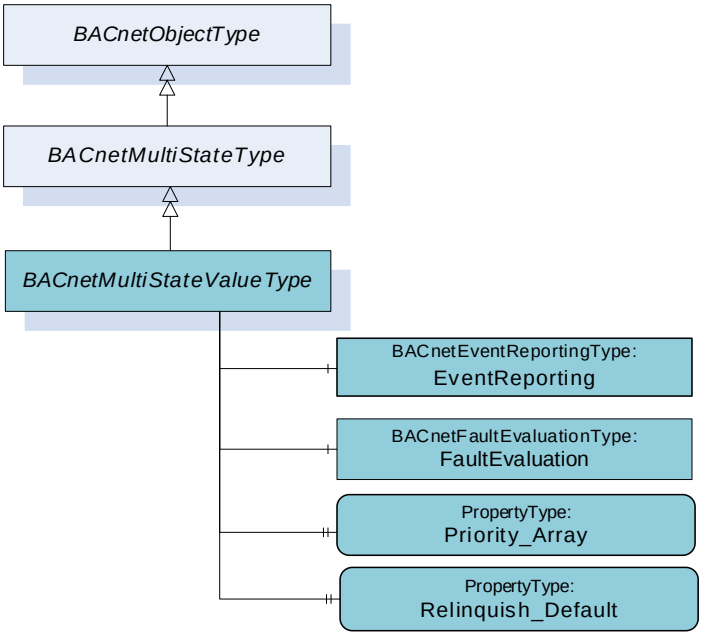


Figure 24 – *BACnetMultiStateValueType overview*

7.15.2 ObjectType definition

The *BACnetMultiStateValueType ObjectType* is formally defined in Table 29.

Table 29 – *BACnetMultiStateValueType* Definition

Attribute	**Value**				
BrowseName	BACnetMultiStateValueType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetMultiStateType* defined in 7.12.					
0:HasComponent	Object	EventReporting		BACnetEvent ReportingType	O
0:HasComponent	Object	FaultEvaluation		BACnetFaultEvaluationT ype	O
0:HasProperty	Variable	Priority_Array	BACnetPriorityValue [16]	0:PropertyType	O
0:HasProperty	Variable	Relinquish_Default	0:UInteger	0:PropertyType	O

The *BACnetMultiStateValueType ObjectType* is a concrete type and can be used directly.

The components of the *BACnetMultiStateValueType* have additional subcomponents which are defined in Table 30.

Table 30 – BACnetMultiStateValueType Additional Subcomponents

BrowsePath	**References**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetChangeOfSta teAlgorithmType	M
EventReporting EventAlgorithm	0:HasProperty	Variable	AlarmValues	0:UInteger [ ]	0:PropertyType	M
FaultEvaluation	0:HasComponent	Object	FaultAlgorithm		BACnetFaultStateAlg orithmType	O
FaultEvaluation FaultAlgorithm	0:HasProperty	Variable	FaultValues	0:UInteger [ ]	0:PropertyType	M

7.15.3 ObjectType Description

7.15.3.1 EventReporting Object override

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetChangeOfStateAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetMultiStateValueType*. The *BACnetChangeOfStateAlgorithmType* is defined in 8.10.

On the *EventAlgorithm* instance of the *BACnetChangeOfStateAlgorithmType*, the *DataType* of the *Alarm_Values Property* is changed to *0:UInteger* array.

The BACnet property Alarm_Values of the BACnet object type Multi State Value is mapped to the *AlarmValues Property* of the *BACnetChangeOfStateAlgorithmType*.

7.15.3.2 FaultEvaluation Object override

The instance declaration *Object FaultEvaluation* overrides definitions of the *BACnetFaultEvaluationType*.

The *FaultAlgorithm* component is changed to *TypeDefinition BACnetFaultStateAlgorithmType* if used in the *BACnetMultiStateValueType*. The *BACnetFaultStateAlgorithmType* is defined in 8.25.

On the FaultAlgorithm instance of the *BACnetFaultStateAlgorithmType*, *DataType* of the *Fault_Values Property* is changed to 0:UInteger array.

7.15.3.3 Variable Priority_Array

This OPC UA Property, of *DataType* BACnetPriorityValue [16], represents the BACnet property Priority_Array. The *BACnetPriorityValue DataType* is defined in 10.6.4.

It is an array that contains prioritized commands that are in effect for this object. See 3.2.1 for details on command prioritization.

7.15.3.4 Variable Relinquish_Default

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Relinquish_Default.

It is the default value to be used for the Present_Value when all command priority values in the Priority_Array have a NULL value. See 3.2.1 for details on command prioritization.

7.15.3.5 Variable Fault_Values

This OPC UA *Property*, of *DataType UInteger* [], represents the BACnet property Fault_Values.

The Fault_Values provide the values used by the fault state algorithm to determine whether the monitored value is in a fault state.

7.16 BACnetCalendarType

7.16.1 General

This OPC UA *ObjectType* represents a *BACnet Calendar* object type. A calendar is used to describe a list of calendar dates. Each entry in the list describes a specific date or date pattern, range of dates, or month/week-of-month/day-of-week specification.

Figure 25 shows an overview for the *BACnetCalendarType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 31.

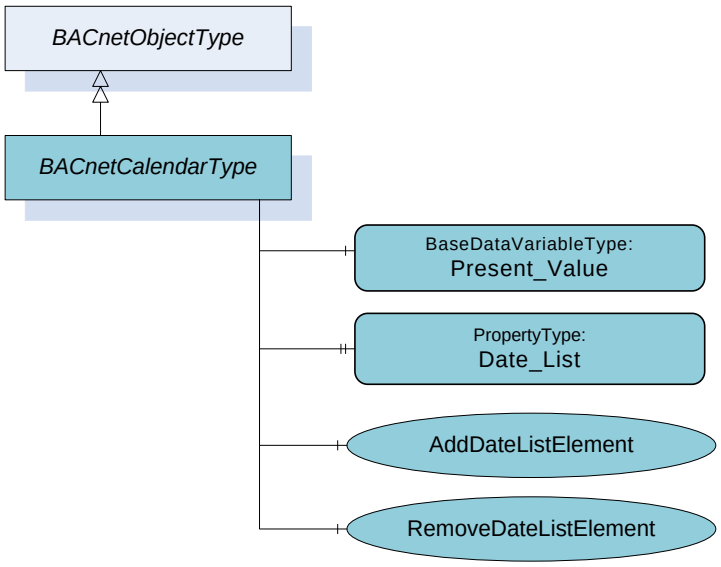


Figure 25 – *BACnetCalendarType overview*

7.16.2 ObjectType definition

The *BACnetCalendarType ObjectType* is formally defined in Table 31.

Table 31 – *BACnetCalendarType* Definition

Attribute	**Value**				
BrowseName	BACnetCalendarType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasComponent	Variable	Present_Value	0:Boolean	0:BaseDataVariableType	M
0:HasProperty	Variable	Date_List	BACnetCalendarEntry [ ]	0:PropertyType	M
0:HasComponent	Method	AddDateListElements			M
0:HasComponent	Method	RemoveDateListElements			M

The *BACnetCalendarType ObjectType* is a concrete type and can be used directly.

7.16.3 ObjectType Description

7.16.3.1 Variable Present_Value

This OPC UA *Variable*, of *DataType* Boolean, represents the BACnet property Present_Value.

It indicates the current value of the calendar: True if the current date is in the Date_List and False if it is not.

7.16.3.2 Variable Date_List

This OPC UA *Property*, of *DataType BACnetCalendarEntry* [], represents the BACnet property Date_List. The *BACnetCalendarEntry DataType* is defined in 10.6.2.

It is an array of elements of which each is either a specific date or date pattern (BACnetDate see 10.5.6), range of dates (BACnetDateRange see 10.5.7), or month/week-of-month/day-of-week specification (BACnetWeekNDay see 10.5.35).

7.16.3.3 Method AddDateListElements

This Method adds entries to the BACnet property Date_List.

Signature

```
AddDateListElements (
   [in]  BACnetCalendarEntry[]          CalendarEntries
   [out] 0:UInt32                       FirstFailedElementNumber
   );
```

Argument	Description
CalendarEntries	Array of calendar entries to add to the entries in the BACnet property Date_List. The BACnetCalenderEntry DataType is defined in 10.6.2.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the CalendarEntries. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE and DATATYPE_NOT_SUPPORTED
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadOutOfMemory	This status is returned for the BACnet error code NO_SPACE_TO_ADD_LIST_ELEMENT

7.16.3.4 Method RemoveDateListElements

This Method removes entries from the BACnet property Date_List.

Signature

```
RemoveDateListElements (
   [in]  BACnetCalendarEntry []       CalendarEntries
   [out] 0:UInt32                     FirstFailedElementNumber
   );
```

Argument	Description
CalendarEntries	Array of calendar entries to remove from the entries in the BACnet property Date_List. The BACnetCalenderEntry DataType is defined in 10.6.2.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the CalendarEntries. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadNotFound	This status is returned for the BACnet error code LIST_ELEMENT_NOT_FOUND

7.17 BACnetScheduleType

7.17.1 General

This OPC UA *ObjectType* represents a *BACnet Schedule* object type. This type defines a periodic schedule that can recur over a range of dates. The schedule may have optional exceptions at arbitrary times or dates. The basic unit of a schedule is days, which are divided into two types: normal days within a week and exception days. A priority mechanism defines which scheduled event is currently valid. The schedule includes a Present_Value variable whose value describes the current state of the schedule, including a default value when no schedules are in effect.

Figure 26 shows an overview for the *BACnetScheduleType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 32.

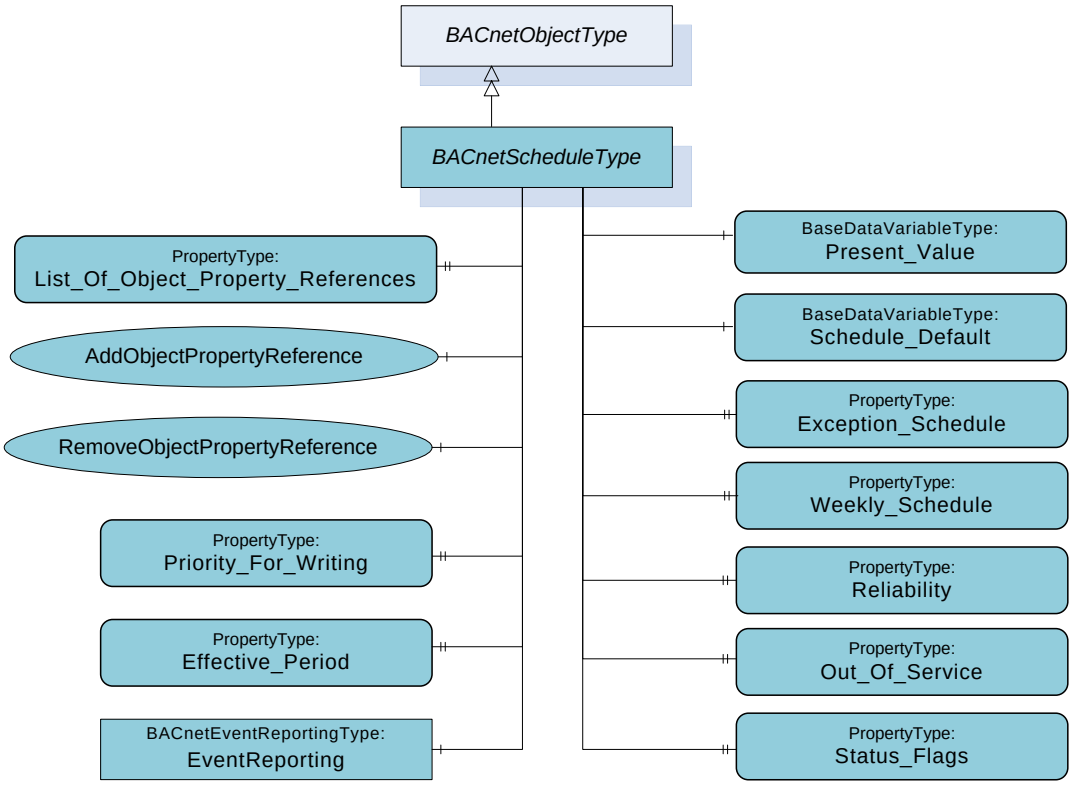


Figure 26 – *BACnetScheduleType overview*

7.17.2 ObjectType definition

The *BACnetScheduleType ObjectType* is formally defined in Table 32.

Table 32 – *BACnetScheduleType* Definition

Attribute	Value				
BrowseName	BACnetScheduleType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasComponent	Variable	Present_Value	0:BaseDataType	0:BaseDataVariableType	M
0:HasComponent	Variable	Schedule_Default	0:BaseDataType	0:BaseDataVariableType	M
0:HasProperty	Variable	Exception_Schedule	BACnetSpecialEvent []	0:PropertyType	O
0:HasProperty	Variable	Weekly_Schedule	BACnetDailySchedule[7]	0:PropertyType	O
0:HasProperty	Variable	List_Of_Object_Property_References	BACnetDeviceObject PropertyReference[]	0:PropertyType	M
0:HasComponent	Method	AddObjectProperty References			M
0:HasComponent	Method	RemoveObject PropertyReferences			M
0:HasProperty	Variable	Priority_For_Writing	0:Byte	0:PropertyType	M
0:HasComponent	Object	FaultEvaluation		BACnetFaultEvaluation Type	O
0:HasProperty	Variable	Out_Of_Service	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Status_Flags	BACnetStatusFlags	0:PropertyType	M
0:HasProperty	Variable	Effective_Period	BACnetDateRange	0:PropertyType	M
0:HasComponent	Object	EventReporting		BACnetEvent ReportingType	O

The *BACnetScheduleType ObjectType* is a concrete type and can be used directly.

7.17.3 ObjectType Description

7.17.3.1 Variable Present_Value

This OPC UA *Variable*, of *DataType* BaseDataType, represents the BACnet property Present_Value.

It indicates the current value of the schedule. Most analog, binary, and enumerated values may be scheduled using this abstract *DataType*.

7.17.3.2 Variable Schedule_Default

This OPC UA *Variable*, of *DataType* BaseDataType, represents the BACnet property Schedule_Default.

It represents a default value to be used for the Present_Value property when no other scheduled value is in effect.

7.17.3.3 Variable Exception_Schedule

This OPC UA *Property*, of *DataType BACnetSpecialEvent* [], represents the BACnet property Exception_Schedule. The *BACnetSpecialEvent DataType* is defined in 10.5.31.

Each BACnetSpecialEvent describes a sequence of schedule actions that take precedence over a normal day's behaviour on a special day or days.

At least one of the two BACnet properties Exception_Schedule and Weekly_Schedule must be present.

7.17.3.4 Variable Weekly_Schedule

This OPC UA *Property*, of *DataType BACnetDailySchedule* [7], represents the BACnet property Weekly_Schedule. The *BACnetDailySchedule DataType* is defined in 10.5.5.

The Weekly_Schedule property contains exactly 7 elements, one for each day of the week. Each element describes a sequence of time/value pairs that provides a sequence of schedule actions on one day of the week when no Exception_Schedule is in effect.

At least one of the two BACnet properties Exception_Schedule and Weekly_Schedule must be present.

7.17.3.5 Variable List_Of_Object_Property_References

This OPC UA *Property*, of *DataType BACnetDeviceObjectPropertyReference*, represents the BACnet property List_Of_Object_Property_References.

The *BACnetDeviceObjectPropertyReference DataType* is defined in 10.5.9.

It specifies the Device_Identifiers, Object_Identifiers, and Property_Identifiers of the properties to be written with specific values at specific times on specific days.

7.17.3.6 Method AddObjectPropertyReferences

This Method adds entries to the BACnet property List_Of_Object_Property_References.

Signature

```
AddObjectPropertyReferences (
   [in]  BACnetDeviceObjectPropertyReference []
                    DeviceObjectPropertyReferences
   [out] 0:UInt32   FirstFailedElementNumber
   );
```

Argument	Description
DeviceObjectProperty References	Array of device object references to add to the entries in the BACnet property List_Of_Object_Property_References. The BACnetDeviceObjectPropertyReference DataType is defined in 10.5.9.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the CalendarEntries. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE and DATATYPE_NOT_SUPPORTED
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadOutOfMemory	This status is returned for the BACnet error code NO_SPACE_TO_ADD_LIST_ELEMENT

7.17.3.7 Method RemoveObjectPropertyReferences

This Method removes entries from the BACnet property List_Of_Object_Property_References.

Signature

```
RemoveObjectPropertyReferences (
   [in]  BACnetDeviceObjectPropertyReference []
                    DeviceObjectPropertyReferences
```

```
    [out] 0:UInt32    FirstFailedElementNumber
    );
```

Argument	Description
DeviceObjectProperty References	Array of device object references to add to the entries in the BACnet property List_Of_Object_Property_References. The BACnetDeviceObjectPropertyReference DataType is defined in 10.5.9.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the CalendarEntries. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadNotFound	This status is returned for the BACnet error code LIST_ELEMENT_NOT_FOUND

7.17.3.8 Variable Priority_For_Writing

This OPC UA Property, of *DataType* Byte, represents the BACnet property Priority_For_Writing.

It defines the priority at which the referenced properties are commanded. Valid values are in the range 1-16, with 1 being considered the highest priority and 16 the lowest. See 3.2.1 for details on command prioritization.

7.17.3.9 Object FaultEvaluation

The *FaultEvaluation Object* contains status and configuration information for the fault evaluation of the *BACnetScheduleType*. The *BACnetFaultEvaluationType* is defined in 8.23. The *Object* is optional and is not present if fault evaluation is not activated for the BACnet object.

7.17.3.10 Variable Out_Of_Service

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Out_Of_Service.

It is an indication of whether or not the internal calculations of the schedule object are used to determine the value of the Present_Value property. Other functions that depend on the state of the Present_Value shall respond to changes made to that property while Out_Of_Service is True, as if those changes had occurred by internal calculations.

7.17.3.11 Variable Status_Flags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

It represents four boolean flags that represent the general health of an analog input. The flags are IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE.

7.17.3.12 Variable Effective_Period

This OPC UA *Property*, of *DataType BACnetDateRange*, represents the BACnet property Status_Flags. The *BACnetDateRange DataType* is defined in 10.5.7.

It specifies the range of dates within which the Schedule object is active. Upon entering its effective period, the object shall calculate its Present_Value.

7.17.3.13 Object EventReporting

BACnet schedule objects may optionally support event reporting to facilitate the reporting of fault conditions. The *BACnetEventReportingType* is defined in 8.8.

7.18 BACnetLoopType

7.18.1 General

This OPC UA *ObjectType* represents a *BACnet Loop* object type. This type defines the externally visible characteristics of a feedback control loop.

Figure 27 shows an overview for the *BACnetLoopType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 33.

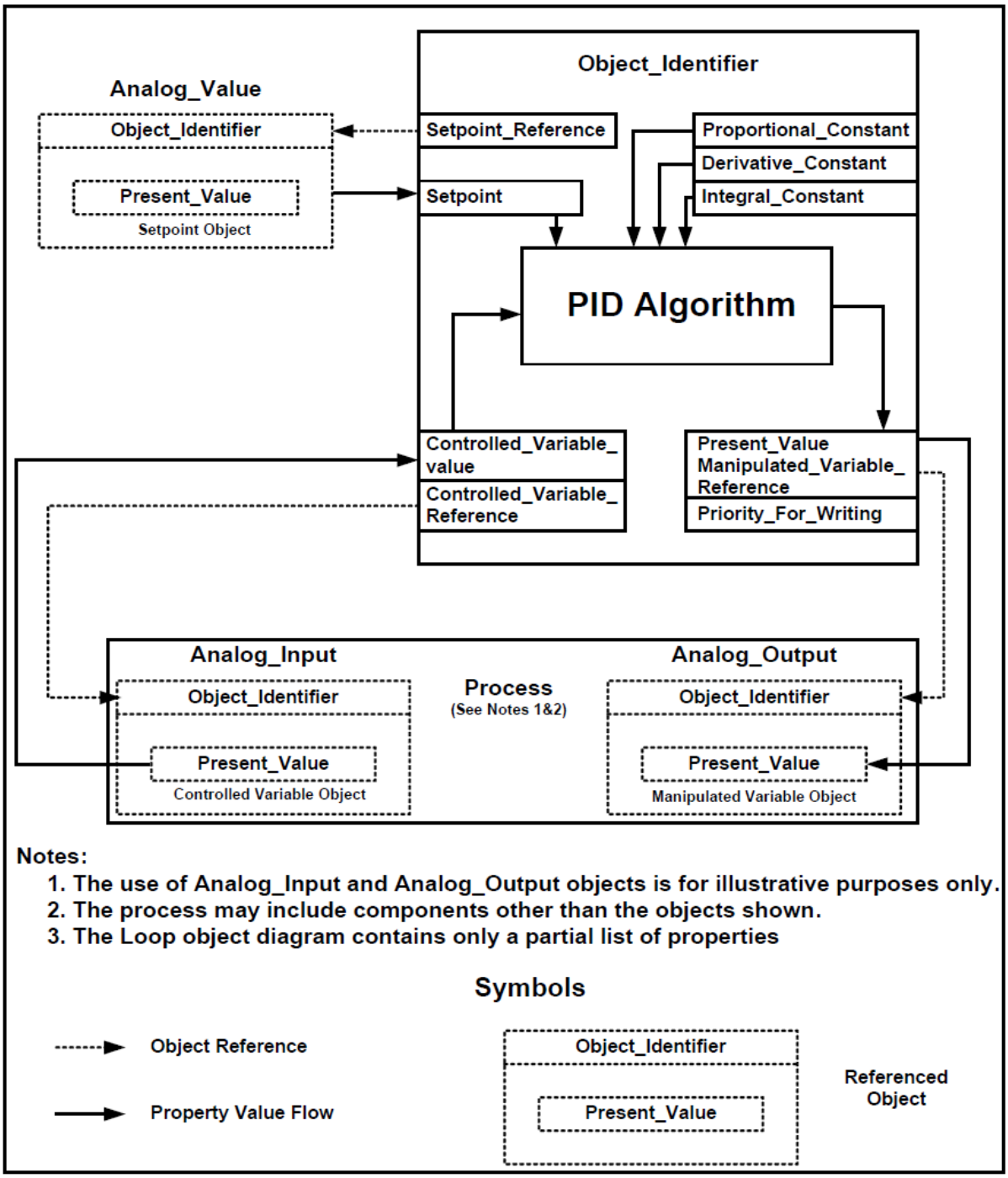


Figure 27 – *BACnetLoopType overview (example PID loop)*

7.18.2 ObjectType definition

The *BACnetLoopType ObjectType* is formally defined in Table 33.

Table 33 – *BACnetLoopType* Definition

Attribute	Value				
BrowseName	BACnetLoopType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasComponent	Variable	Present_Value	0:Float	0:AnalogUnitType	M
0:HasProperty	Variable	Status_Flags	BACnetStatusFlags	0:PropertyType	M
0:HasComponent	Object	FaultEvaluation		BACnetFaultEvaluation Type	O
0:HasProperty	Variable	Out_Of_Service	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Manipulated_Variable _Reference	BACnetDeviceObject PropertyReference	0:PropertyType	M
0:HasProperty	Variable	Controlled_Variable_R eference	BACnetDeviceObject PropertyReference	0:PropertyType	M
0:HasComponent	Variable	Controlled_Variable_V alue	0:Float	0:AnalogUnitType	M
0:HasProperty	Variable	Setpoint_Reference	BACnetDeviceObject PropertyReference	0:PropertyType	M
0:HasComponent	Variable	Setpoint	0:Float	0:AnalogUnitType	M
0:HasProperty	Variable	Action	BACnetAction	0:PropertyType	M
0:HasComponent	Variable	Proportional_Constant	0:Float	0:AnalogUnitType	O
0:HasComponent	Variable	Integral_Constant	0:Float	0:AnalogUnitType	O
0:HasComponent	Variable	Derivative_Constant	0:Float	0:AnalogUnitType	O
0:HasComponent	Variable	Bias	0:Float	0:AnalogUnitType	O
0:HasProperty	Variable	Priority_For_Writing	0:Byte	0:PropertyType	O
0:HasProperty	Variable	COV_Increment	0:Float	0:PropertyType	O
0:HasComponent	Object	EventReporting		BACnetEventReportingT ype	O

The *BACnetLoopType ObjectType* is a concrete type and can be used directly.

The components of the *BACnetLoopType* have additional subcomponents which are defined in Table 34.

Table 34 – BACnetLoopType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetFloatingLimit AlgorithmType	M

7.18.3 ObjectType Description

7.18.3.1 Variable Present_Value

This OPC UA *Variable*, of *DataType* Float, represents the BACnet property Present_Value. The OPC UA *EngineeringUnits Property* of the *Variable* represents the BACnet property Output_Units.

The Present_Value of the BACnetLoopType indicates the current output value of the loop algorithm and describes the engineering units of the value. If the object supports event reporting, then the Present_Value is the monitored value for the object's event algorithm. See the BACnet specification for information on the BACnet event enrolment model.

This OPC UA *AnalogItemType Variable* represents the BACnet properties Present_Value, Minimum_Output, Maximum_Output, Output_Units and Update_Interval. The OPC UA *VariableType AnalogItemType* is defined in OPC 10000-8. It defines the OPC UA *Properties EURange*, *EngineeringUnits* and *InstrumentRange*.

The following list provides the BACnet property and the mapping to the corresponding data member in the OPC UA AnalogItem.

- Present_Value represented by *Value Attribute* of the *Variable*.
- Minimum_Output represented by *Low* part of *EURange Property* of the *Variable*. If the optional BACnet property Minimum_Output is not present, the *EURange Property* should not be provided.
- Maximum_Output represented by the *High* part of *EURange Property* of the *Variable*. If the optional BACnet property Maximum_Output is not present, the *EURange Property* should not be provided.
- Output_Units represented by the *EngineeringUnits Property* of the *Variable*. The mapping of BACnet units to OPC UA units is defined in 11.
- Update_Interval represented by *MinSamplingInterval Attribute* of the *Variable*.

7.18.3.2 Variable Status_Flags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

It represents four *Boolean* flags that represent the general health of a *BACnetLoopType*. The flags are IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE.

7.18.3.3 Object FaultEvaluation

The *FaultEvaluation Object* contains status and configuration information for the fault evaluation of the *BACnetLoopType*. The *BACnetFaultEvaluationType* is defined in 8.23. The *Object* is optional and is not present if fault evaluation is not activated for the BACnet object.

7.18.3.4 Variable Out_Of_Service

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Out_Of_Service.

It is an indication of whether or not the physical input that the object represents is not in service. While *Out_Of_Service* is True, *Present_Value* and *Reliability* may be changed to any value as a means of simulation and/or testing.

7.18.3.5 Variable Manipulated_Variable_Reference

This OPC UA *Property*, of *DataType BACnetDeviceObjectPropertyReference*, represents the BACnet property Manipulated_Variable_Reference. The *BACnetDeviceObjectProperty Reference DataType* is defined in 10.5.9.

This property indicates an object and property that the output (Present_Value) of the control loop is written to. Generally this is a BACnet Analog Output object used to position a device, but it could also be another object or property. See the BACnet specification for additional examples.

7.18.3.6 Variable Controlled_Variable_Reference

This OPC UA *Property*, of *DataType BACnetDeviceObjectPropertyReference*, represents the BACnet property Controlled_Variable_Reference. The *BACnetDeviceObjectProperty Reference DataType* is defined in 10.5.9.

This property identifies the property used to set the Controlled_Variable_Value property of the Loop object. Normally this is the Present_Value of an Analog Input object that is used to measure a process variable (for example, temperature) but it could also be another object. See the BACnet specification for additional examples.

7.18.3.7 Variable Controlled_Variable_Value

This OPC UA *Variable*, of *DataType* Float, represents the BACnet property Controlled_Variable_Value. The OPC UA *EngineeringUnits Property* of the *Variable* represents the BACnet property Controlled_Variable_Units.

This property indicates the value of the property (and its engineering units) of the object that is referenced by the Controlled_Variable_Reference property. The control loop compares this value with the Setpoint to calculate the error.

7.18.3.8 Variable Setpoint_Reference

This OPC UA *Property*, of *DataType BACnetDeviceObjectPropertyReference*, represents the BACnet property Setpoint_Reference. The *BACnetDeviceObjectProperty Reference DataType* is defined in 10.5.9. The value 4194303 is used as *objectIdentifier* to indicate that the property is not initialized.

This property represents either zero or one reference. If the reference is zero, then the setpoint for this control loop is fixed and is contained in the Setpoint property. The presence of a reference indicates that the property of another object contains the setpoint value used for the Loop object and the reference specifies that property.

7.18.3.9 Variable Setpoint

This OPC UA *Variable*, of *DataType* Float, represents the BACnet property Setpoint. The OPC UA *EngineeringUnits Property* of the *Variable* represents the BACnet property Output_Units.

The Setpoint of the *BACnetLoopType* is the value of the loop setpoint or the property of the object referenced by the Setpoint_Reference (in engineering units described by the Controlled_Variable_Value property).

7.18.3.10 Variable Action

This OPC UA *Property*, of *DataType BACnetAction*, represents the BACnet property Action. The DataType *BACnetAction* is defined in 10.4.2.

This property indicates whether the loop is “direct” or “reverse”.

7.18.3.11 Variable Proportional_Constant

This OPC UA *Variable*, of *DataType* Float, represents the BACnet property Proportional_Constant. The OPC UA *EngineeringUnits Property* of the *Variable* represents the BACnet property Proportional_Constant_Units.

This property indicates the value and engineering unit of the proportional gain parameter that is used by the loop algorithm. It is used to represent the various forms of gain for the proportional control mode (overall gain, throttling range, or proportional band).

7.18.3.12 Variable Integral_Constant

This OPC UA *Variable*, of *DataType* 0:Float, represents the BACnet property Integral_Constant. The OPC UA *EngineeringUnits Property* of the *Variable* represents the BACnet property Integral_Constant_Units.

This property indicates the value and engineering units of the integral gain parameter that is used by the loop algorithm. It may be used to represent any of the various forms of gain for the integral control mode (reset time or rate).

7.18.3.13 Variable Derivative_Constant

This OPC UA *Variable*, of *DataType* 0:Float, represents the BACnet property Derivative_Constant. The OPC UA *EngineeringUnits Property* of the *Variable* represents the BACnet property Derivative_Constant_Units.

This property indicates the value and engineering units of the derivative gain parameter used by the loop algorithm. It may be used to represent any of the various forms of gain for the derivative control mode (derivative time or rate time).

7.18.3.14 Variable Bias

This OPC UA *Variable*, of *DataType* 0:Float, represents the BACnet property Bias. The OPC UA *EngineeringUnits Property* of the *Variable* represents the BACnet property Output_Units.

This property indicates the bias value that is used by the loop algorithm.

7.18.3.15 Variable Priority_For_Writing

This OPC UA *Variable*, of *DataType* Byte, represents the BACnet property Priority_For_Writing.

This property provides a priority to be used by the command prioritization mechanism. This allows loop objects to be used to control the commandable property of an object. In particular, it identifies the particular priority slot in the Priority_Array of the Manipulated_Variable_Reference that is controlled by this loop. See the BACnet specification for the specific range of allowed values.

7.18.3.16 Variable COV_Increment

This OPC UA *Variable*, of *DataType* 0:Float, represents the BACnet property COV_Increment.

This property specifies the minimum change in Present_Value that will cause a COV Notification to be issued to COV clients. This property is required if COV reporting is supported by this object.

7.18.3.17 Object EventReporting

The *EventReporting Object* contains status and configuration information for the event reporting of *BACnetLoopType*. The *BACnetEventReportingType* is defined in 8.8. The *Object* is optional and is not present if event generation is not activated for the BACnet object.

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetFloatingLimitAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetLoopType*. The *BACnetFloatingLimitAlgorithmType* is defined in 8.12.

On the EventAlgorithm instance of the *BACnetFloatingLimitAlgorithmType* the BACnet property Error_Limit is mapped to both LowDiffLimit and HighDiffLimit. The BACnet property Deadband is mapped to pDeadband. The parameter SetpointReference contains the reference to the BACnet property Setpoint of this object.

7.19 BACnetEventEnrollmentType

7.19.1 General

This OPC UA *ObjectType* represents a status, error, and event reporting configuration for a BACnet device.

Figure 28 shows an overview for the *BACnetEventEnrollmentType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 35.

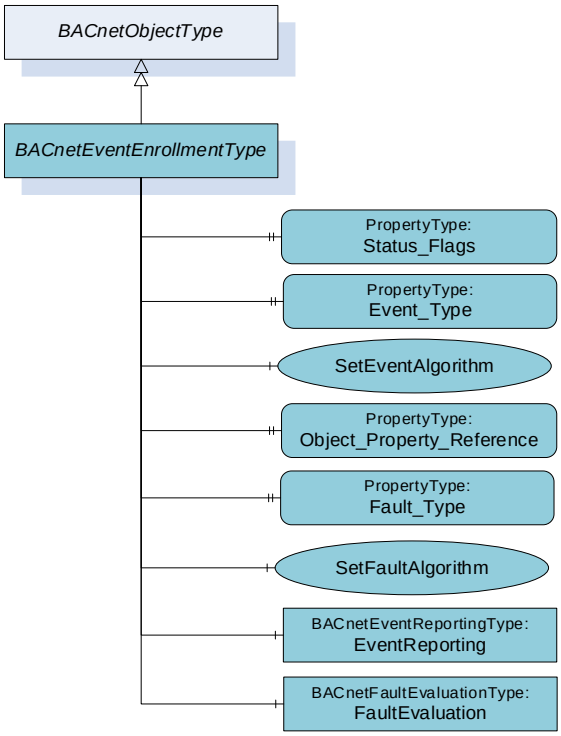


Figure 28 – BACnetEventEnrollmentType overview

7.19.2 ObjectType definition

The *BACnetEventEnrollmentType ObjectType* is formally defined in Table 35.

Table 35 – BACnetEventEnrollmentType Definition

Attribute	Value				
BrowseName	BACnetEventEnrollmentType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasProperty	Variable	Event_State	BACnetEventState	0:PropertyType	M
0:HasProperty	Variable	Status_Flags	BACnetStatusFlags	0:PropertyType	M
0:HasProperty	Variable	Event_Type	BACnetEventType	0:PropertyType	M
0:HasComponent	Method	SetEventAlgorithm			M
0:HasProperty	Variable	Object_Property_Reference	BACnetDeviceObjectPropertyReference	0:PropertyType	M
0:HasProperty	Variable	Fault_Type	BACnetFaultType	0:PropertyType	M
0:HasComponent	Method	SetFaultAlgorithm			O
0:HasComponent	Object	FaultEvaluation		BACnetFaultEvaluationType	M
0:HasComponent	Object	EventReporting		BACnetEventReportingType	M

The *BACnetEventEnrollmentType ObjectType* is a concrete type and can be used directly.

7.19.3 ObjectType Description

7.19.3.1 Variable Status_Flags

This OPC UA *Variable*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

It represents four 0:Boolean flags that represent the general health of an analog input. The flags are IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE.

7.19.3.2 Variable Event_Type

This OPC UA *Property*, of DataType *BACnetEventType*, represents the BACnet property Event_Type. The *BACnetEventType DataType* is defined in 10.4.12.

The value of this *Variable* is an enumeration that indicates the type of event algorithm that is to be used to detect the occurance of events and the event value notification parameters that are conveyed in event notifications.

7.19.3.3 Method SetEventAlgorithm

This Method sets the event algorithm for the *EventReporting Object*.

Signature

```
SetEventAlgorithm (
   [in]  BACnetEventParameter        EventParameters
   );
```

Argument	Description
EventParameters	The new event algorithm. The *BACnetEventParameter DataType* is defined in 10.6.4.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE and DATATYPE_NOT_SUPPORTED
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED

7.19.3.4 Variable Object_Property_Reference

This OPC UA *Property*, of *DataType BACnetDeviceObjectPropertyReference*, represents the BACnet property Object_Property_Reference. The *BACnetDeviceObjectProperty Reference DataType* is defined in 10.5.9.

It indicates the object and property that is referenced by the event enrollement object. The event algorithm specified by the Event_Type property is applied to the referenced property in order to determine the Event_State of the event.

7.19.3.5 Variable Fault_Type

This OPC UA *Property*, of DataType *BACnetFaultType*, represents the BACnet property Fault_Type. The *BACnetFaultType DataType* is defined in 10.4.13.

It is an enumeration (NONE, FAULT_CHARACTERSTRING, FAULT_EXTENDED, and so on) that indicates the type of fault algorithm that is applied by the event enrollment object.

7.19.3.6 Method SetFaultAlgorithm

This Method sets the fault algorithm for the *FaultEvaluation Object*.

Signature

```
SetFaultAlgorithm (
   [in]  BACnetFaultParameter        FaultParameters
   );
```

Argument	Description
FaultParameters	The new event algorithm. The *BACnetFaultParameter DataType* is defined in 10.6.4.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE and DATATYPE_NOT_SUPPORTED
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED

7.19.3.7 Object FaultEvaluation

The *FaultEvaluation Object* contains status and configuration information for the fault evaluation of the *BACnetEventEnrollmentType*. The *BACnetFaultEvaluationType* is defined in 8.23. The *Object* is optional and is not present if fault evaluation is not activated for the BACnet object.

7.19.3.8 Object EventReporting

The *EventReporting Object* contains status and configuration information for the event reporting of *BACnetEventEnrollmentType Objects*. The *BACnetEventReportingType* is defined in 0. The *Object* is optional and is not present if event generation is not activated for the BACnet object.

7.20 BACnetLogType

7.20.1 General

This OPC UA *ObjectType* represents an object which monitors a property of a referenced object and serializes the timestamp as local time and value of the property into an internal buffer when a pre-defined condition is met. The serialization may be invoked on a periodic basis, using a trigger or upon a change of value.

Figure 29 shows an overview for the *BACnetLogType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 36.

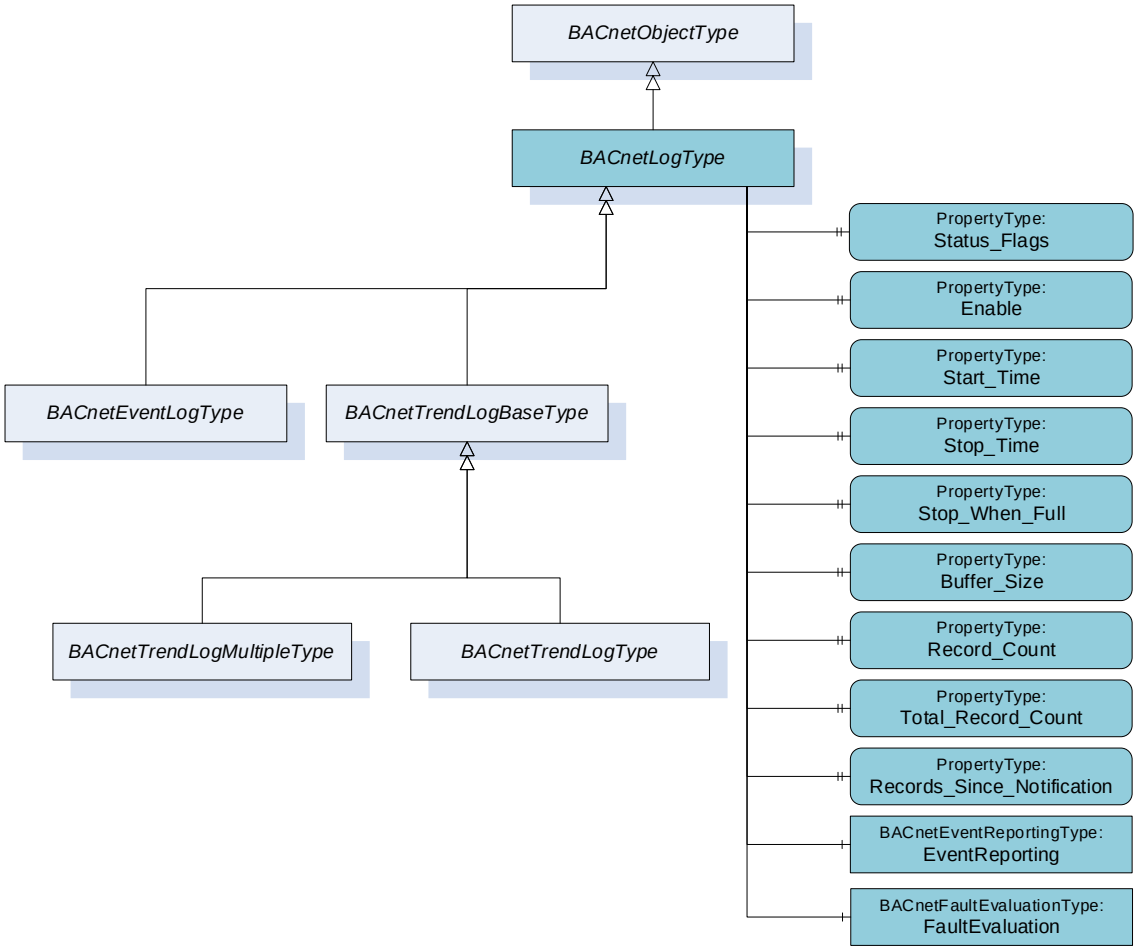


Figure 29 – BACnetLogType overview

7.20.2 ObjectType definition

The *BACnetLogType ObjectType* is formally defined in Table 36.

Table 36 – BACnetLogType Definition

Attribute	Value				
BrowseName	BACnetLogType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasProperty	Variable	Status_Flags	BACnetStatusFlags	0:PropertyType	M
0:HasProperty	Variable	Enable	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Start_Time	BACnetDateTime	0:PropertyType	O
0:HasProperty	Variable	Stop_Time	BACnetDateTime	0:PropertyType	O
0:HasProperty	Variable	Stop_When_Full	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Buffer_Size	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	Record_Count	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	Total_Record_Count	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	Records_Since_Notification	0:UInt32	0:PropertyType	O
0:HasComponent	Object	FaultEvaluation		BACnetFault Evaluation Type	M
0:HasComponent	Object	EventReporting		BACnetEvent ReportingType	M

The *BACnetLogType ObjectType* is an abstract type and cannot be used directly.

The components of the *BACnetLogType* have additional subcomponents which are defined in Table 37.

Table 37 – BACnetLogType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
EventReporting	0:HasComponent	Object	EventAlgorithm		BACnetBufferReadyAlgorithmType	M

7.20.3 ObjectType Description

7.20.3.1 Variable Status_Flags

This OPC UA *Variable*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

It represents four 0:Boolean flags that represent the general health of an analog input. The flags are IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE.

7.20.3.2 Variable Enable

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Enable.

When the Enable value is true, logging is enabled. When the Enable variable value is false, logging is not enabled. Logging occurs if and only if the Enable variable value is true and the device Local_Time.

7.20.3.3 Variable Start_Time

This OPC UA *Property*, of *DataType BACnetDateTime*, represents the BACnet property Start_Time. The *BACnetDateTime DataType* is defined in 10.5.8.

The Start_Time value indicates the date and time at or after which logging shall be enabled by this variable. The variable is O, and if unspecified is ignored for the purpose of determining whether or not logging is enabled. If Start_Time specifies a time and date that is equal to or later than the Stop_Time then logging shall be disabled.

7.20.3.4 Variable Stop_Time

This OPC UA *Property*, of *DataType BACnetDateTime*, represents the BACnet property Stop_Time. The *BACnetDateTime DataType* is defined in 10.5.8.

The Stop_Time value indicates the date and time at or after which logging shall be disabled by this variable. The variable is O, and if unspecified is ignored for the purpose of determining whether or not logging is disabled. If Stop_Time indicates a date and time that is earlier than the Start_Time, then logging is disabled.

7.20.3.5 Variable Stop_When_Full

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Stop_When_Full.

When the Stop_When_Full value is true, logging shall cease when the serialization buffer is full. When logging ceases because the next event would fill the buffer, an event indicating that the buffer is full is logged and the Enable variable shall be set false.

7.20.3.6 Variable Buffer_Size

This OPC UA *Property*, of *DataType UInt32*, represents the BACnet property Buffer_Size.

The Buffer_Size value indicates the number of records that may be contained by the buffer.

7.20.3.7 Variable Record_Count

This OPC UA *Property*, of *DataType UInt32*, represents the BACnet property Record_Count.

The Record_Count value indicates the number of records that are currently present in the buffer. When a value of zero is written to this variable, all records in the buffer shall be deleted and the Records_Since_Notification variable value shall be reset to zero.

7.20.3.8 Variable Total_Record_Count

This OPC UA *Property*, of *DataType UInt32*, represents the BACnet property Total_Record_Count.

The Total_Record_Count value indicates the total number of records that have been logged since the object was created. When the count is incremented past the maximum 32-bit integer value, the next value of the variable shall be 1.

7.20.3.9 Variable Records_Since_Notification

This OPC UA *Property*, of *DataType UInt32*, represents the BACnet property Records_Since_Notification.

The Records_Since_Notification indicates the number of records that have been logged since the buffer was last cleared (notification), or since the beginning of the logging process if no notification has been delivered.

7.20.3.10 Object FaultEvaluation

The *FaultEvaluation Object* contains status and configuration information for the fault evaluation of the *BACnetLoopType*. The *BACnetFaultEvaluationType* is defined in 8.23. The *Object* is optional and is not present if fault evaluation is not activated for the BACnet object.

7.20.3.11 Object EventReporting

The *EventReporting Object* contains status and configuration information for the event reporting of *BACnetTrendLogType*. The *BACnetEventReportingType* is defined in 8.8. The *Object* is optional and is not present if event generation is not activated for the BACnet object.

The instance declaration *Object EventReporting* overrides definitions of the *BACnetEventReportingType*.

The *EventAlgorithm* component is changed to *TypeDefinition BACnetFloatingLimitAlgorithmType* and *ModellingRule Mandatory* if used in the *BACnetLoopType*. The *BACnetBufferReadyAlgorithmType* is defined in 8.14.

The Properties TimeDelay and TimeDelayNormal are set to 0 and shall be read only since they do not exist on a BACnet Trend Log object.

On the EventAlgorithm instance of the *BACnetBufferReadyAlgorithmType* the BACnet property Notification_Threshold is mapped to Threshold. The BACnet property Last_Notify_Record is mapped to PreviousCount.

7.21 BACnetTrendLogBaseType

7.21.1 General

This OPC UA *ObjectType* represents the intrinsic configuration paremeters of a trend log. This primarily represents the basis upon which logging is performed: periodic, “triggered”, or based upon a change of value.

Figure 30 shows an overview for the *BACnetTrendLogBaseType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 38.

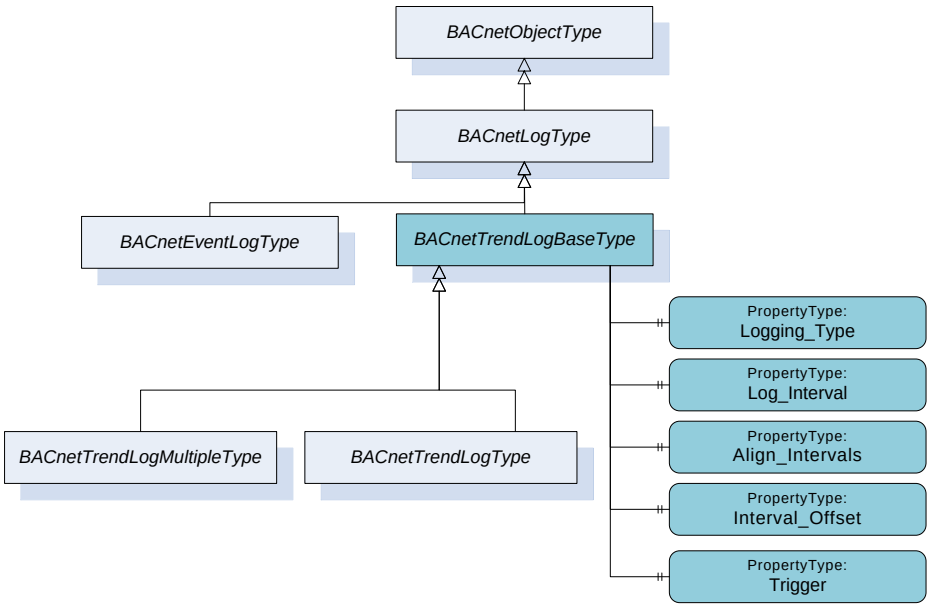


Figure 30 – BACnetTrendLogBaseType overview

7.21.2 ObjectType definition

The *BACnetTrendLogBaseType ObjectType* is formally defined in Table 38.

Table 38 – BACnetTrendLogBaseType Definition

Attribute		Value			
BrowseName		BACnetTrendLogBaseType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetLogType defined in 7.20.					
0:HasProperty	Variable	Logging_Type	BACnetLoggingType	0:PropertyType	O
0:HasProperty	Variable	Log_Interval	0:UInteger	0:PropertyType	O
0:HasProperty	Variable	Align_Intervals	0:Boolean	0:PropertyType	O
0:HasProperty	Variable	Interval_Offset	0:UInteger	0:PropertyType	O
0:HasProperty	Variable	Trigger	0:Boolean	0:PropertyType	O

The *BACnetTrendLogBaseType ObjectType* is an abstract type and cannot be used directly.

7.21.3 ObjectType Description

7.21.3.1 Variable Logging_Type

This OPC UA *Property*, of *DataType BACnetLoggingType*, represents the BACnet property Logging_Type. The *BACnetLoggingType DataType* is defined in 10.4.16.

The Logging_Type value specifies whether the trend log collects record periodically, by change of value, or via “triggered” acquisition.

7.21.3.2 Variable Log_Interval

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Log_Interval.

The Log_Interval value specifies a periodic value, in hundredths of a second, that the referenced property is logged when the Logging_Type variable value is POLLED. The variable value shall be zero and ignored when the Logging_Type variable value is anything other than POLLED.

7.21.3.3 Variable Align_Intervals

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Align_Intervals.

The Align_Intervals indicates whether clock-aligned periodic logging shall be enabled. When clock-aligned periodic logging is enabled, and the Log_Interval is a factor of a second, minute, hour, or day, then the beginning of the priod shall coincide with the next clock second, minute, hour, or day.

7.21.3.4 Variable Interval_Offset

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Interval_Offset.

The Interval_Offset value indicates an offset in hundredths of seconds from the beginning of the specified logging period until logging shall actually begin. The offset shall be interpreted as the Interval_Offset modulo the Log_Interval. For example, if the Log_Interval is 30 and the Interval_Offset is 31, then the offset shall be 1.

7.21.3.5 Variable Trigger

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Trigger.

When the Trigger variable value transitions from true to false and the Logging_Type is TRIGGERED then the trend log shall serialize the referenced property. The Trigger variable value is reset to false when data acquisition is completed.

7.22 BACnetTrendLogType

7.22.1 General

This OPC UA *ObjectType* represents an object that monitors a single property of a referenced object and logs its value and timestamp into an internal buffer when a set of pre-defined conditions are met.

Figure 31 shows an overview for the *BACnetTrendLogType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 39.

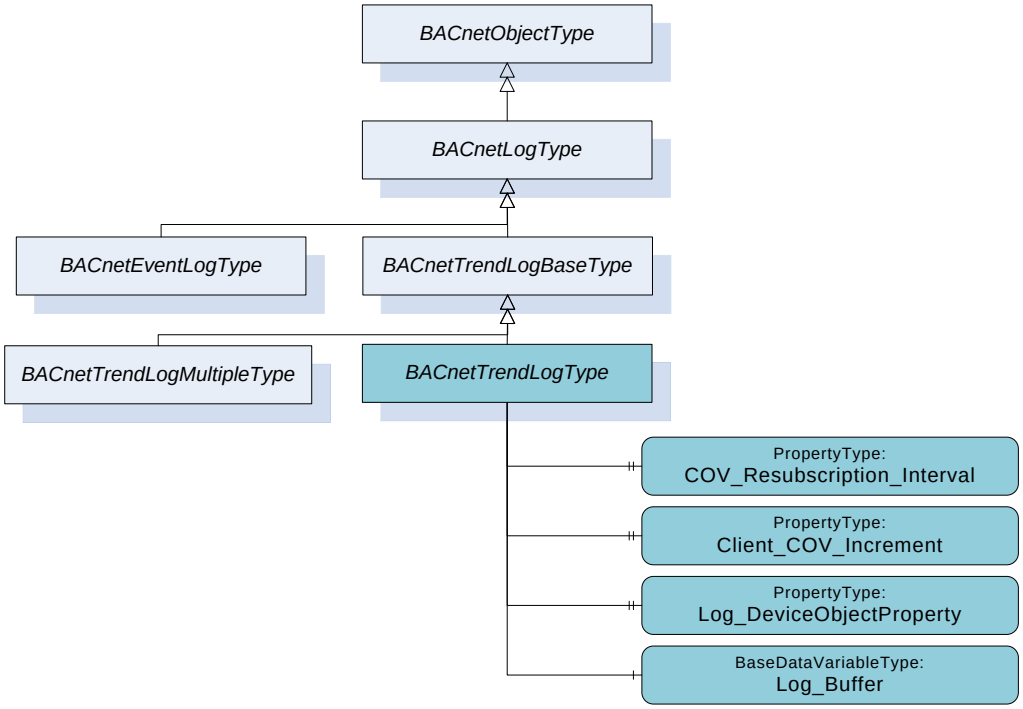


Figure 31 – BACnetTrendLogType overview

7.22.2 ObjectType definition

The *BACnetTrendLogType ObjectType* is formally defined in Table 39.

Table 39 – BACnetTrendLogType Definition

Attribute	Value				
BrowseName	BACnetTrendLogType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetTrendLogBaseType defined in 7.20.3.10.					
0:HasProperty	Variable	COV_Resubscription_Interval	0:UInteger	0:PropertyType	O
0:HasProperty	Variable	Client_COV_Increment	BACnetClientCOV	0:PropertyType	O
0:HasProperty	Variable	Log_DeviceObjectProperty	BACnetDeviceObject PropertyReference	0:PropertyType	O
0:HasComponent	Variable	Log_Buffer	BaseDataType	0:BaseData VariableType	M

The *BACnetTrendLogType ObjectType* is a concrete type and can be used directly.

7.22.3 ObjectType Description

7.22.3.1 Variable COV_Resubscription_Interval

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property COV_Resubscription_Interval.

The COV_Resubscription_Interval specifies the time in seconds between COV resubscriptions. The first COV subscription is issued when the trend log begins operations or when the Enable variable value becomes true.

7.22.3.2 Variable Client_COV_Increment

This OPC UA *Property*, of *DataType BACnetClientCOV*, represents the BACnet property Client_COV_Increment. The *BACnetClientCOV DataType* is defined in 10.6.3.

The Client_COV_Increment value indicates the increment that is used to determine whether a change of value has occurred.

7.22.3.3 Variable Log_DeviceObjectProperty

This OPC UA *Property*, of *DataType BACnetDeviceObjectPropertyReference* [], represents the BACnet property Log_DeviceObjectProperty. The *BACnetDeviceObjectPropertyReference DataType* is defined in 10.5.9.

The Log_DeviceObjectProperty variable specifies the Device Identifier, Object Identifier, and Property Identifier of the property to be trend logged.

7.22.3.4 Variable Log_Buffer

This OPC UA *Variable* represents the BACnet property Log_Buffer. It represents a list of logged values and can only be accessed with with the BACnet service ReadRange. Therefore, the OPC UA representation requires the *AccessLevel HistoryReadable* to allow access of the logged values through OPC UA *Service HistoryRead* (see Table 40).

The current value is not readable since it is not provided through BACnet directly. The *BACnetUaMapper* may provide the current value by reading the referenced BACnet property that is logged by the BACnet Trend Log object. In this case the AccessLevel can also have the Readable flag set.

Table 40 – Log_Buffer Attribute definition

OPC UA Attribute	Value
BrowseName	Log_Buffer
AccessLevel	HistoryReadable
Historizing	True if BACnet property Enable is True and the current time is between the BACnet property values Start_Time and Stop_Time

7.23 BACnetTrendLogMultipleType

7.23.1 General

This OPC UA *ObjectType* represents an object that monitors one or more properties of one or more referenced objects and logs the values and timestamps into an internal buffer when a set of pre-defined conditions are met.

Figure 32 shows an overview for the *BACnetTrendLogMultipleType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 41.

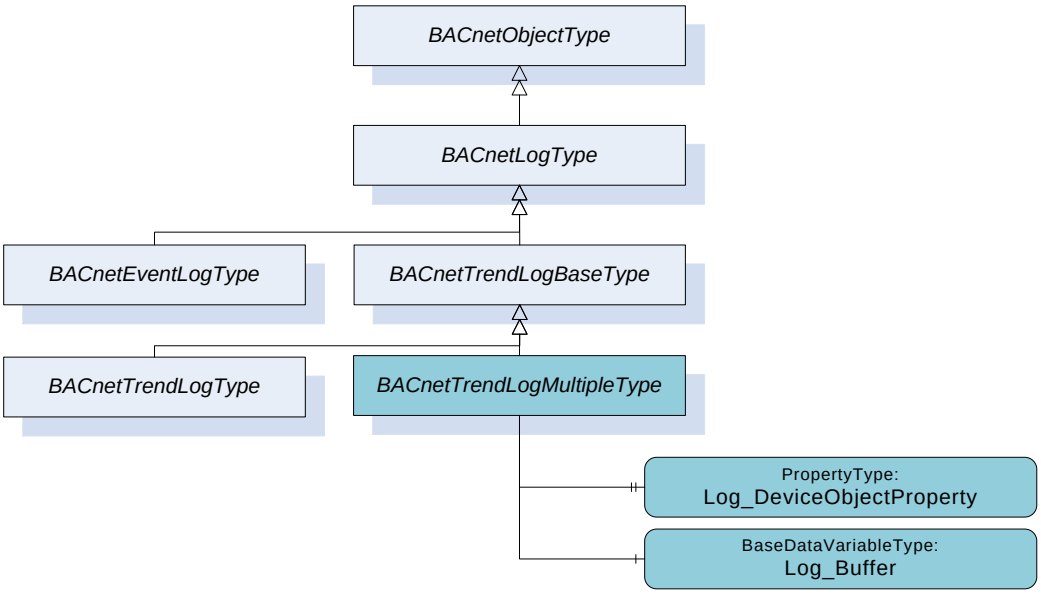


Figure 32 – BACnetTrendLogMultipleType overview

7.23.2 ObjectType definition

The *BACnetTrendLogMultipleType ObjectType* is formally defined in Table 41.

Table 41 – BACnetTrendLogMultipleType Definition

Attribute	Value				
BrowseName	BACnetTrendLogMultipleType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetTrendLogBaseType defined in 7.20.3.10.					
0:HasProperty	Variable	Log_DeviceObjectProperty	BACnetDeviceObjectPropertyReference [ ]	0:PropertyType	O
0:HasComponent	Variable	Log_Buffer	BaseDataType[]	0:BaseDataVariableType	M

The *BACnetTrendLogMultipleType ObjectType* is a concrete type and can be used directly.

7.23.3 ObjectType Description

7.23.3.1 Variable Log_DeviceObjectProperty

This OPC UA *Property*, of *DataType BACnetDeviceObjectPropertyReference* [], represents the BACnet property Log_DeviceObjectProperty. The *BACnetDeviceObjectPropertyReference DataType* is defined in 10.5.9.

The Log_DeviceObjectProperty array houses the set of properties that will be monitored and subsequently logged. When an element of the array has an object or device instance number equal to 4194303, this indicates that the element is uninitialized. For uninitialized elements, an indication that no property was specified shall be written to the corresponding entry in each log record.

7.23.3.2 Variable Log_Buffer

This OPC UA *Property*, of *DataType BACnetLogMultipleRecord*, represents the BACnet property Log_Buffer.

7.24 BACnetEventLogType

7.24.1 General

This OPC UA *ObjectType* represents an object that records event notifications with timestamps and other pertinent data into an internal buffer for subsequent retrieval.

Figure 33 shows an overview for the *BACnetEventLogType* with its *Properties* and related *ObjectTypes*. It is formally defined in Table 42.

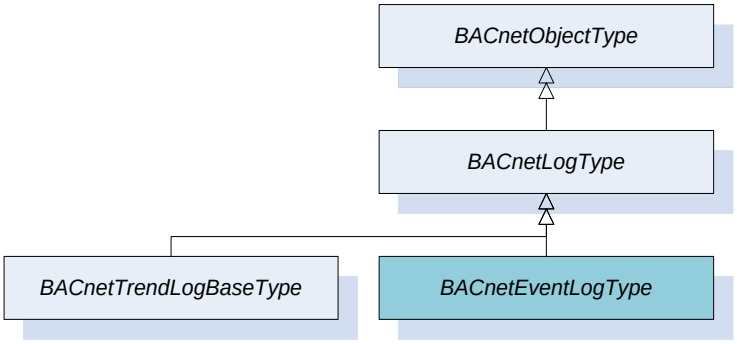


Figure 33 – BACnetEventLogType overview

7.24.2 ObjectType definition

The *BACnetEventLogType ObjectType* is formally defined in Table 42.

Table 42 – BACnetEventLogType Definition

Attribute	Value				
BrowseName	BACnetEventLogType				
IsAbstract	False				
EventNotifier	HistoryRead				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetLogType defined in 7.20.					

The *BACnetEventLogType ObjectType* is a concrete type and can be used directly.

7.25 BACnetStructuredViewType

7.25.1 General

This OPC UA *ObjectType* represents an object that provides a hierarchical view to the BACnet objects contained in a BACnet device. It is formally defined in Table 43.

7.25.2 ObjectType definition

The *BACnetStructuredViewType ObjectType* is formally defined in Table 43.

Table 43 – *BACnetStructuredViewType* Definition

Attribute	Value				
BrowseName	BACnetStructuredViewType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasComponent	Object	<BACnetStructuredView>		BACnetStructuredViewType	OP
0:HasComponent	Object	<BACnetObject>		BACnetObjectType	OP
0:HasProperty	Variable	Node_Type	BACnetNodeType	0:PropertyType	M
0:HasProperty	Variable	Node_Subtype	0:String	0:PropertyType	O
0:HasProperty	Variable	Subordinate_List	BACnetDeviceObjectPropertyReference[]	0:PropertyType	M
0:HasProperty	Variable	Subordinate_Annotations	0:String[]	0:PropertyType	O

The *BACnetStructuredViewType ObjectType* is an abstract type and cannot be used directly.

7.25.3 ObjectType Description

7.25.3.1 <BACnetStructuredViewName>

All *BACnetStructureViews* of the *Subordinate_List* shall be referenced with a *0:HasComponent* Reference.

7.25.3.2 <BACnetObjectType>

All *BACnetObjects* of the *Subordinate_List* shall be referenced with a *0:HasComponent Reference*.

7.25.3.3 Node_Type

This *Property* represents the type of BACnet node.

7.25.3.4 Variable Node_Subtype

This *Property* is a string of printable characters whose content is not restricted. It provides a more specific classification of the object in the hierarchy of objects, providing a short description of the item represented by the node.

7.25.3.5 Variable Subordinate_List

This property is an array of BACnetDeviceObjectPropertyReference that defines the members of the current Structured View.

By including references to 'child' Structured View objects, multilevel hierarchies may be created.

If the optional device identifier is not present for a particular Subordinate_List member, then that object must reside in the same device that maintains the Structured View object. To avoid recursion, it is suggested that a single Structured View object should be referenced only once in the hierarchy. If the size of the Subordinate_List array is changed, the size of the Subordinate_Annotations array, if present, shall also be changed to the same size.

7.25.3.6 Variable Subordinate_Annotations

This property, an array of String, shall be used to define a text string description for each member of the Subordinate_List. The content of these strings is not restricted. If the size of this array is changed, the size of the Subordinate_List array shall also be changed to the same size.

7.26 BACnetNotifierType

7.26.1 ObjectType definition

The *BACnetNotifierType* is formally defined in Table 44.

Table 44 – *BACnetNotifierType* Definition

Attribute	Value				
BrowseName	BACnetNotifierType				
IsAbstract	True				
EventNotifier	SubscribeToEvents				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetObjectType* defined in 7.1.					
0:HasProperty	Variable	Recipient_List	BACnetDestination	0:PropertyType	M

The *BACnetNotifierType ObjectType* is an abstract type and cannot be used directly.

7.26.2 ObjectType Description

7.26.2.1 Variable Recipient_List

This OPC UA *Property*, of *DataType BACnetDestination*, represents the BACnet property Recipient_List. The *BACnetDestination DataType* is defined in 10.5.9.

7.27 BACnetNotificationClassType

7.27.1 ObjectType definition

The *BACnetNotificationClass* is formally defined in Table 45.

Table 45 – *BACnetNotificationClassType* Definition

Attribute	Value				
BrowseName	BACnetNotificationClassType				
IsAbstract	False				
EventNotifier	SubscribeToEvents				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetNotifierType* defined in 7.26.					
0:HasProperty	Variable	Notification_Class	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	Priority	0:Byte[3]	0:PropertyType	M
0:HasProperty	Variable	Ack_Required	BACnetEventTransitionBits	0:PropertyType	M

The *BACnetNotificationClass ObjectType* is a concrete type and can be used directly.

Instances of this ObjectType are used as event notifiers and have the EventNotifier attribute always set to SubscribeToEvents.

7.27.2 ObjectType Description

7.27.2.1 Variable NotificationClass

This OPC UA Property, of type UInt32, indicates the numeric value of the BACnet Notification Class object used in event initiating objects to refere to the BACnet Notification Class object.

7.27.2.2 Priority

This *Property*, of type Byte[3], shall convey the priority to be used for event notifications for TO_OFFNORMAL, TO_FAULT, and TO_NORMAL events, respectively. Priorities shall range from 0 - 255 inclusive. A lower number indicates a higher priority.

7.27.2.3 Ack_Required

This OPC UA *Property*, of *DataType BACnetEventTransitionBits*, represents the BACnet property Ack_Required. The *BACnetEventTransitionBits DataType* is defined in 10.3.3.

8 ObjectTypes used for grouping of object properties

8.1 BACnetTimeManagementType

8.1.1 ObjectType definition

The *BACnetTimeManagementType* is formally defined in Table 46.

Table 46 – BACnetTimeManagementType Definition

Attribute	Value				
BrowseName	BACnetTimeManagementType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Local_Date	BACnetDate	0:PropertyType	O
0:HasProperty	Variable	Local_Time	BACnetTime	0:PropertyType	O
0:HasProperty	Variable	UTC_Offset	0:Int16	0:PropertyType	O
0:HasProperty	Variable	Daylight_Savings_Status	0:Boolean	0:PropertyType	O
0:HasComponent	Method	TimeSynchronization			O

The *BACnetTimeManagementType ObjectType* is a concrete type and can be used directly.

8.1.2 ObjectType Description

8.1.2.1 Variable Local_Date

This OPC UA *Property*, of *DataType BACnetTime*, represents the BACnet property Local_Date. The *BACnetTime DataType* is defined in 10.5.32.

The Local_Date shall indicate the current date. When unable to track the date, the value of the variable shall be initialized to a date on or before January 1, 1990.

8.1.2.2 Variable Local_Time

This OPC UA *Property*, of *DataType BACnetDate*, represents the BACnet property Local_Time. The *BACnetDate DataType* is defined in 10.5.6.

The Local_Time shall indicate the current local time. When unable to track the time, the value of the variable shall be initialized to the time 00:00:00.00.

8.1.2.3 Variable UTC_Offset

This OPC UA *Property*, of *DataType Int16*, represents the BACnet property UTC_Offset.

The UTC_Offset shall indicate the number of minutes (-780 to 780) offset between local standard time and Universal Time Coordinated (UTC). The time zones to the west of the zero

degree meridian shall be positive, and the time zones to the east of the zero degree meridian shall be negative.

8.1.2.4 Variable Daylight_Savings_Status

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Daylight_Savings_Status.

When the Daylight_Savings_Status variable value is true, daylight savings time is in effect at the BACnet device's location.

8.1.2.5 Method TimeSynchronization

This Method is used to update the time of a BACnet device. See B.3 for more details.

Signature

```
TimeSynchronization (
   [in]  UtcTime     Time
   );
```

Argument	Description
Time	The UTC time used to update the time of the BACnet device.

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

8.2 BACnetAutomaticTimeSynchronizationMasterType

8.2.1 ObjectType definition

The *BACnetAutomaticTimeSynchronizationMasterType* is formally defined in Table 47.

Table 47 – BACnetAutomaticTimeSynchronizationMasterType Definition

Attribute	Value				
BrowseName	BACnetAutomaticTimeSynchronizationMasterType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetTimeManagementType* defined in 8.1.1					
0:HasProperty	Variable	Time_Synchronization_Recipients	BACnetRecipient[]	0:PropertyType	O
0:HasProperty	Variable	UTC_Time_Synchronization_Recipients	BACnetRecipient[]	0:PropertyType	O
0:HasProperty	Variable	Time_Synchronization_Interval	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Align_Intervals	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	Interval_Offset	0:UInteger	0:PropertyType	M
0:HasComponent	Method	AddTimeSynchronizationRecipients			M
0:HasComponent	Method	RemoveTimeSynchronization Recipients			M

The *BACnetTimeManagementType ObjectType* is a concrete type and can be used directly.

8.2.2 ObjectType Description

8.2.2.1 Variable Time_Synchronization_Recipients

This OPC UA *Property*, of *DataType BACnetRecipient* [], represents the BACnet property Time_Synchronization_Recipients. The *BACnetRecipient DataType* is defined in 10.6.9.

The Time_Synchronization_Recipients value contains a list of devices that the BACnet device may send time synchronizations to. When the list is empty, the device performs no time synchronization.

8.2.2.2 Variable UTC_Time_Synchronization_Recipients

This OPC UA *Property*, of *DataType BACnetRecipient* [], represents the BACnet property UTC_Time_Synchronization_Recipients. The *BACnetRecipient DataType* is defined in 10.6.9.

The UTC_Time_Synchronization_Recipients value contains a list of devices that the BACnet device may send UTC time synchronizations to. When the list is empty, the device performs no UTC time synchronization.

8.2.2.3 Variable Time_Synchronization_Interval

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Time_Synchronization_Interval.

The Time_Synchronization_Interval specifies the periodic interval in minutes at which local and/or UTC time synchronization requests shall be sent. If the variable value is zero, then all time synchronization shall be disabled.

8.2.2.4 Variable Align_Intervals

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Align_Intervals.

The Align_Intervals indicates whether clock-aligned periodic logging shall be enabled. When clock-aligned periodic logging is enabled, and the Log_Interval is a factor of an hour, or day, then the beginning of the priod specified for time synchronization shall align with the next clock hour, or dayAlign_Intervals.

8.2.2.5 Variable Interval_Offset

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Interval_Offset.

The Interval_Offset value indicates an offset in minutes from the period specified for time synchronization and the time that the synchronization requests are actually sent. The value used shall be the Interval_Offset modulo the value of Time_Synchronization_Interval. For example, if the Time_Synchronization_Interval is 30 and the Interval_Offset is 31, then the offset shall be 1.

8.2.2.6 Method AddTimeSynchronizationRecipients

This Method adds entries to the BACnet property Time_Synchronization_Recipients or UTC_Time_Synchronization_Recipients.

Signature

```
AddTimeSynchronizationRecipients (
   [in]  0:Boolean                  AddToUtcList
   [in]  BACnetRecipient []         TimeSynchronizationRecipients
   [out] 0:UInt32                   FirstFailedElementNumber
   );
```

Argument	Description
AddToUtcList	Indicates if the recipient is added to Time_Synchronization_Recipients or to UTC_Time_Synchronization_Recipients.
TimeSynchronizationRecipients	Array of time syncronization recipients. Based on the AddToUtcList parameter, the recipients are added to the BACnet property Time_Synchronization_Recipients or UTC_Time_Synchronization_Recipients. The *BACnetRecipient DataType* is defined in 10.6.9.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the TimeSynchronizationRecipients. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE and DATATYPE_NOT_SUPPORTED
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadOutOfMemory	This status is returned for the BACnet error code NO_SPACE_TO_ADD_LIST_ELEMENT

8.2.2.7 Method RemoveTimeSynchronizationRecipients

This Method removes entries from the BACnet property Time_Synchronization_Recipients or UTC_Time_Synchronization_Recipients.

Signature

```
RemoveTimeSynchronizationRecipients (
   [in]  0:Boolean                    RemoveFromUtcList
   [in]  BACnetRecipient []           TimeSynchronizationRecipients
   [out] 0:UInt32                     FirstFailedElementNumber
   );
```

Argument	Description
RemoveFromUtcList	Indicates if the recipient is removed from Time_Synchronization_Recipients or to UTC_Time_Synchronization_Recipients.
TimeSynchronizationRecipients	Array of time synchronization recipients. Based on the AddToUtcList parameter, the recipients are removed from the BACnet property Time_Synchronization_Recipients or UTC_Time_Synchronization_Recipients. The *BACnetRecipient DataType* is defined in 10.6.9.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the TimeSynchronizationRecipients. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadNotFound	This status is returned for the BACnet error code LIST_ELEMENT_NOT_FOUND

8.3 BACnetBackupRestoreType

8.3.1 ObjectType definition

The *BACnet*BackupRestore*Type* is formally defined in Table 48.

Table 48 – BACnetBackupRestoreType Definition

Attribute	Value				
BrowseName	BACnetBackupRestoreType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Configuration_Files	BACnetDeviceObject PropertyReference []	0:PropertyType	M
0:HasProperty	Variable	Last_Restore_Time	BACnetTimeStamp	0:PropertyType	M
0:HasProperty	Variable	Backup_Failure_Timeout	0:UInt16	0:PropertyType	M
0:HasProperty	Variable	Backup_Preparation_Time	0:UInt16	0:PropertyType	O
0:HasProperty	Variable	Restore_Preparation_Time	0:UInt16	0:PropertyType	O
0:HasProperty	Variable	Restore_Completion_Time	0:UInt16	0:PropertyType	O
0:HasProperty	Variable	Backup_And_Restore_State	BACnetBackupState	0:PropertyType	O
0:HasComponent	Method	BACnetBackup			M
0:HasComponent	Method	BACnetRestore			M

The *BACnet*BackupRestore*Type ObjectType* is a concrete type and can be used directly.

8.3.2 ObjectType Description

8.3.2.1 Variable Configuration_Files

This OPC UA *Property*, of *DataType BACnetDeviceObjectPropertyReference* [], represents the BACnet property Configuration_Files. The *BACnetDeviceObjectPropertyReference DataType* is defined in 10.5.9.

The Configuration_Files represents a collection of file names within a device that define the device's image and that can be backed up. Note that the content is only required to be valid when a backup is in progress.

8.3.2.2 Variable Last_Restore_Time

This OPC UA *Property*, of *DataType BACnetTimeStamp*, represents the BACnet property Configuration_Files. The *BACnetTimeStamp DataType* is defined in 10.6.11.

The Last_Restore_Time value represents the time at which the device image was last backed up.

8.3.2.3 Variable Backup_Failure_Timeout

This OPC UA *Property*, of *DataType UInt16*, represents the BACnet property Backup_Failure_Timeout.

The Backup_Failure_Timeout value represents the time, in seconds, that the device being backed up or restored must wait until ending the backup or restore operation.

8.3.2.4 Variable Backup_Preparation_Time

This OPC UA *Property*, of *DataType UInt16*, represents the BACnet property Backup_Preparation_Time.

The Backup_Preperation_Time value represents the time, in seconds, that the device that is being backed up may remain unresponsive after the backup procedure is initiated.

8.3.2.5 Variable Restore_Preparation_Time

This OPC UA *Property*, of *DataType UInt16*, represents the BACnet property Restore_Preparation_Time.

The Restore_Preperation_Time value represents the time, in seconds, that the device that is being restored may remain unresponsive after the restore procedure is initiated.

8.3.2.6 Variable Restore_Completion_Time

This OPC UA *Property*, of *DataType UInt16*, represents the BACnet property Restore_Completion_Time.

The Restore_Completion_Time variable value represents the time, in seconds, that the device that is being restored may remain unresponsive after the restore procedure has ended.

8.3.2.7 Variable Backup_And_Restore_State

This OPC UA *Property*, of *DataType BACnetBackupState*, represents the BACnet property Backup_And_Restore_State. The *BACnetBackupState DataType* is defined in 10.4.3.

The Backup_And_Restore_State value represents the current state of the device that is performing a backup or restore procedure.

8.3.2.8 Method BACnetBackup

This Method is used to trigger a backup.

Signature

```
BACnetBackup (
    );
```

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

8.3.2.9 Method BACnetRestore

This Method is used to trigger a restore.

Signature

```
BACnetRestore (
    );
```

Method Result Codes

ResultCode	Description
	Common StatusCodes defined in OPC 10000-4

8.4 BACnetMstpMasterType

8.4.1 ObjectType definition

The *BACnetMstpMasterType* is formally defined in Table 49.

Table 49 – BACnetMstpMasterType Definition

Attribute		Value			
BrowseName		BACnetMstpMasterType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Max_Master	0:Byte	0:PropertyType	M
0:HasProperty	Variable	Max_Info_Frames	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Slave_Proxy_Enable	0:Boolean	0:PropertyType	O
0:HasProperty	Variable	Manual_Slave_Address_Binding	BACnetAddressBinding[]	0:PropertyType	O
0:HasProperty	Variable	Auto_Slave_Discovery	0:Boolean	0:PropertyType	O
0:HasProperty	Variable	Slave_Address_Binding	BACnetAddressBinding[]	0:PropertyType	O

The *BACnetMstpMasterType ObjectType* is a concrete type and can be used directly.

8.4.2 ObjectType Description

8.4.2.1 Variable Max_Master

This OPC UA *Property*, of *DataType Byte*, represents the BACnet property Max_Master.

The Max_Master value represents the highest possible maximum node address and shall be equal to or less than 127. The default value shall be 127.

8.4.2.2 Variable Max_Info_Frames

This OPC UA *Property*, of *DataType UInteger*, represents the BACnet property Max_Info_Frames.

The Max_Info_Frames value represents the maximum number of information frames that may be sent before passing the communications token. The default value shall be 1.

8.4.2.3 Variable Slave_Proxy_Enable

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Slave_Proxy_Enable.

When the Slave_Proxy_Enable value is true, the device will perform Slave/Proxy functions on the device MS/TP port.

8.4.2.4 Variable Manual_Slave_Address_Binding

This OPC UA *Property*, of *DataType BACnetAddressBinding* [], represents the BACnet property Manual_Slave_Address_Binding. The *BACnetAddressBinding DataType* is defined in 10.5.3.

The Manual_Slave_Address_Binding value represents a collection of manually configured slave devices for which this device is acting as a Slave Proxy.

8.4.2.5 Variable Auto_Slave_Discovery

This OPC UA *Property*, of *DataType Boolean*, represents the BACnet property Auto_Slave_Discovery.

When true, the Auto_Slave_Discovery value indicates that the MS/TP master will perform automatic slave detection functions on the master MS/TP port.

8.4.2.6 Variable Slave_Address_Binding

This OPC UA *Property*, of *DataType BACnetAddressBinding* [], represents the BACnet property Slave_Address_Binding. The *BACnetAddressBinding DataType* is defined in 10.5.3.

The Slave_Address_Binding value represents a collection of slave devices for which this device is acting as a Slave Proxy. This set includes the manually configured devices represented by the Master_Slave_Address_Binding and automatically discovered slave devices.The Slave_Address_Binding shall be periodically maintained, but the period at which the list is maintained shall be locally defined.

8.5 BACnetDeviceRestartType

8.5.1 ObjectType definition

The *BACnetDeviceRestartType* is formally defined in Table 50.

Table 50 – BACnetDeviceRestartType Definition

Attribute	**Value**				
BrowseName	BACnetDeviceRestartType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Last_Restart_Reason	BACnetRestartReason	0:PropertyType	M
0:HasProperty	Variable	Time_Of_Device_Restart	BACnetTimeStamp	0:PropertyType	M
0:HasProperty	Variable	Restart_Notification_Recipients	BACnetRecipient[]	0:PropertyType	M
0:HasComponent	Method	AddRestartRecipients			M
0:HasComponent	Method	RemoveRestartRecipients			M

The *BACnetDeviceRestartType ObjectType* is a concrete type and can be used directly.

8.5.2 ObjectType Description

8.5.2.1 Variable Last_Restart_Reason

This OPC UA *Property*, of *DataType BACnetRestartReason*, represents the BACnet property Last_Restart_Reason. The *BACnetRestartReason DataType* is defined in 10.4.29.

The Last_Restart_Reason value indicates the reason for the last device restart. Some possible values for this variable are UNKNOWN, COLDSTART, WARMSTART, DETECTED_POWER_LOST, DETECTED_POWER_OFF, HARDWARE_WATCHDOG, SOFTWARE_WATCHDOG, and SUSPENDED.

8.5.2.2 Variable Time_Of_Device_Restart

This OPC UA *Property*, of *DataType BACnetTimeStamp*, represents the BACnet property Time_Of_Device_Restart. The *BACnetTimeStamp DataType* is defined in 10.6.11.

The Time_Of_Device_Restart value represents the time at which the device was last restarted.

8.5.2.3 Variable Restart_Notification_Recipients

This OPC UA *Property*, of *DataType BACnetRecipient* [], represents the BACnet property Restart_Notification_Recipients. The *BACnetRecipient DataType* is defined in 10.6.9.

The Restart_Notification_Recipients variable value is a collection of recipients that shall be notified when the device is restarted. The default value is the network broadcast address. If the collection size is zero, no restart notifications shall be sent.

8.5.2.4 Method AddRestartRecipients

This Method adds entries to the BACnet property Restart_Notification_Recipients.

Signature

```
AddRestartRecipients (
   [in]  BACnetRecipient []          RestartNotificationRecipients
   [out] 0:UInt32                    FirstFailedElementNumber
   );
```

Argument	Description
RestartNotificationRecipients	Array of time restart notification recipients. The recipients are added to the BACnet property Restart_Notification_Recipients. The *BACnetRecipient DataType* is defined in 10.6.9.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the RestartNotificationRecipients. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE and DATATYPE_NOT_SUPPORTED
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadOutOfMemory	This status is returned for the BACnet error code NO_SPACE_TO_ADD_LIST_ELEMENT

8.5.2.5 Method RemoveRestartRecipients

This Method removes entries from the BACnet property Restart_Notification_Recipients.

Signature

```
RemoveRestartRecipients (
   [in]  BACnetRecipient []          RestartNotificationRecipients
   [out] 0:UInt32                    FirstFailedElementNumber
   );
```

Argument	Description
RestartNotificationRecipients	Array of restart notifctation recipients. The recipients are removed from the BACnet property Restart_Notification_Recipients. The *BACnetRecipient DataType* is defined in 10.6.9.
FirstFailedElementNumber	The numerical position, starting at 1, of the failed element in the RestartNotificationRecipients. If the call succeeds or fails for other reasons, the returned value shall be 0.

Method Result Codes

ResultCode	Description
BadNodeIdUnknown	This status is returned for the BACnet error codes UNKNOWN_OBJECT and UNKNOWN_PROPERTY
BadTypeMismatch	This status is returned for the BACnet error codes INVALID_DATATYPE
BadOutOfRange	This status is returned for the BACnet error code VALUE_OUT_OF_RANGE
BadNotWritable	This status is returned for the BACnet error code WRITE_ACCESS_DENIED
BadNotFound	This status is returned for the BACnet error code LIST_ELEMENT_NOT_FOUND

8.6 BACnetChangeOfStateCountType

8.6.1 General

This OPC UA *ObjectType* represents a set of variables related to the *BACnet Change_of_State* concept. The type represents metadata related to an object's Present_Value: when the Present_Value changed, how often it changed, and how often this count is reset.

8.6.2 ObjectType definition

The *BACnetChangeOfStateCountType* is formally defined in Table 51.

Table 51 – *BACnetChangeOfStateCountType* Definition

Attribute	**Value**				
BrowseName	BACnetChangeOfStateCountType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Change_Of_State_Time	BACnetDateTime	0:PropertyType	M
0:HasProperty	Variable	Change_Of_State_Count	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	Time_Of_State_Count_Reset	0:DateTime	0:PropertyType	M
0:HasComponent	Method	Reset			M

The *BACnetChangeOfStateCountType ObjectType* is a concrete type and can be used directly.

8.6.3 ObjectType Description

8.6.3.1 Variable Change_Of_State_Time

This OPC UA Property, of type DateTime, represents the date and time at which the most recent change of state occurred. A "change of state" shall be defined as any event that alters the Present_Value property.

8.6.3.2 Variable Change_Of_State_Count

This OPC UA Property, of type UInt32, represents the number of times that the Present_Value property has changed state since the Change_Of_State_Count property was most recently set to a zero value. A "change of state" shall be defined as any event that alters the Present_Value property.

8.6.3.3 Variable Time_Of_State_Count_Reset

This OPC UA Property, of type DateTime, represents the date and time at which the Change_Of_State_Count property was most recently set to a zero value.

8.6.3.4 Method Reset

This Method is used to trigger a reset.

Signature

```
BACnetReset (
   );
```

Method Result Codes

ResultCode	**Description**
	Common StatusCodes defined in OPC 10000-4

8.7 BACnetElapsedActiveTimeType

8.7.1 General

This OPC UA *ObjectType* represents a set of variables related to the *BACnet Elapsed_Active_Time* concept. The type represents metadata related to an object's Present_Value: the accumulated number of seconds that the Present_Value variable has been Active, and the date and time at which the Elapsed_Active_Time was most recently set to a zero value.

8.7.2 ObjectType definition

The *BACnetElapsedActiveTimeType* is formally defined in Table 52.

Table 52 – *BACnetElapsedActiveTimeType* Definition

Attribute	**Value**				
BrowseName	BACnetElapsedActiveTimeType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Elapsed_Active_Time	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	Time_Of_Active_Time_Reset	0:DateTime	0:PropertyType	M
0:HasComponent	Method	Reset			M

The *BACnetElapsedActiveTimeType ObjectType* is a concrete type and can be used directly.

8.7.3 ObjectType Description

8.7.3.1 Variable Elapsed_Active_Time

This OPC UA Property, of type UInt32, represents the accumulated number of seconds that the Present_Value property has had the value Active since the Elapsed_Active_Time property was most recently set to a zero value.

8.7.3.2 Variable Time_Of_Active_Time_Reset

This OPC UA Property, of type DateTime, represents the date and time at which the Elapsed_Active_Time property was most recently set to a zero value.

8.7.3.3 Method Reset

This Method is used to trigger a reset.

Signature

```
BACnetReset (
    );
```

Method Result Codes

ResultCode	**Description**
	Common StatusCodes defined in OPC 10000-4

8.8 BACnetEventReportingType

8.8.1 General

This OPC UA *ObjectType* represents a set of variables related to the *BACnet event Reporting* concept. Event reporting is used to define alarm or event conditions that are intrinsic to a particular BACnet object type and require only the properties of that object type.

This OPC UA *ObjectType* covers both BACnet event detection and event reporting parameters.

8.8.2 ObjectType definition

The *BACnetEventReportingType* is formally defined in Table 53.

Table 53 – *BACnetEventReportingType* Definition

Attribute	**Value**				
BrowseName	BACnetEventReportingType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Event_State	BACnetEventState	0:PropertyType	M
0:HasProperty	Variable	Notification_Class	UInt32	0:PropertyType	M
0:HasProperty	Variable	Event_Enable	BACnetEventTransitionBits	0:PropertyType	M
0:HasProperty	Variable	Acked_Transitions	BACnetEventTransitionBits	0:PropertyType	M
0:HasProperty	Variable	Notify_Type	BACnetNotifyType	0:PropertyType	M
0:HasProperty	Variable	Event_Time_Stamps	BACnetTimeStamp [3]	0:PropertyType	M
0:HasProperty	Variable	Event_Message_Texts	String [3]	0:PropertyType	O
0:HasProperty	Variable	Event_Message_Texts_ Config	String [3]	0:PropertyType	O
0:HasProperty	Variable	Event_Detection_Enable	0:Boolean	0:PropertyType	O
0:HasProperty	Variable	Event_Algorithm_Inhibit _Ref	BACnetDeviceObjectProperty Reference	0:PropertyType	O
0:HasProperty	Variable	Event_Algorithm_Inhibit	0:Boolean	0:PropertyType	O
0:HasComponent	Object	EventAlgorithm		BACnetEvent AlgorithmType	O

The *BACnetEventReportingType ObjectType* is a concrete type and can be used directly.

8.8.3 ObjectType Description

8.8.3.1 Variable Event_State

This OPC UA Property, *of DataType BACnetEventState*, is included in order to provide a way to determine whether this object has an active event state associated with it. If the object does not support event reporting then the value of this property shall be Normal. The *BACnetEventState DataType* is defined in 10.4.10.

While BACnet object types normally require the BACnet property Event_State, the OPC UA Object EventReporting is O. In absence of an event algorithm the property Event_State is defined by BACnet to be Normal. And therefore does not provide any valuable information and is not visible in this case.

8.8.3.2 Variable Notification_Class

This OPC UA Property, of type UInt32, shall specify the instance of the Notification Class object to use for event-notification description.

The value of this property results in a HasEventSource Reference from the OPC UA Object representing the BACnet Notification Class object and the BACnet object containing the event reporting object. The inverse reference shall be supported if event reporting is available.

8.8.3.3 Variable Event_Enable

This OPC UA Property, of type *BACnetEventTransitionBits*, is a set of three flags that separately configure delivery of TO_OFFNORMAL, TO_FAULT, and TO_NORMAL notifications. The *BACnetEventTransitionBits DataType* is defined in 10.3.3.

8.8.3.4 Variable Acked_Transitions

This OPC UA Property, of type *BACnetEventTransitionBits*, is a set of three flags that separately indicate the acknowledgement state for TO_OFFNORMAL, TO_FAULT, and TO_NORMAL events. Each flag shall have the value True if no event of that type has ever occurred for the object. The *BACnetEventTransitionBits DataType* is defined in 10.3.3.

8.8.3.5 Variable Notify_Type

This OPC UA Property, of type *BACnetNotifyType*, shall indicate whether the notifications generated by the object are Events or Alarms. The *BACnetNotifyType DataType* is defined in 10.4.20.

8.8.3.6 Variable Event_Time_Stamps

This OPC UA Property, of type BACnetTimeStamp [3], shall indicate the times of the last TO_OFFNORMAL, TO_FAULT, and TO_NORMAL events. The *BACnetTimeStamp* DataType is defined in 10.6.11.

8.8.3.7 Variable Event_Message_Texts

This OPC UA Property, of type String [3], shall indicate the message text values of the last TO_OFFNORMAL, TO_FAULT, and TO_NORMAL events. If a particular type of event has yet to occur, an empty string shall be stored in the respective array element.

8.8.3.8 Variable Event_Detection_Enable

This OPC UA Property, of *DataType* Boolean, represents the BACnet property Event_Detection_Enable.

When true, the Event_Detection_Enable value indicates that intrinsic reporting is enabled in the device and that the device should be considered by event summarization services. This value is expected to be statically configured and is not expected to change dynamically.

When false, the Event_State shall be NORMAL and the variables Acked_Transitions, Event_Time_Stamps, and Event_Message_Texts shall be equal to their respective initial conditions.

8.8.3.9 Variable Event_Algorithm_Inhibit_Ref

This OPC UA Property, of *DataType BACnetDeviceObjectPropertyReference* represents the BACnet property Event_Algorithm_Inhibit_Ref. The *DataType* is defined in 10.5.9.

The Event_Algorithm_Inhibit_Ref variable value represents the property which controls the value of the property Event_Algorithm_Inhibit

8.8.3.10 Variable Event_Algorithm_Inhibit

This OPC UA Property, of *DataType* Boolean, represents the BACnet property Event_Algorithm_Inhibit.

When the Event_Algorithm_Inhibit variable value is true, the event algorithm is disabled for the object. When the Event_Algorithm_Inhibit_Ref variable is initialized (its value is other than

4194303) then the Event_Algorithm_Inhibit variable shall reflect the value of the property referenced by Event_Algorithm_Inhibit_Ref.

8.8.3.11 Object EventAlgorithm

This object, of type *BACnetEventAlgorithmType* describes the BACnet event algorithm used by the physical device.

8.9 BACnetEventAlgorithmType

8.9.1 General

This OPC UA *ObjectType* represents an abstract type related to the *BACnet Algorithmic Change Reporting* mechanism. Algorithmic change reporting is a general concept that can be applied to properties of any object. The actual algorithms used to generate the events are similar to those used by *EventReporting*.

The subtypes of BACnetEventAlgorithmType defined in the following clauses represent the event algorithms defined in chapter 13.2 Event Algorithms in the BACnet standard.

8.9.2 ObjectType definition

The *BACnetEventAlgorithmType* is formally defined in Table 54.

Table 54 – *BACnetEventAlgorithmType* Definition

Attribute	**Value**				
BrowseName	BACnetEventAlgorithmType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	TimeDelay	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	TimeDelayNormal	0:UInt32	0:PropertyType	M

The *BACnetEventAlgorithmType* is an abstract type and cannot be used directly.

8.9.3 ObjectType Description

8.9.3.1 Variable TimeDelay

This OPC UA *Property*, of type *UInt32*, is the TimeDelay parameter for the object's event algorithm. It represents the time, in seconds, that the offnormal conditions must exist before an offnormal event state is indicated.

It is normally used to represent the BACnet property Time_Delay.

8.9.3.2 Variable TimeDelayNormal

This OPC UA *Property*, of type *UInt32*, is the TimeDelayNormal parameter for the object's event algorithm. It represents the time, in seconds, that the normal conditions must exist before a NORMAL event state is indicated.

It is normally used to represent the BACnet property Time_Delay_Normal. If the BACnet property Time_Delay_Normal is not present it has the same value as the BACnet property Time_Delay.

8.10 BACnetChangeOfStateAlgorithmType

8.10.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet Change of State* event algorithm.

The *BACnetChangeOfStateAlgorithmType* is formally defined in Table 55.

Table 55 – *BACnetChangeOfStateAlgorithmType* Definition

Attribute	Value				
BrowseName	BACnetChangeOfStateAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	AlarmValues	0:BaseDataType{Any}	0:PropertyType	M

The *BACnetChangeOfStateAlgorithmType ObjectType* is a concrete type and can be used directly.

8.10.2 ObjectType Description

8.10.2.1 Variable AlarmValues

This OPC UA Property is the AlarmValues parameter for the object's event algorithm. It contains a list of discrete values that represent the offnormal values.

8.11 BACnetCommandFailureAlgorithmType

8.11.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet Command Failure* event algorithm.

The *BACnetCommandFailureAlgorithmType* is formally defined in Table 56.

Table 56 – *BACnetCommandFailureAlgorithmType* Definition

Attribute	Value				
BrowseName	BACnetCommandFailureAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	FeedbackValueRef	BACnetDeviceObjectProperty Reference	0:PropertyType	M

The *BACnetCommandFailureAlgorithmType ObjectType* is a concrete type and can be used directly.

8.11.2 ObjectType Description

8.11.2.1 Variable FeedbackValueRef

This OPC UA Property, of *DataType BACnetDeviceObjectPropertyReference*, references the value that is used as the pFeedbackValue parameter of the BACnet event algorithm. The *DataType* is defined in 10.5.10.

If the event algortithm is used in objects other than event enrollment, the value contains the reference to the BACnet Feedback_Value property.

8.12 BACnetFloatingLimitAlgorithmType

8.12.1 ObjectType definition

This OPC UA *ObjectType* represents the BACnet Floating Limit event algorithm.

The *BACnetFloatingLimitAlgorithmType* is formally defined in Table 57.

Table 57 – BACnetFloatingLimitAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetFloatingLimitAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	SetpointReference	BACnetDeviceObject PropertyReference	0:PropertyType	M
0:HasProperty	Variable	LowDiffLimit	0:Float	0:PropertyType	M
0:HasProperty	Variable	HighDiffLimit	0:Float	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:Float	0:PropertyType	M

The *BACnetFloatingLimitAlgorithmType ObjectType* is a concrete type and can be used directly.

8.12.2 ObjectType Description

8.12.2.1 Variable SetpointReference

This OPC UA Property, of *DataType BACnetDeviceObjectPropertyReference*, references the value that is used as the SetpointReference parameter of the BACnet event algorithm. The *DataType* is defined in 10.5.10.

The SetpointReference variable value represents a reference to the value that defines the reference interval.

8.12.2.2 Variable LowDiffLimit

This OPC UA *Property*, of *DataType Float*, is the LowDiffLimit parameter for the object's event algorithm.

Subtracted from the setpoint reference value, the LowDiffLimit variable value represents the lower limit of the range considered normal.

8.12.2.3 Variable HighDiffLimit

This OPC UA *Property*, of *DataType Float*, is the HighDiffLimit parameter for the object's event algorithm.

Added to the setpoint reference value, the HighDiffLimit variable value represents the upper limit of the range considered normal.

8.12.2.4 Variable Deadband

This OPC UA *Property*, of *DataType Float*, is the Deadband parameter for the object's event algorithm.

The Deadband variable value represents the deadband that is applied to the limit before a return to Normal event state is indicated.

8.13 BACnetOutOfRangeAlgorithmType

8.13.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet out of Range* event algorithm.

The *BACnetOutOfRangeAlgorithmType* is formally defined in Table 58.

Table 58 – *BACnetOutOfRangeAlgorithmType* Definition

Attribute	Value				
BrowseName	BACnetOutOfRangeAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	HighLimit	0:Float	0:PropertyType	M
0:HasProperty	Variable	LowLimit	0:Float	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:Float	0:PropertyType	M
0:HasProperty	Variable	LimitEnable	BACnetLimitEnable	0:PropertyType	M

The *BACnetOutOfRangeAlgorithmType ObjectType* is a concrete type and can be used directly.

8.13.2 ObjectType Description

8.13.2.1 Variable HighLimit

This OPC UA Property, of *DataType Float*, is the HighLimit parameter for the object's event algorithm.

The HighLimit value represents the upper limit of the range considered normal.

8.13.2.2 Variable LowLimit

This OPC UA Property, of *DataType Float*, is the LowLimit parameter for the object's event algorithm.

The LowLimit value represents the lower limit of the range considered normal.

8.13.2.3 Variable Deadband

This OPC UA Property, of *DataType Float*, is the Deadband parameter for the object's event algorithm.

The Deadband value represents the deadband that is applied to the limit before a return to Normal event state is indicated.

8.13.2.4 Variable LimitEnable

This OPC UA Property, of *DataType BACnetLimitEnable*, shall indicate three separate flags that enable and disable the delivery of TO_OFFNORMAL, TO_FAULT, and TO_NORMAL notifications. The *DataType BACnetLimitEnable* is defined in 10.3.4.

8.14 BACnetBufferReadyAlgorithmType

8.14.1 ObjectType definition

This OPC UA *ObjectType* represents the BACnet buffer ready event algorithm.

The *BACnetBufferReadyAlgorithmType* is formally defined in Table 59.

Table 59 – BACnetBufferReadyAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetBufferReadyAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	Threshold	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	PreviousCount	0:UInt32	0:PropertyType	M

The *BACnetBufferReadyAlgorithmType ObjectType* is a concrete type and can be used directly.

8.14.2 ObjectType Description

8.14.2.1 Variable Threshold

This OPC UA Property, of *DataType UInt32*, is the Threshold parameter for the object's event algorithm.

The Threshold value represents the number of records that, when added to the log buffer, will result in a transition to NORMAL. If this variable has a value of zero, then no transitions will be indicated by the algorithm.

8.14.2.2 Variable PreviousCount

This OPC UA Property, of *DataType UInt32*, is the PreviousCount parameter for the object's event algorithm.

The PreviousCount value represents the value of pMonitoredValue at the time the most recent transition to NORMAL was indicated. Upon initialization of the event algorithm, this parameter shall be set to the value of pMonitoredValue. When a transition to NORMAL is indicated, this parameter shall be updated to the value of pMonitoredValue.

8.15 BACnetChangeOfBitStringAlgorithmType

8.15.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet change of bitstring* event algorithm.

The *BACnetChangeOfBitStringAlgorithmType* is formally defined in Table 60.

Table 60 – BACnetChangeOfBitStringAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetChangeOfBitStringAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	AlarmValues	0:OptionSet	0:PropertyType	M

The *BACnetChangeOfBitStringAlgorithmType ObjectType* is a concrete type and can be used directly.

8.15.2 ObjectType Description

8.15.2.1 Variable AlarmValues

This OPC UA Property, of *DataType OptionSet*, is the AlarmValues parameter for the object's event algorithm.

The AlarmValues value represents a set of values that are considred off-NORMAL.

The OptionSet DataType contains the bitmask that defines the bits of the monitored value that are significant for comparison with the values in the AlarmValues set. The value is bitwise AND'ed with the monitored value before comparison with the AlarmValues values.

8.16 BACnetChangeOfValueAlgorithmType

8.16.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet change of value* event algorithm.

The *BACnetChangeOfValueAlgorithmType* is formally defined in Table 61.

Table 61 – BACnetChangeOfValueAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetChangeOfValueAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	Increment	0:Float	0:PropertyType	M
0:HasProperty	Variable	Bitmask	0:OptionSet	0:PropertyType	M

The *BACnetChangeOfValueAlgorithmType ObjectType* is a concrete type and can be used directly.

8.16.2 ObjectType Description

8.16.2.1 Variable Increment

This OPC UA Property, of *DataType Float*, is the Increment parameter for the object's event algorithm.

The Increment value represents the positive increment by which a monitored value of Float type must change for a new transition.

8.16.2.2 Variable Bitmask

This OPC UA Property, of *DataType OptionSet*, is the Bitmask parameter for the object's event algorithm.

The Bitmask value represents the bitmask that defines the bits of pMonitoredValue that are significant for detecting a change of value. This value is bit-wise AND'ed with the monitored value before comparison with the value that has caused the last transition to NORMAL.

8.17 BACnetUnsignedRangeAlgorithmType

8.17.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet unsigned range* event algorithm.

The *BACnetUnsignedRangeAlgorithmType* is formally defined in Table 62.

Table 62 – BACnetUnsignedRangeAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetUnsignedRangeAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	LowLimit	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	HighLimit	0:UInteger	0:PropertyType	M

The *BACnetUnsignedRangeAlgorithmType ObjectType* is a concrete type and can be used directly.

8.17.2 ObjectType Description

8.17.2.1 Variable LowLimit

This OPC UA Property, of *DataType UInteger*, is the LowLimit parameter for the object's event algorithm.

The LowLimit value represents the lower limit of the range considered normal.

8.17.2.2 Variable HighLimit

This OPC UA Property, of *DataType UInteger*, is the HighLimit parameter for the object's event algorithm.

The HighLimit variable value represents the upper limit of the range considered normal.

8.18 BACnetChangeOfStatusFlagsAlgorithmType

8.18.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet change of status flags* event algorithm.

The *BACnetChangeOfStatusFlagsAlgorithmType* is formally defined in Table 63.

Table 63 – BACnetChangeOfStatusFlagsAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetChangeOfStatusFlagsAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	SelectedFlags	BACnetStatusFlags	0:PropertyType	M

The *BACnetChangeOfStatusFlagsAlgorithmType ObjectType* is a concrete type and can be used directly.

8.18.2 ObjectType Description

8.18.2.1 Variable SelectedFlags

This OPC UA Property, of *DataType BACnetStatusFlags*, is the SelectedFlags parameter for the object's event algorithm. The *DataType BACnetStatusFlags* is defined in 10.3.7.

The SelectedFlags value represents the flags of the monitored value that are significant for evaluation. A status flag variable value of true indicates that the corresponding monitored value flag is significant for evaluation.

8.19 BACnetDoubleOutOfRangeAlgorithmType

8.19.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet double out of Range* event algorithm.

The *BACnetDoubleOutOfRangeAlgorithmType* is formally defined in Table 64.

Table 64 – BACnetDoubleOutOfRangeAlgorithmType Definition

Attribute	**Value**				
BrowseName	BACnetDoubleOutOfRangeAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	HighLimit	0:Double	0:PropertyType	M
0:HasProperty	Variable	LowLimit	0:Double	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:Double	0:PropertyType	M
0:HasProperty	Variable	LimitEnable	BACnetLimitEnable	0:PropertyType	M

The *BACnetDoubleOutOfRangeAlgorithmType ObjectType* is a concrete type and can be used directly.

8.19.2 ObjectType Description

8.19.2.1 Variable HighLimit

This OPC UA Property, of *DataType Double*, is the HighLimit parameter for the object's event algorithm.

The HighLimit value represents the upper limit of the range considered normal.

8.19.2.2 Variable LowLimit

This OPC UA Property, of *DataType Double*, is the LowLimit parameter for the object's event algorithm.

The LowLimit value represents the lower limit of the range considered normal.

8.19.2.3 Variable Deadband

This OPC UA Property, of *DataType Double*, is the Deadband parameter for the object's event algorithm.

The Deadband variable value represents the deadband that is applied to the limit before a return to Normal event state is indicated.

8.19.2.4 Variable LimitEnable

This OPC UA Property, of *DataType BACnetLimitEnable*, shall indicate three separate flags that enable and disable the delivery of TO_OFFNORMAL, TO_FAULT, and TO_NORMAL notifications. The *DataType BACnetLimitEnable* is defined in 10.3.4.

8.20 BACnetSignedOutOfRangeAlgorithmType

8.20.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet signed out of Range* event algorithm.

The *BACnetSignedOutOfRangeAlgorithmType* is formally defined in Table 65.

Table 65 – BACnetSignedOutOfRangeAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetSignedOutOfRangeAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	HighLimit	0:Integer	0:PropertyType	M
0:HasProperty	Variable	LowLimit	0:Integer	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:Integer	0:PropertyType	M
0:HasProperty	Variable	LimitEnable	BACnetLimitEnable	0:PropertyType	M

The *BACnetSignedOutOfRangeAlgorithmType ObjectType* is a concrete type and can be used directly.

8.20.2 ObjectType Description

8.20.2.1 Variable HighLimit

This OPC UA Property, of *DataType Integer*, is the HighLimit parameter for the object's event algorithm.

The HighLimit value represents the upper limit of the range considered normal.

8.20.2.2 Variable LowLimit

This OPC UA Property, of *DataType Integer*, is the LowLimit parameter for the object's event algorithm.

The LowLimit value represents the lower limit of the range considered normal.

8.20.2.3 Variable Deadband

This OPC UA Property, of *DataType Integer*, is the Deadband parameter for the object's event algorithm.

The Deadband value represents the deadband that is applied to the limit before a return to Normal event state is indicated.

8.20.2.4 Variable LimitEnable

This OPC UA Property, of type BACnetLimitEnable, shall indicate three separate flags that enable and disable the delivery of TO_OFFNORMAL, TO_FAULT, and TO_NORMAL notifications. The *DataType BACnetLimitEnable* is defined in 10.3.4.

8.21 BACnetUnsignedOutOfRangeAlgorithmType

8.21.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet unsigned out of Range* event algorithm.

The *BACnetUnsignedOutOfRangeAlgorithmType* is formally defined in Table 66.

Table 66 – BACnetUnsignedOutOfRangeAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetUnsignedOutOfRangeAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	HighLimit	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	LowLimit	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	LimitEnable	BACnetLimitEnable	0:PropertyType	M

The *BACnetUnsignedOutOfRangeAlgorithmType ObjectType* is a concrete type and can be used directly.

8.21.2 ObjectType Description

8.21.2.1 Variable HighLimit

This OPC UA Property, of *DataType UInteger*, is the HighLimit parameter for the object's event algorithm.

The HighLimit value represents the upper limit of the range that is considered normal.

8.21.2.2 Variable LowLimit

This OPC UA Property, of *DataType UInteger*, is the LowLimit parameter for the object's event algorithm.

The LowLimit value represents the lower limit of the range that is considered normal.

8.21.2.3 Variable Deadband

This OPC UA Property, of *DataType UInteger*, is the Deadband parameter for the object's event algorithm.

The Deadband value represents the deadband that is applied to the limit before a return to Normal event state is indicated.

8.21.2.4 Variable LimitEnable

This OPC UA Property, of *DataType* BACnetLimitEnable, shall indicate three separate flags that enable and disable the delivery of TO_OFFNORMAL, TO_FAULT, and TO_NORMAL notifications. The *DataType BACnetLimitEnable* is defined in 10.3.4.

8.22 BACnetChangeOfCharacterStringAlgorithmType

8.22.1 ObjectType definition

This OPC UA *ObjectType* represents the *BACnet change of status flags* event algorithm.

The *BACnetChangeOfCharacterStringAlgorithmType* is formally defined in Table 67.

Table 67 – BACnetChangeOfCharacterStringAlgorithmType Definition

Attribute	Value				
BrowseName	BACnetChangeOfCharacterStringAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetEventAlgorithmType defined in 8.9.					
0:HasProperty	Variable	AlarmValues	0:String[]	0:PropertyType	M

The *BACnetChangeOfCharacterStringAlgorithmType ObjectType* is a concrete type and can be used directly.

8.22.2 ObjectType Description

8.22.2.1 Variable AlarmValues

This OPC UA Property, of *DataType String*[], is the AlarmValues parameter for the object's event algorithm.

The AlarmValues variable value represents a collection of character strings that are considered alarm values.

8.23 BACnetFaultEvaluationType

8.23.1 General

This OPC UA *ObjectType* represents a set of variables related to the BACnet fault evaluation concept.

This OPC UA *ObjectType* covers BACnet reliability evaluation. Parameters for BACnet fault reporting are covered by *BACnetEventReportingType*.

8.23.2 ObjectType definition

The *BACnetFaultEvaluationType* is formally defined in Table 68.

Table 68 – BACnetFaultEvaluationType Definition

Attribute	Value				
BrowseName	BACnetFaultEvaluationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					
0:HasProperty	Variable	Reliability	BACnetReliability	0:PropertyType	M
0:HasProperty	Variable	Reliability_Evaluation_Inhibit	0:Boolean	0:PropertyType	O
0:HasComponent	Object	FaultAlgorithm		BACnetFault AlgorithmType	O

The *BACnetFaultEvaluationType ObjectType* is a concrete type and can be used directly.

8.23.3 ObjectType Description

8.23.3.1 Variable Reliability

This OPC UA Property, of *DataType BACnetReliability*, represents the BACnet property Reliability. The *BACnetReliability DataType* is defined in 10.4.28.

It provides an indication of whether the Present_Value or the operation of the physical I/O in question is “reliable” as far as the BACnet Device or operator can determine and, if not, why.

8.23.3.2 Variable Reliability_Evaluation_Inhibit

This OPC UA Property, of *DataType Boolean*, represents the BACnet property Reliability_Evaluation_Inhibit.

It indicates whether or not reliability-evaluation is disabled in the object.

8.23.3.3 Object FaultAlgorithm

This object, of type *BACnetFaultAlgorithmType* describes the BACnet fault algorithm used by the physical device.

8.24 BACnetFaultAlgorithmType

8.24.1 General

This OPC UA *ObjectType* represents describes the BACnet fault algorithm used by a BACnet device.

8.24.2 ObjectType definition

The *BACnetFaultAlgorithmType* is formally defined in Table 69.

Table 69 – *BACnetFaultAlgorithmType* Definition

Attribute	Value				
BrowseName	BACnetFaultAlgorithmType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseObjectType* defined in OPC 10000-5					

The *BACnetFaultAlgorithmType* is an abstract type and cannot be used directly.

8.25 BACnetFaultStateAlgorithmType

8.25.1 ObjectType definition

This OPC UA *ObjectType* represents the BACnet state fault algorithm.

The *BACnetFaultStateAlgorithmType* is formally defined in Table 70.

Table 70 – *BACnetFaultStateAlgorithmType* Definition

Attribute	Value				
BrowseName	BACnetFaultStateAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetFaultAlgorithmType defined in 8.24.					
0:HasProperty	Variable	FaultValues	0:BaseDataType []	0:PropertyType	M

The *BACnetFaultStateAlgorithmType ObjectType* is a concrete type and can be used directly.

8.25.2 ObjectType Description

8.25.2.1 Variable FaultValues

The FaultValues parameter provides the values used by the fault state algorithm to determine whether the monitored value is in a fault state.

8.26 BACnetFaultCharacterStringAlgorithmType

8.26.1 ObjectType definition

This OPC UA *ObjectType* represents the BACnet character string fault algorithm.

The *BACnetFaultCharacterStringAlgorithmType* is formally defined in Table 71.

Table 71 – *BACnetFaultCharacterAlgorithmType* Definition

Attribute	Value				
BrowseName	BACnetFaultCharacterStringAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetFaultAlgorithmType defined in 8.24.					
0:HasProperty	Variable	FaultValues	0:String []	0:PropertyType	M

The *BACnetFaultCharacterStringAlgorithmType ObjectType* is a concrete type and can be used directly.

8.26.2 ObjectType Description

8.26.2.1 Variable FaultValues

The FaultValues parameter provides the values used by the fault state algorithm to determine whether the monitored value is in a fault state.

8.27 BACnetFaultStatusFlagsAlgorithmType

8.27.1 ObjectType definition

This OPC UA *ObjectType* represents the BACnet status flags fault algorithm.

The *BACnetFaultStatusFlagsAlgorithmType* is formally defined in Table 72.

Table 72 – *BACnetFaultStatusFlagsAlgorithmType* Definition

Attribute	Value				
BrowseName	BACnetFaultStatusFlagsAlgorithmType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BACnetFaultAlgorithmType defined in 8.24.					

The *BACnetFaultStatusFlagsAlgorithmType ObjectType* is a concrete type and can be used directly.

9 ConditionTypes

9.1 General

9.2 Mapping of BACnet Event Notification to OPC UA Event Fields

The mapping of BACnet Event Notifications to OPC UA Event Fields is defined in Table 73.

Table 73 – Mapping BACnet Event Notification to OPC UA Event Fields

BACnet Event Notificaiton	OPC UA Event Field	Description
Process Identifier		Not visible for OPC UA client. Only used by the BACnetUaMapper internally.
Initiating Device Identifier	BaseEventType::SourceNode BaseEventType::SourceName	
Event Object Identifier		
Time Stamp	BaseEventType::ReceiveTime	The BACnetUaMapper creates the ReceiveTime if the BACnet device sends a sequence number as Time Stamp.
Notification Class	BACnetNotificationType:: Notification_Class	
Priority	BaseEventType::Severity	The BACnet value range 0-255 is mapped to the OPC UA value range of 1-1000. In OPC UA 1000 is the highest Severity. In BACnet 0 is the highest Priority.
Event Type	BaseEventType::EventType	
Message Text	BaseEventType::Message	
Notify Type	BACnetNotificationType:: Notifiy_Type	
AckRequired	AcknowledgableConditionType:: AckedState	
From State	BACnetNotificationType:: From_State	
To State	BACnetNotificationType:: To_State	

9.3 BACnetNotificationType

9.3.1 ObjectType definition

The *BACnetNotificationType* is formally defined in Table 74.

Table 74 – *BACnetNotificationType* Definition

Attribute	Value				
BrowseName	BACnetNotificationType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:AlarmConditionType defined in OPC 10000-9					
0:HasProperty	Variable	Notification_Class	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	Notify_Type	BACnetNotifyType	0:PropertyType	M
0:HasProperty	Variable	From_State	BACnetEventState	0:PropertyType	O
0:HasProperty	Variable	To_State	BACnetEventState	0:PropertyType	M

The *BACnetNotificationType* is an abstract type and cannot be used directly.

BACnet event information From_State and To_State is used to determine the active state of *BACnetFaultNotificationType* and *BACnetEventNotificationType Conditions*.

9.3.2 ObjectType Description

9.3.2.1 Variable Notification_Class

This OPC UA Property, of type UInt32, shall specify the instance of the Notification Class object to use for event-notification distribution.

9.3.2.2 Variable Notifiy_Type

This OPC UA Property, of type *BACnetNotifyType*, shall indicate whether the notifications generated by the object should be Events or Alarms. The value of the property is used as the value of the 'Notify Type' service parameter in event notifications generated by the object. The *DataType BACnetNotifyType* is defined in 10.4.20.

9.4 BACnetFaultNotificationType

9.4.1 ObjectType definition

The *BACnetFaultNotificationType* is formally defined in Table 75.

Table 75 – BACnetFaultNotificationType Definition

Attribute	**Value**				
BrowseName	BACnetFaultNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetNotificationType* defined in 9.3					
0:HasProperty	Variable	Reliability	BACnetReliability	0:PropertyType	O
0:HasProperty	Variable	Status_Flags	BACnetStatusFlags	0:PropertyType	O

The *BACnetFaultNotificationType ObjectType* is a concrete type and can be used directly.

The *ConditionType Property ConditionClass* set to *SystemConditionClass*.

9.4.2 ObjectType Description

9.4.2.1 Variable Reliability

This OPC UA Property, of *DataType BACnetReliability*, represents the BACnet property Reliability. The *BACnetReliability DataType* is defined in 10.4.28.

It provides an indication of whether the Present_Value or the operation of the physical I/O in question is "reliable" as far as the BACnet Device or operator can determine and, if not, why.

9.4.2.2 Variable Status_Flags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

It represents four *0:Boolean* flags that represent the general health. The flags are IN_ALARM, FAULT, OVERRIDDEN, and OUT_OF_SERVICE.

9.5 BACnetChangeOfReliabilityNotificationType

9.5.1 ObjectType definition

The *BACnetChangeOfReliabilityNotificationType* is formally defined in Table 76.

Table 76 – BACnetChangeOfReliabilityNotificationType Definition

Attribute	Value				
BrowseName	BACnetChangeOfReliabilityNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetFaultNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	PropertyValues	BaseDataType[ ]	0:PropertyType	M

9.5.2 ObjectType Description

9.5.2.1 Variable PropertyValues

Represents the sequence of BACnetPropertyValues.

9.6 BACnetEventNotificationType

9.6.1 ObjectType definition

The *BACnetEventNotificationType* is formally defined in Table 77.

Table 77 – BACnetEventNotificationType Definition

Attribute	Value				
BrowseName	BACnetEventNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetNotificationType* defined in 9.3					
0:HasProperty	Variable	Event_Values	BaseDataType[]	0:PropertyType	M

The *ConditionType Property ConditionClass* set to *ProcessConditionClass*.

9.7 BACnetChangeOfBitStringNotificationType

9.7.1 ObjectType definition

The *BACnetChangeOfBitStringNotificationType* is formally defined in Table 78.

Table 78 – BACnetChangeOfBitStringNotificationType Definition

Attribute	Value				
BrowseName	BACnetChangeOfBitStringNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	ReferencedBitString	0:Boolean[]	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.7.2 ObjectType Description

9.7.2.1 Variable ReferencedBitString

The *Property ReferencedBitString* represents the values of the referenced bit string.

9.7.2.2 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.8 BACnetChangeOfStateNotificationType

9.8.1 ObjectType definition

The *BACnetChangeOfStateNotificationType* is formally defined in Table 79.

Table 79 – BACnetChangeOfStateNotificationType Definition

Attribute	Value				
BrowseName	BACnetChangeOfStateNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	NewState	BACnetPropertyStates	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.8.2 ObjectType Description

9.8.2.1 Variable NewState

The *Property NewState* represents the new state of the monitored value.

9.8.2.2 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.9 BACnetChangeOfValueNotificationType

9.9.1 ObjectType definition

The *BACnetChangeOfValueNotificationType* is formally defined in Table 80.

Table 80 – BACnetChangeOfValueNotificationType Definition

Attribute	Value				
BrowseName	BACnetChangeOfValueNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	NewValue	0:OptionSet	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.9.2 ObjectType Description

9.9.2.1 Variable NewValue

The Property NewValue represents the new value of the monitored value.

9.9.2.2 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.10 BACnetChangeOfRealValueNotificationType

9.10.1 ObjectType definition

The *BACnetChangeOfRealValueNotificationType* is formally defined in Table 81.

Table 81 – BACnetChangeOfRealValueNotificationType Definition

Attribute	Value				
BrowseName	BACnetChangeOfRealValueNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	NewValue	0:Float	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.10.2 ObjectType Description

9.10.2.1 Variable NewValue

The Property NewValue represents the new value of the monitored value.

9.10.2.2 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.11 BACnetCommandFailureNotificationType

9.11.1 ObjectType definition

The *BACnetCommandFailureNotificationType* is formally defined in Table 82.

Table 82 – BACnetCommandFailureNotificationType Definition

Attribute	Value				
BrowseName	BACnetCommandFailureNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	CommandValue	0:BaseDataType	0:PropertyType	M
0:HasProperty	Variable	FeedbackValue	0:BaseDataType	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.11.2 ObjectType Description

9.11.2.1 Variable CommandValue

The Property CommandValue represents the monitored command value.

9.11.2.2 Variable FeedbackValue

The Property FeedbackValue represents the feedback command value.

9.11.2.3 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.12 BACnetFloatingLimitNotificationType

9.12.1 ObjectType definition

The *BACnetFloatingLimitNotificationType* is formally defined in Table 83.

Table 83 – BACnetFloatingLimitNotificationType Definition

Attribute	Value				
BrowseName	BACnetFloatingLimitNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	ReferenceValue	0:Float	0:PropertyType	M
0:HasProperty	Variable	SetpointValue	0:Float	0:PropertyType	M
0:HasProperty	Variable	ErrorLimit	0:Float	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.12.2 ObjectType Description

9.12.2.1 Variable ReferenceValue

The Property ReferenceValue represents the monitored value.

9.12.2.2 Variable SetpointValue

The Property SetpointValue represents the setpoint value.

9.12.2.3 Variable ErrorLimit

The Property ErrorLimit represents the error limit. It is the LowDiffLimit if

a) the new state is LOW_LIMIT, or

b) CurrentState is LOW_LIMIT and the new state is NORMAL

It is the HighDiffLimit if

a) the new state is HIGH_LIMIT, or

b) CurrentState is HIGH_LIMIT and the new state is NORMAL

9.12.2.4 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.13 BACnetOutOfRangeNotificationType

9.13.1 ObjectType definition

The *BACnetOutOfRangeNotificationType* is formally defined in Table 84.

Table 84 – BACnetOutOfRangeNotificationType Definition

Attribute	Value				
BrowseName	BACnetOutOfRangeNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	ExceedingValue	0:Float	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:Float	0:PropertyType	M
0:HasProperty	Variable	ExceedingLimit	0:Float	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.13.2 ObjectType Description

9.13.2.1 Variable ExceedingValue

The Property ExceedingValue represents the monitored value.

9.13.2.2 Variable Deadband

The Property Deadband represents the deadband value.

9.13.2.3 Variable ExceedingLimit

The Property ExceedingLimit represents the exceeding limit.

It is the LowLimit if

a) the new state is LOW_LIMIT, or

b) CurrentState is LOW_LIMIT and the new state is NORMAL

It is the HighLimit if

a) the new state is HIGH_LIMIT, or

b) CurrentState is HIGH_LIMIT and the new state is NORMAL

9.13.2.4 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.14 BACnetBufferReadyNotificationType

9.14.1 ObjectType definition

The *BACnetBufferReadyNotificationType* is formally defined in Table 85.

Table 85 – BACnetBufferReadyNotificationType Definition

Attribute	Value				
BrowseName	BACnetBufferReadyNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	BufferProperty	BACnetDeviceObjectPropertyReference	0:PropertyType	M
0:HasProperty	Variable	PreviousNotification	0:UInt32	0:PropertyType	M
0:HasProperty	Variable	CurrentNotification	0:UInt32	0:PropertyType	M

9.14.2 ObjectType Description

9.14.2.1 Variable BufferProperty

The Property BufferProperty represents a reference to the log buffer.

9.14.2.2 Variable PreviousNotification

The Property PreviousNotification represents the count of the previous notification.

9.14.2.3 Variable CurrentNotification

The Property CurrentNotification represents the monitored value.

9.15 BACnetUnsignedRangeNotificationType

9.15.1 ObjectType definition

The *BACnetUnsignedRangeNotificationType* is formally defined in Table 86.

Table 86 – BACnetUnsignedRangeNotificationType Definition

Attribute	Value				
BrowseName	BACnetUnsignedRangeNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	ExceedingValue	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	ExceedingLimit	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.15.2 ObjectType Description

9.15.2.1 Variable ExceedingValue

The Property ExceedingValue represents the monitored value.

9.15.2.2 Variable ExceedingLimit

The Property ExceedingLimit represents the exceeding limit.

It is the LowLimit if

a) the new state is LOW_LIMIT, or

b) CurrentState is LOW_LIMIT and the new state is NORMAL

It is the HighLimit if

a) the new state is HIGH_LIMIT, or

b) CurrentState is HIGH_LIMIT and the new state is NORMAL

9.15.2.3 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.16 BACnetDoubleOutOfRangeNotificationType

9.16.1 ObjectType definition

The *BACnetDoubleOutOfRangeNotificationType* is formally defined in Table 87.

Table 87 – BACnetDoubleOutOfRangeNotificationType Definition

Attribute	Value				
BrowseName	BACnetDoubleOutOfRangeNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	ExceedingValue	0:Double	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:Double	0:PropertyType	M
0:HasProperty	Variable	ExceedingLimit	0:Double	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.16.2 ObjectType Description

9.16.2.1 Variable ExceedingValue

The Property ExceedingValue represents the monitored value.

9.16.2.2 Variable Deadband

The Property Deadband represents the deadband value.

9.16.2.3 Variable ExceedingLimit

The Property ExceedingLimit represents the exceeding limit.

It is the LowLimit if

a) the new state is LOW_LIMIT, or

b) CurrentState is LOW_LIMIT and the new state is NORMAL

It is the HighLimit if

a) the new state is HIGH_LIMIT, or

b) CurrentState is HIGH_LIMIT and the new state is NORMAL

9.16.2.4 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.17 BACnetSignedOutOfRangeNotificationType

9.17.1 ObjectType definition

The *BACnetSignedOutOfRangeNotificationType* is formally defined in Table 88.

Table 88 – BACnetSignedOutOfRangeNotificationType Definition

Attribute	Value				
BrowseName	BACnetSignedOutOfRangeNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	ExceedingValue	0:Integer	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:Integer	0:PropertyType	M
0:HasProperty	Variable	ExceedingLimit	0:Integer	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.17.2 ObjectType Description

9.17.2.1 Variable ExceedingValue

The Property ExceedingValue represents the monitored value.

9.17.2.2 Variable Deadband

The Property Deadband represents the deadband value.

9.17.2.3 Variable ExceedingLimit

The Property ExceedingLimit represents the exceeding limit.

It is the LowLimit if

a) the new state is LOW_LIMIT, or

b) CurrentState is LOW_LIMIT and the new state is NORMAL

It is the HighLimit if

a) the new state is HIGH_LIMIT, or

b) CurrentState is HIGH_LIMIT and the new state is NORMAL

9.17.2.4 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.18 BACnetUnsignedOutOfRangeNotificationType

9.18.1 ObjectType definition

The *BACnetUnsignedOutOfRangeNotificationType* is formally defined in Table 89.

Table 89 – BACnetUnsignedOutOfRangeNotificationType Definition

Attribute	Value				
BrowseName	BACnetUnsignedOutOfRangeNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	ExceedingValue	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	Deadband	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	ExceedingLimit	0:UInteger	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.18.2 ObjectType Description

9.18.2.1 Variable ExceedingValue

The Property ExceedingValue represents the monitored value.

9.18.2.2 Variable Deadband

The Property Deadband represents the deadband value.

9.18.2.3 Variable ExceedingLimit

The Property ExceedingLimit represents the exceeding limit.

It is the LowLimit if

a) the new state is LOW_LIMIT, or

b) CurrentState is LOW_LIMIT and the new state is NORMAL

It is the HighLimit if

a) the new state is HIGH_LIMIT, or

b) CurrentState is HIGH_LIMIT and the new state is NORMAL

9.18.2.4 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

9.19 BACnetChangeOfCharacterStringNotificationType

9.19.1 ObjectType definition

The *BACnetChangeOfCharacterStringNotificationType* is formally defined in Table 90.

Table 90 – BACnetChangeOfCharacterStringNotificationType Definition

Attribute	Value				
BrowseName	BACnetChangeOfCharacterStringNotificationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BACnetEventNotificationType* defined in 9.4.2.1					
0:HasProperty	Variable	ChangedValue	0:String	0:PropertyType	M
0:HasProperty	Variable	AlarmValue	0:String	0:PropertyType	M
0:HasProperty	Variable	StatusFlags	BACnetStatusFlags	0:PropertyType	M

9.19.2 ObjectType Description

9.19.2.1 Variable ChangedValue

The Property ChangedValue represents the monitored value.

9.19.2.2 Variable AlarmValue

The Property AlarmValue conveys the character string of pAlarmValues related to the event state transition reported:

(a) for transitions to OFFNORMAL, the character string of pAlarmValues that matches pMonitoredValue,

(b) for transitions to NORMAL, the character string of pAlarmValues that match pMonitoredValue at the time of the most recent transition to OFFNORMAL.

9.19.2.3 Variable StatusFlags

This OPC UA *Property*, of *DataType BACnetStatusFlags*, represents the BACnet property Status_Flags. The *BACnetStatusFlags DataType* is defined in 10.3.7.

10 Mapping of DataTypes

10.1 Primitive data types

The mapping of primitive data types is described in Table 91.

Table 91 – Mapping of primitive data types

BACnet	OPC UA
Null	Null
0:Boolean	0:Boolean
Unsigned Integer Abstract unlimited type	There are three different options for mapping the abstract BACnet data type to OPC UA. • The BACnet data type is mapped to one of the concrete OPC UA DataTypes Byte, UInt16, UInt32, UInt64 if a byte length is defined in BACnet. • The BACnet data type is mapped to the abstract OPC UA data type 0:UInteger if the DataType is used in a Variable definition • The BACnet data type is mapped to UInt64, UInt32 or a subtype of ByteString if used in Structure DataTypes.
Signed Integer	There are three different options for mapping the abstract BACnet data type to OPC UA. • The BACnet data type is mapped to one of the concrete OPC UA DataTypes SByte, Int16, Int32, Int64 if a byte length is defined in BACnet. • The BACnet data type is mapped to the abstract OPC UA data type Integer if the DataType is used in a Variable definition The BACnet datat type is mapped to Int64, Int32 or a subtype of ByteString if used in Structure DataTypes.
Real	0:Float
Double	0:Double
Octet String	0:ByteString
Character String	0:String
Bit String	OPC UA OptionSet See 10.3 for the definition of option set data types.
Enumerated	OPC UA Enumeration " -" in names replaced by "" For reserved/deprecated members: Gaps are used See 10.4 for the definition of enumerated data types.
Date	Structure BACnetDate
Time	Structure BACnetTime
BACnetObjectIdentifier	See 10.2.1

10.1.1 SEQUENCE with optional fields

- Currently Structures with special indication for optional fields

10.1.2 SEQUENCE OF

- Within structure definition field with IsArray = true

10.1.3 SEQUENCE with CHOICE

- new virtual structure: TYPE+"SubTypes" subtypes + sub structure definitions for choices

Example:

```
BACnetSpecialEvent::= SEQUENCE {
        period CHOICE {
                                calendarEntry           [0] BACnetCalendarEntry,
                                calendarReference       [1] BACnetObjectIdentifier
                        },
        listOfTimeValues        [2] SEQUENCE OF BACnetTimeValue,
        eventPriority           [3] Unsigned (1..16)
}
```

Will be mapped to:

OPC UA Structure: BACnetSpecialEventPeriod (IsUnion = true)
calendarEntry: BACnetCalendarEntry
calendarReference: BACnetObjectIdentifier

OPC UA structure: BACnetSpecialEvent
period : BACnetSpecialEventPeriod
listOfTimeValues : BACnetTimeValue (ValueRank = Array)
eventPriority : UInt32

10.1.4 BACnetARRAY[7] of TYPE

- DataType = TYPE (Structure)
- ValueRank = Array
- ArrayDimension[0] = 7

10.1.5 BACnetARRAY[N] of TYPE

- DataType = TYPE (Structure)
- ValueRank = Array

10.1.6 List Of of TYPE

- DataType = TYPE (Structure)
- ValueRank = Array
- Methods for Adding/Removing Elements

10.1.7 Any

- BaseDataType

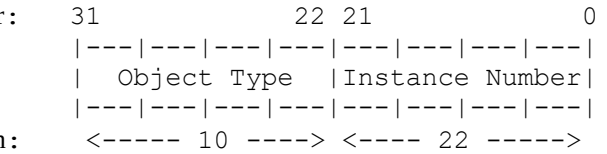
10.2 DataTypes derived from OPC UA Built-In types

10.2.1 BACnetObjectIdentifier

A BACnet Object Identifier value shall consist of two components:

(1) A 10-bit object type, representing the BACnetObjectType of the object, with bit 9 the most significant bit and bit 0 the least significant.

(2) A 22-bit object instance number, with bit 21 the most significant bit and bit 0 the least significant.



The encoding of an object identifier value shall be primitive, with four contents octets as follows:

Bits 9 through 2 of the object type shall be encoded in bits 7 through 0 of the first contents octet. Bits 1 through 0 of the object type shall be encoded in bits 7 through 6 of the second contents octet. Bits 21 through 16 of the object instance shall be encoded in bits 5 through 0 of the second contents octet. Bits 15 through 8 of the object instance shall be encoded in bits 7 through 0 of the third contents octet. Bits 7 through 0 of the object instance shall be encoded in bits 7 through 0 of the fourth contents octet.

Example: Application-tagged object identifier value

ASN.1 = ObjectIdentifier

Value = (Binary Input, 15)

Application Tag = ObjectIdentifier (Tag Number = 12)

Encoded Tag = X'C4'

Encoded Data = X'00C0000F'

The *BACnetObjectIdentifier* is a simple DataType and subtype of 0:UInt32. Its representation in the *AddressSpace* is defined in Table 92.

Table 92 – BACnetObjectIdentifier Definition

Attribute		Value			
BrowseName		BACnetObjectIdentifier			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:UInt32					

10.2.2 BACnetYear

This *DataType* identifies a BACnetYear. It is a subtype of *0:UInt16*. 0 = Undefined. Its representation in the *AddressSpace* is defined in Table 93.

Table 93 – BACnetYear Definition

Attribute		Value			
BrowseName		BACnetYear			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:UInt16					

10.3 OptionSet DataTypes used for BACnet bit strings

10.3.1 General

BACnet bit strings data types are mapped to OPC UA *Structure DataTypes* derived from the *OptionSet DataType* defined in OPC 10000-5.

The *OptionSet* structure contains the field value as ByteString and the field validBits as ByteString. The validBits field indicates which bits in the field value are valid.

All *DataTypes* defined in this clause are derived from the *OptionSet DataType* and expose the list of bit names defined in the tables as *0:PropertyType OptionSetValues* of the *DataType* node. The *OptionSetValues 0:PropertyType* provides an array of *LocalizedText*.

10.3.2 BACnetDaysOfWeek

This *DataType* is a subtype of 0:*OptionSet* representing the days of a week. Its values are defined in Table 94.

Table 94 – BACnetDaysOfWeek Values

Value	Bit No.	Description
monday	0	
tuesday	1	
wednesday	2	
thursday	3	
friday	4	
saturday	5	
sunday	6	

Its representation in the AddressSpace is defined in Table 95.

Table 95 – BACnetDaysOfWeek Definition

Attribute		Value			
BrowseName		BACnetDaysOfWeek			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the OptionSet DataType defined in OPC 10000-5					
0:HasProperty	Variable	0:OptionSetValues	0:LocalizedText []	0:PropertyType	

10.3.3 BACnetEventTransitionBits

This *DataType* is a subtype of 0:*OptionSet* representing BACnet event transition bits. Its values are defined in Table 96.

Table 96 – BACnetEventTransitionBits Values

Value	Bit No.	Description
to-offnormal	0	
to-fault	1	
to-normal	2	

Its representation in the AddressSpace is defined in Table 97.

Table 97 – BACnetEventTransitionBits Definition

Attribute		Value			
BrowseName		BACnetEventTransitionBits			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the OptionSet DataType defined in OPC 10000-5					
0:HasProperty	Variable	0:OptionSetValues	0:LocalizedText []	0:PropertyType	

10.3.4 BACnetLimitEnable

This *DataType* is a subtype of the 0:*OptionSet* representing if limits are enabled. Its values are defined in Table 98.

Table 98 – BACnetLimitEnable Values

Value	Bit No.	Description
lowLimitEnable	0	
highLimitEnable	1	

Its representation in the AddressSpace is defined in Table 99.

Table 99 – BACnetLimitEnable Definition

Attribute		Value			
BrowseName		BACnetLimitEnable			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the OptionSet DataType defined in OPC 10000-5					
0:HasProperty	Variable	0:OptionSetValues	0:LocalizedText []	0:PropertyType	

10.3.5 BACnetObjectTypeSupportedBits

This *DataType* is a subtype of 0:*OptionSet* and represents an option set that defines the object types that are supported by the BACnet Device. The *OptionSetValues* are defined in Table 100.

Table 100 – BACnetObjectTypeSupportedBits OptionSetValues

Name	Bit No.	Description
analog-input	0	
analog-output	1	
analog-value	2	
binary-input	3	
binary-output	4	
binary-value	5	
calendar	6	
command	7	
device	8	
event-enrollment	9	
file	10	
group	11	
loop	12	
multi-state-input	13	
multi-state-output	14	
notification-class	15	
program	16	
schedule	17	
averaging	18	
multi-state-value	19	
trend-log	20	
life-safety-point	21	
life-safety-zone	22	
accumulator	23	
pulse-converter	24	
event-log	25	
global-group	26	
trend-log-multiple	27	
load-control	28	
structured-view	29	
access-door	30	
UNASSIGNED_31	31	
access-credential	32	
access-point	33	
access-rights	34	
access-user	35	
access-zone	36	
credential-data-input	37	
network-security	38	
bitstring-value	39	
characterstring-value	40	
date-pattern-value	41	
date-value	42	
datetime-pattern-value	43	
datetime-value	44	
integer-value	45	
large-analog-value	46	
octetstring-value	47	
positive-integer-value	48	
time-pattern-value	49	
time-value	50	
notification-forwarder	51	
alert-enrollment	52	
channel	53	
lighting-output	54	

Its representation in the AddressSpace is defined in Table 101.

Table 101 – BACnetObjectTypeSupportedBits Definition

Attribute	Value				
BrowseName	BACnetObjectTypeSupportedBits				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the OptionSet DataType defined in OPC 10000-5					
0:HasProperty	Variable	0:OptionSetValues	0:LocalizedText []	0:PropertyType	

10.3.6 BACnetServicesSupportedBits

This *DataType* is a subtype of 0:*OptionSet* and represents an option set that defines the BACnet service set supported by the BACnet device. The *OptionsSetValues* are defined in Table 102.

Table 102 – BACnetServicesSupportedBits OptionSetValues

Name	Bit No.	Description
acknowledgeAlarm	0	
confirmedCOVNotification	1	
confirmedEventNotification	2	
getAlarmSummary	3	
getEnrollmentSummary	4	
subscribeCOV	5	
atomicReadFile	6	
atomicWriteFile	7	
addListElement	8	
removeListElement	9	
createObject	10	
deleteObject	11	
readProperty	12	
UNASSIGNED_13	13	
readPropertyMultiple	14	
writeProperty	15	
writePropertyMultiple	16	
deviceCommunicationControl	17	
confirmedPrivateTransfer	18	
reinitializeDevice	19	
vtOpen	20	
vtClose	21	
vtData	22	
UNASSIGNED_24	23	
UNASSIGNED_25	24	
i-Am	25	
i-Have	26	
unconfirmedCOVNotification	27	
unconfirmedEventNotification	28	
unconfirmedPrivateTransfer	29	
unconfirmedTextMessage	30	
timeSynchronization	31	
who-Has	32	
who-Is	33	
readRange	34	
utcTimeSynchronization	35	
lifeSafetyOperation	36	
subscribeCOVProperty	37	
getEventInformation	38	
writeGroup	39	

Its representation in the AddressSpace is defined in Table 103.

Table 103 – BACnetServicesSupportedBits Definition

Attribute		Value			
BrowseName		BACnetServicesSupportedBits			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the OptionSet DataType defined in OPC 10000-5					
0:HasProperty	Variable	0:OptionSetValues	0:LocalizedText []	0:PropertyType	

10.3.7 BACnetStatusFlags

This *DataType* is a subtype of 0:*OptionSet* and represents an option set that defines the general health of the device. Three of the values are flags that are associated with values of other properties of the object. Read these values to obtain more specific error information. See the BACnet specification for more information. The *OptionSetValues* are defined in Table 104.

Table 104 – BACnetStatusFlags OptionSetValues

Name	Bit No.	Description
InAlarm	0	
Fault	1	
Overriden	2	
OutOfService	3	

Its representation in the AddressSpace is defined in Table 105.

Table 105 – BACnetStatusFlags Definition

Attribute		Value			
BrowseName		BACnetStatusFlags			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the OptionSet DataType defined in OPC 10000-5					
0:HasProperty	Variable	0:OptionSetValues	0:LocalizedText []	0:PropertyType	

10.4 Enumeration DataTypes

10.4.1 General

BACnet ENUMERATED definitions are mapped to OPC UA *Enumeration DataTypes*. All *DataTypes* in this clause derived from the *Enumeration DataType* defined in in OPC 10000-3.

10.4.2 BACnetAction

This *DataType* is an enumeration that indicates that a loop is acting in either the DIRECT or REVERSE direction. Its values are defined in Table 106.

Table 106 – BACnetAction Values

Name	Value	Description
direct	0	
reverse	1	

Its representation in the AddressSpace is defined in Table 107.

Table 107 – BACnetAction Definition

Attribute	Value				
BrowseName	BACnetAction				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

10.4.3 BACnetBackupState

This *DataType* is an enumeration that represents the backup state. Its values are defined in Table 108.

Table 108 – BACnetBackupState Values

Name	Value	Description
Idle	0	
Preparing_For_Backup	1	
Preparing_For_Restore	2	
Performing_A_Backup	3	
Performing_A_Restore	4	
Backup_Failure	5	
Restore_Failure	6	

Its representation in the AddressSpace is defined in Table 109.

Table 109 – BACnetBackupState Definition

Attribute	Value				
BrowseName	BACnetBackupState				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

10.4.4 BACnenBinaryPV

This *DataType* is an enumeration that expresses date ranges within a month. For example, “last 7 days of this month”, “days numbered 8-14”, and so on. Its values are defined in Table 110.

Table 110 – BACnetBinaryPV Values

Name	Value	Description
Inactive	0	
Active	1	

Its representation in the AddressSpace is defined in Table 111.

Table 111 – BACnetBinaryPV Definition

Attribute	Value				
BrowseName	BACnetBinaryPV				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

10.4.5 BACnetDay

This *DataType* is an enumeration that expresses date ranges within a month. For example, “last 7 days of this month”, “days numbered 8-14”, and so on. Its values are defined in Table 112.

Table 112 – BACnetDay Values

Name	Value	Description
days numbered 1-7	1	
days numbered 8-14	2	
days numbered 15-21	3	
days numbered 22-28	4	
days numbered 29-31	5	
last 7 days of this month	6	
any week of this month	255	

Its representation in the AddressSpace is defined in Table 113.

Table 113 – BACnetDay Definition

Attribute	Value				
BrowseName	BACnetDay				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	

10.4.6 BACnetDayOfMonth

This *DataType* is an enumeration that indicates specific days of the month by specific date (“1”, “22”, and so on) or by relative position (“last day of month”, “even day of month”, and so on). Its values are defined in Table 114.

Table 114 – BACnetDayOfMonth Values

Name	Value	Description
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	
11	11	
12	12	
13	13	
14	14	
15	15	
16	16	
17	17	
18	18	
19	19	
20	20	
21	21	
22	22	
23	23	
24	24	
25	25	
26	26	
27	27	
28	28	
29	29	
30	30	
31	31	
Last day of month	32	
Odd day of month	33	

Even day of month	34	
Unspecified	255	

Its representation in the AddressSpace is defined in Table 115.

Table 115 – BACnetDayOfMonth Definition

Attribute		Value			
BrowseName		BACnetDayOfMonth			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	

10.4.7 BACnetDayOfWeek

This *DataType* is an enumeration that indicates each of the seven days of the week, or "unspecified". Its values are defined in Table 116.

Table 116 – BACnetDayOfWeek Values

Name	Value	Description
Monday	1	
Tuesday	2	
Wednesday	3	
Thursday	4	
Friday	5	
Saturday	6	
Sunday	7	
unspecified	255	

Its representation in the AddressSpace is defined in Table 117.

Table 117 – BACnetDayOfWeek Definition

Attribute		Value			
BrowseName		BACnetDayOfWeek			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	

10.4.8 BACnetDeviceCommunicationEnabled

This *DataType* is an enumeration that provides information if the BACnet communication is enabled. Its values are defined in Table 118.

Table 118 – BACnetDeviceCommunicationEnabled Values

Name	Value	Description
Enable	0	Enables the communication of the peer device.
Disable	1	Disables all communication by the peer device. Peer device is still listening to Reinitialize or DeviceCommunication services.
DisableInitiation	2	Disables only the communication actively initiated by the peer device, e.g. COV- or Event Notifications. Peer device still responds to services like ReadProperty, Write Property, etc.

Its representation in the AddressSpace is defined in Table 119.

Table 119 – BACnetDeviceCommunicationEnabled Definition

Attribute		Value			
BrowseName		BACnetDeviceCommunicationEnabled			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

10.4.9 BACnetDeviceStatus

This *DataType* is an enumeration that indicates the current physical and logical status of the BACnet Device. Its values are defined in Table 120. The meaning of these states (except for BackupInProgress_5) is local matters and is not defined.

Table 120 – BACnetDeviceStatus Values

Name	Value	Description
Operational	0	
OperationalReadOnly	1	
DownloadRequired	2	
DownloadInProgress	3	
NonOperational	4	
BackupInProgress	5	

Its representation in the AddressSpace is defined in Table 121.

Table 121 – BACnetDeviceStatus Definition

Attribute		Value			
BrowseName		BACnetDeviceStatus			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.10 BACnetEventState

This *DataType* is an enumeration that represents the BACnet event state. Its values are defined in Table 122.

Table 122 – BACnetEventState Values

Name	Value	Description
Normal	0	
Fault	1	
OffNormal	2	
HighLimit	3	
LowLimit	4	
LifeSafetyAlarm	5	

Its representation in the AddressSpace is defined in Table 123.

Table 123 – BACnetEventState Definition

Attribute		Value			
BrowseName		BACnetEventState			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.11 BACnetEventEnumType

This *DataType* is an enumeration that represents the BACnet event type. Its values are defined in Table 124.

Table 124 – BACnetEventEnumType Values

Name	Value	Description
ChangeOfBitstring	0	
ChangeOfState	1	
ChangeOfValue	2	
CommandFailure	3	
FloatingLimit	4	
OutOfRange	5	
ChangeOfLifeSafety	8	
Extended	9	
BufferReady	10	
UnsignedRange	11	

Its representation in the AddressSpace is defined in Table 125.

Table 125 – BACnetEventEnumType Definition

Attribute	Value				
BrowseName	BACnetEventEnumType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumValues	0: EnumValueType[]	0:PropertyType	

10.4.12 BACnetEventType

This *DataType* is an enumeration that represents the BACnet event type. Its values are defined in Table 126.

Table 126 – BACnetEventType Values

Name	Value	Description
change-of-bitstring	0	
change-of-state	1	
change-of-value	2	
command-failure	3	
out-of-range	5	
change-of-life-safety	8	
floating-limit	4	
extended	9	
buffer-ready	10	
unsigned-range	11	
access-event	13	
double-out-of-range	14	
signed-out-of-range	15	
unsigned-out-of-range	16	
change-of-characterstring	17	
change-of-status-flags	18	

Its representation in the AddressSpace is defined in Table 127.

Table 127 – BACnetEventType Definition

Attribute		Value			
BrowseName		BACnetEventType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumValues	0: EnumValueType[]	0:PropertyType	

10.4.13 BACnetFaultType

This *DataType* is an enumeration that indicates the BACnet fault type. Its values are defined in Table 128.

Table 128 – BACnetFaultType Values

Name	Value	Description
none	0	
fault-characterstring	1	
fault-exended	2	
fault-life-safety	3	
fault-state	4	
fault-status-flags	5	

Its representation in the AddressSpace is defined in Table 129.

Table 129 – BACnetFaultType Definition

Attribute		Value			
BrowseName		BACnetFaultType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.14 BACnetLifeSafetyMode

This *DataType* is an enumeration that represents the BACnet logging type. Its values are defined in Table 130.

Table 130 – BACnetLifeSafetyMode Values

Name	Value	Description
Off	0	
On	1	
Test	2	
Manned	3	
UnManned	4	
Armed	5	
Disarmed	6	
Prearmed	7	
Slow	8	
Fast	9	
Disconnected	10	
Enabled	11	
Disabled	12	
AutomaticReleaseDisabled	13	
Default	14	

Its representation in the AddressSpace is defined in Table 131.

Table 131 – BACnetLifeSafetyMode Definition

Attribute		Value			
BrowseName		BACnetLifeSafetyMode			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.15 BACnetLifeSafetyOperation

This *DataType* is an enumeration that represents the BACnet logging type. Its values are defined in Table 130.

Table 132 – BACnetLifeSafetyOperation Values

Name	Value	Description
None	0	
Silence	1	
SilenceAudible	2	
SilenceVisible	3	
Reset	4	
ResetAlarm	5	
ResetFault	6	
Unsilence	7	
UnsilenceAudible	8	
UnsilenceVisible	9	

Its representation in the AddressSpace is defined in Table 131.

Table 133 – BACnetLifeSafetyOperation Definition

Attribute		Value			
BrowseName		BACnetLifeSafetyMode			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.16 BACnetLoggingType

This *DataType* is an enumeration that represents the BACnet logging type. Its values are defined in Table 134.

Table 134 – BACnetLoggingType Values

Name	Value	Description
Polled	0	
COV	1	
Triggered	2	

Its representation in the AddressSpace is defined in Table 135.

Table 135 – BACnetLoggingType Definition

Attribute		Value			
BrowseName		BACnetLoggingType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.17 BACnetMessagePriority

This *DataType* is an enumeration that represents the BACnet logging type. Its values are defined in Table 136.

Table 136 – BACnetMessagePriority Values

Name	Value	Description
normal	0	
urgent	1	

Its representation in the AddressSpace is defined in Table 137.

Table 137 – BACnetMessagePriority Definition

Attribute	Value				
BrowseName	BACnetMessagePriority				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.18 BACnetMonth

This *DataType* is an enumeration that indicates either a specific Julian calendar month, or a relative relationship (“odd”, “even”, and “unspecified”). Its values are defined in Table 138.

Table 138 – BACnetMonth Values

Name	Value	Description
January	1	
February	2	
March	3	
April	4	
May	5	
June	6	
July	7	
August	8	
September	9	
October	10	
November	11	
December	12	
Odd	13	
Even	14	
Unspecified	255	

Its representation in the AddressSpace is defined in Table 139.

Table 139 – BACnetMonth Definition

Attribute	Value				
BrowseName	BACnetMonth				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	

10.4.19 BACnetNodeType

This *DataType* is an enumeration that indicates either a specific Julian calendar month, or a relative relationship (“odd”, “even”, and “unspecified”). Its values are defined in Table 140.

Table 140 – BACnetNodeType Values

Name	Value	Description
UNKNOWN	0	Indicates that a value for Node_Type is not available or has not been configured at this time
SYSTEM	1	An entire mechanical system
NETWORK	2	A communications network
DEVICE	3	Contains a set of elements which collectively represents a BACnet device, a logical device, or a physical device
ORGANIZATIONAL	4	Business concepts such as departments or people
AREA	5	Geographical concept such as a campus, building, floor, etc.
EQUIPMENT	6	Single piece of equipment that may be a collection of "Points"
POINT	7	Contains a set of elements which collectively defines a single point of data, either a physical input or output of a control or monitoring device, or a software calculation or configuration setting
COLLECTION	8	A generic container used to group things together, such as a collection of references to all space temperatures in a building
PROPERTY	9	Defines a characteristic or parameter of the parent node
FUNCTIONAL	10	Single system component such as a control module or a logical component such as a function block
OTHER	11	Everything that does not fit into one of these broad categories

Its representation in the AddressSpace is defined in Table 141.

Table 141 – BACnetNodeType Definition

Attribute	Value				
BrowseName	BACnetNodeType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

10.4.20 BACnetNotifyType

This *DataType* is an enumeration that represents the BACnet notify type. Its values are defined in Table 142.

Table 142 – BACnetNotifyType Values

Name	Value	Description
Alarm	0	
Event	1	
AckNotification	2	

Its representation in the AddressSpace is defined in Table 143.

Table 143 – BACnetNotifyType Definition

Attribute	Value				
BrowseName	BACnetNotifyType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.21 BACnetObjectTypeEnum

This *DataType* is an enumeration that represents the BACnet object type. Its values are defined in Table 144.

Table 144 – BACnetObjectTypeEnum Values

Name	Value	Description
analog-input	0	This object type represents physical analog input information, e. g. a sensor value.
analog-output	1	This object type represents physical analog output information, e. g. a 0-10V output.
analog-value	2	This object type represents an analog value (virtual) information, e. g. a setpoint value.
binary-input	3	This object type represents a binary input information, e. g. the state of a lamp or fuse.
binary-output	4	This object type represents a binary output information, e. g. a switch.
binary-value	5	This object type represents a binary value (virtual) information, e. g. an error state.
calendar	6	This object type represents calendar (date-based) information.
command	7	This object type represents command (scene) information.
device	8	This object type represents the physical device. It provides information like the local clock, the vendor, model-name and more.
event-enrollment	9	This object type is used to apply event monitoring in addition to the intrinsic reporting, e. g. to implement warning limits.
file	10	This object type represents files, e. g. the current configuration or persistent data.
group	11	This object type represents a group of objects local to the device.
loop	12	This object type represents controls loops, e. g. a PI or PID loop.
multi-state-input	13	This object type represents a physical multistate input information, e. g. a local operating mode switch.
multi-state-output	14	This object type represents a physical multistate output information, e. g. an operating mode switch controlled by the PLC.
notification-class	15	This object type represents an alarm class to notify recipients.
program	16	This object type represents the PLC program.
schedule	17	This object type represents a schedule (time based) used to specify weekly and/or exception schedule actions.
averaging	18	This object type represents an averaging object which provides statistic information.
multi-state-value	19	This object type represents a multistate value (virtual) information, e. g. program parameter.
trend-log	20	This object type represents a trendlog object support a single channel.
life-safety-point	21	This object type represents initiating and indicating devices in fire, life safety and security applications.
life-safety-zone	22	This object type represents an arbitrary group of BACnet Life Safety Point and Life Safety Zone objects in fire, life safety and security applications.
accumulator	23	This object type represents accumulated (impulse) values.
pulse-converter	24	This object type represents a converted impulse information, e. g. energy consumption in kWh.
event-log	25	This object type represents an eventlog buffer, e. g. to store alarms locally.
global-group	26	This object type represents a group of objects in one or more devices.
trend-log-multiple	27	This object type represents a trendlog object supporting multiple channels.
load-control	28	This object type represents the externally visible characteristics of a mechanism for controlling load requirements.
structured-view	29	This object type represents a user-oriented object hierarchy.
access-door	30	This object type represents a door in access-control systems.
unassigned	31	n/a
access-credential	32	This object type represents credentials in access-control systems.
access-point	33	This object type represents an access point in access-control systems.
access-rights	34	This object type represents the access rights in access-control systems.
access-user	35	This object type represents the user information in access-control systems.
access-zone	36	This object type represents the zone in access-control systems.
credentional-data-input	37	This object type represents the credential input (e. g. a card-reader) in access-control systems.
network-security	38	removed n/a
bitstring-value	39	This object type represents a bitstring information.
characterstring-value	40	This object type represents a string information.
date-pattern-value	41	This object type represents a date pattern. The pattern value 255 can be used as a wildcard.
date-value	42	This object type represents a specific single date information (day, month, year-1900, day-of-week).
datetime-pattern-value	43	This object type represents a combination of date and time supporting patterns.
datetime-value	44	This object type represents a combination of a specific date and time.
integer-value	45	This object type represents a signed integer value.
large-analog-value	46	This object type represents a large analog (8 BYTE LREAL) value.
octetstring-value	47	This object type represents an octetstring (hexadecimal) information.

positive-integer-value	48	This object type represents a positive integer (UNSIGNED) value.
time-pattern-value	49	This object type represents a time value supporting patterns.
time-value	50	This object type represents a specific time (hour, minute, second, hundredth of seconds).
notification-forwarder	51	This object type represents the characteristics required for the re-distribution of event notifications.
alert-enrollment	52	This object type represents the information required for managing information alerts from a BACnet device.
channel	53	This object type represents a channel in lighting applications.
lighting-output	54	This object type represents a lighting device.

Its representation in the AddressSpace is defined in Table 145.

Table 145 – BACnetObjectTypeEnum Definition

Attribute		Value			
BrowseName		BACnetObjectTypeEnum			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.22 BACnetPolarity

This DataType is an enumeration that defines the relationship between the physical state of some value and its logical state. Its values are defined in Table 146.

Table 146 – BACnetPolarity Values

Name	Value	Description
Normal	0	
Reverse	1	

Its representation in the AddressSpace is defined in Table 147.

Table 147 – BACnetPolarity Definition

Attribute		Value			
BrowseName		BACnetPolarity			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.23 BACnetProgramError

This DataType is an enumeration that defines the relationship between the physical state of some value and its logical state. Its values are defined in Table 148.

Table 148 – BACnetProgramError Values

Name	Value	Description
Normal	0	
LoadFailed	1	
Internal	2	
Program	3	
Other	4	

Its representation in the AddressSpace is defined in Table 149.

Table 149 – BACnetProgramError Definition

Attribute	Value				
BrowseName	BACnetProgramError				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.24 BACnetProgramRequest

This DataType is an enumeration that defines the relationship between the physical state of some value and its logical state. Its values are defined in Table 150.

Table 150 – BACnetProgramRequest Values

Name	Value	Description
Ready	0	
Load	1	
Run	2	
Halt	3	
Restart	4	
Unload	5	

Its representation in the AddressSpace is defined in Table 151.

Table 151 – BACnetProgramRequest Definition

Attribute	Value				
BrowseName	BACnetProgramRequest				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.25 BACnetProgramStates

This DataType is an enumeration that defines the relationship between the physical state of some value and its logical state. Its values are defined in Table 152.

Table 152 – BACnetProgramStates Request

Name	Value	Description
Idle	0	
Loading	1	
Running	2	
Waiting	3	
Halted	4	
Unloading	5	

Its representation in the AddressSpace is defined in Table 153.

Table 153 – BACnetProgramStates Definition

Attribute	Value				
BrowseName	BACnetProgramStates				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.26 BACnetPropertyIdentifier

This DataType is an enumeration that defines identifiers for properties that may exist as part of an object, such as acknowledged transactions, door alarm state, and so forth. Its values are defined in Table 154.

Table 154 – BACnetPropertyIdentifier Values

Name	Value	Description
AckedTransitions	0	
AckRequired	1	
Action	2	
ActionText	3	
ActiveText	4	
ActiveVtSessions	5	
AlarmValue	6	
AlarmValues	7	
All	8	
AllWritesSuccessful	9	
ApduSegmentTimeout	10	
ApduTimeout	11	
ApplicationSoftwareVersion	12	
Archive	13	
Bias	14	
ChangeOfStateCount	15	
ChangeOfStateTime	16	
NotificationClass	17	
this property deleted	18	
ControlledVariableReference	19	
ControlledVariableUnits	20	
ControlledVariableValue	21	
CovIncrement	22	
DateList	23	
DaylightSavingsStatus	24	
Deadband	25	
DerivativeConstant	26	
DerivativeConstantUnits	27	
Description	28	
DescriptionOfHalt	29	
DeviceAddressBinding	30	
DeviceType	31	
EffectivePeriod	32	
ElapsedActiveTime	33	
ErrorLimit	34	
EventEnable	35	
EventState	36	
EventType	37	
ExceptionSchedule	38	
FaultValues	39	
FeedbackValue	40	
FileAccessMethod	41	
FileSize	42	
FileType	43	
FirmwareRevision	44	
HighLimit	45	
InactiveText	46	
InProcess	47	
InstanceOf	48	
IntegralConstant	49	
IntegralConstantUnits	50	
Removed In Version 1 Revision 4_51	51	
LimitEnable	52	
ListOfGroupMembers	53	
ListOfObjectPropertyReferences	54	
Unassigned_55	55	

LocalDate	56	
LocalTime	57	
Location	58	
LowLimit	59	
ManipulatedVariableReference	60	
MaximumOutput	61	
MaxApduLengthAccepted	62	
MaxInfoFrames	63	
MaxMaster	64	
MaxPresValue	65	
MinimumOffTime	66	
MinimumOnTime	67	
MinimumOutput	68	
MinPresValue	69	
ModelName	70	
ModificationDate	71	
NotifyType	72	
NumberOfApduRetries	73	
NumberOfStates	74	
ObjectIdentifier	75	
ObjectList	76	
ObjectName	77	
ObjectPropertyReference	78	
ObjectType	79	
Optional	80	
OutOfService	81	
OutputUnits	82	
EventParameters	83	
Polarity	84	
PresentValue	85	
Priority	86	
PriorityArray	87	
PriorityForWriting	88	
ProcessIdentifier	89	
ProgramChange	90	
ProgramLocation	91	
ProgramState	92	
ProportionalConstant	93	
ProportionalConstantUnits	94	
Removed In Version 1 Revision 2_95	95	
ProtocolObjectTypesSupported	96	
ProtocolServicesSupported	97	
ProtocolVersion	98	
ReadOnly	99	
ReasonForHalt	100	
Removed In Version 1 Revision 4_101	101	
RecipientList	102	
Reliability	103	
RelinquishDefault	104	
Required	105	
Resolution	106	
SegmentationSupported	107	
Setpoint	108	
SetpointReference	109	
StateText	110	
StatusFlags	111	
SystemStatus	112	
TimeDelay	113	
TimeOfActiveTimeReset	114	
TimeOfStateCountReset	115	
TimeSynchronizationRecipients	116	
Units	117	
UpdateInterval	118	
UtcOffset	119	
VendorIdentifier	120	
VendorName	121	

VtClassesSupported	122	
WeeklySchedule	123	
AttemptedSamples	124	
AverageValue	125	
BufferSize	126	
ClientCovIncrement	127	
CovResubscriptionInterval	128	
Removed In Version 1 Revision 3_129	129	
EventTimeStamps	130	
LogBuffer	131	
LogDeviceObjectProperty	132	
Enable	133	
LogInterval	134	
MaximumValue	135	
MinimumValue	136	
NotificationThreshold	137	
Removed In Version 1 Revision 3_138	138	
ProtocolRevision	139	
RecordsSinceNotification	140	
RecordCount	141	
StartTime	142	
StopTime	143	
StopWhenFull	144	
TotalRecordCount	145	
ValidSamples	146	
WindowInterval	147	
WindowSamples	148	
MaximumValueTimestamp	149	
MinimumValueTimestamp	150	
VarianceValue	151	
ActiveCovSubscriptions	152	
BackupFailureTimeout	153	
ConfigurationFiles	154	
DatabaseRevision	155	
DirectReading	156	
LastRestoreTime	157	
MaintenanceRequired	158	
MemberOf	159	
Mode	160	
OperationExpected	161	
Setting	162	
Silenced	163	
TrackingValue	164	
ZoneMembers	165	
LifeSafetyAlarmValues	166	
MaxSegmentsAccepted	167	
ProfileName	168	
AutoSlaveDiscovery	169	
ManualSlaveAddressBinding	170	
SlaveAddressBinding	171	
SlaveProxyEnable	172	
LastNotifyRecord	173	
ScheduleDefault	174	
AcceptedModes	175	
AdjustValue	176	
Count	177	
CountBeforeChange	178	
CountChangeTime	179	
CovPeriod	180	
InputReference	181	
LimitMonitoringInterval	182	
LoggingObject	183	
LoggingRecord	184	
Prescale	185	
PulseRate	186	
Scale	187	

ScaleFactor	188	
UpdateTime	189	
ValueBeforeChange	190	
ValueSet	191	
ValueChangeTime	192	
AlignIntervals	193	
Unassigned_194	194	
IntervalOffset	195	
LastRestartReason	196	
LoggingType	197	
Unassigned_198	198	
Unassigned_199	199	
Unassigned_200	200	
Unassigned_201	201	
RestartNotificationRecipients	202	
TimeOfDeviceRestart	203	
TimeSynchronizationInterval	204	
Trigger	205	
UtcTimeSynchronizationRecipients	206	
NodeSubtype	207	
NodeType	208	
StructuredObjectList	209	
SubordinateAnnotations	210	
SubordinateList	211	
ActualShedLevel	212	
DutyWindow	213	
ExpectedShedLevel	214	
FullDutyBaseline	215	
Unassigned_216	216	
Unassigned_217	217	
RequestedShedLevel	218	
ShedDuration	219	
ShedLevelDescriptions	220	
ShedLevels	221	
StateDescription	222	
Unassigned_223	223	
Unassigned_224	224	
Unassigned_225	225	
DoorAlarmState	226	
DoorExtendedPulseTime	227	
DoorMembers	228	
DoorOpenTooLongTime	229	
DoorPulseTime	230	
DoorStatus	231	
DoorUnlockDelayTime	232	
LockStatus	233	
MaskedAlarmValues	234	
SecuredStatus	235	
Unassigned_236	236	
Unassigned_237	237	
Unassigned_238	238	
Unassigned_239	239	
Unassigned_240	240	
Unassigned_241	241	
Unassigned_242	242	
Unassigned_243	243	
AbsenteeLimit	244	
AccessAlarmEvents	245	
AccessDoors	246	
AccessEvent	247	
AccessEventAuthenticationFactor	248	
AccessEventCredential	249	
AccessEventTime	250	
AccessTransactionEvents	251	
Accompaniment	252	
AccompanimentTime	253	

ActivationTime	254	
ActiveAuthenticationPolicy	255	
AssignedAccessRights	256	
AuthenticationFactors	257	
AuthenticationPolicyList	258	
AuthenticationPolicyNames	259	
AuthenticationStatus	260	
AuthorizationMode	261	
BelongsTo	262	
CredentialDisable	263	
CredentialStatus	264	
Credentials	265	
CredentialsInZone	266	
DaysRemaining	267	
EntryPoints	268	
ExitPoints	269	
ExpiryTime	270	
ExtendedTimeEnable	271	
FailedAttemptEvents	272	
FailedAttempts	273	
FailedAttemptsTime	274	
LastAccessEvent	275	
LastAccessPoint	276	
LastCredentialAdded	277	
LastCredentialAddedTime	278	
LastCredentialRemoved	279	
LastCredentialRemovedTime	280	
LastUseTime	281	
Lockout	282	
LockoutRelinquishTime	283	
Removed In Version 1 Revision 13_284	284	
MaxFailedAttempts	285	
Members	286	
MusterPoint	287	
NegativeAccessRules	288	
NumberOfAuthenticationPolicies	289	
OccupancyCount	290	
OccupancyCountAdjust	291	
OccupancyCountEnable	292	
Removed In Version 1 Revision 13_293	293	
OccupancyLowerLimit	294	
OccupancyLowerLimitEnforced	295	
OccupancyState	296	
OccupancyUpperLimit	297	
OccupancyUpperLimitEnforced	298	
Removed In Version 1 Revision 13_299	299	
PassbackMode	300	
PassbackTimeout	301	
PositiveAccessRules	302	
ReasonForDisable	303	
SupportedFormats	304	
SupportedFormatClasses	305	
ThreatAuthority	306	
ThreatLevel	307	
TraceFlag	308	
TransactionNotificationClass	309	
UserExternalIdentifier	310	
UserInformationReference	311	
Unassigned_312	312	
Unassigned_313	313	
Unassigned_314	314	
Unassigned_315	315	
Unassigned_316	316	
UserName	317	
UserType	318	
UsesRemaining	319	

ZoneFrom	320	
ZoneTo	321	
AccessEventTag	322	
GlobalIdentifier	323	
Unassigned_324	324	
Unassigned_325	325	
VerificationTime	326	
BaseDeviceSecurityPolicy	327	
DistributionKeyRevision	328	
DoNotHide	329	
KeySets	330	
LastKeyServer	331	
NetworkAccessSecurityPolicies	332	
PacketReorderTime	333	
SecurityPduTimeout	334	
SecurityTimeWindow	335	
SupportedSecurityAlgorithms	336	
UpdateKeySetTimeout	337	
BackupAndRestoreState	338	
BackupPreparationTime	339	
RestoreCompletionTime	340	
RestorePreparationTime	341	
BitMask	342	
BitText	343	
IsUtc	344	
GroupMembers	345	
GroupMemberNames	346	
MemberStatusFlags	347	
RequestedUpdateInterval	348	
CovuPeriod	349	
CovuRecipients	350	
EventMessageTexts	351	
EventMessageTextsConfig	352	
EventDetectionEnable	353	
EventAlgorithmInhibit	354	
EventAlgorithmInhibitRef	355	
TimeDelayNormal	356	
ReliabilityEvaluationInhibit	357	
FaultParameters	358	
FaultType	359	
LocalForwardingOnly	360	
ProcessIdentifierFilter	361	
SubscribedRecipients	362	
PortFilter	363	
AuthorizationExemptions	364	
AllowGroupDelayInhibit	365	
ChannelNumber	366	
ControlGroups	367	
ExecutionDelay	368	
LastPriority	369	
WriteStatus	370	
PropertyList	371	
SerialNumber	372	
BlinkWarnEnable	373	
DefaultFadeTime	374	
DefaultRampRate	375	
DefaultStepIncrement	376	
EgressTime	377	
InProgress	378	
InstantaneousPower	379	
LightingCommand	380	
LightingCommandDefaultPriority	381	
MaxActualValue	382	
MinActualValue	383	
Power	384	
Transition	385	

EgressActive	386	

Its representation in the AddressSpace is defined in Table 155.

Table 155 – BACnetPropertyIdentifier Definition

Attribute		Value			
BrowseName		BACnetPropertyIdentifier			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.27 BACnetReinitializedStateofDevice

This *DataType* is an enumeration that represents the BACnet reinitialization state of a device. Its values are defined in Table 156.

Table 156 – BACnetReinitializedStateofDevice Values

Name	Value	Description
Coldstart	0	The precise interpretation of COLDSTART shall be defined by the vendor.
Warmstart	1	WARMSTART shall mean to reboot the device and start over, retaining all data and programs that would normally be retained during a brief power outage.
Startbackup	2	Starts a backup procedure.
Endbackup	3	Ends a backup procedure.
Startrestore	4	Starts a restore procedure.
Endrestore	5	Ends a restore procedure.
Abortrestore	6	Aborts a restore procedure.

Its representation in the AddressSpace is defined in Table 157.

Table 157 – BACnetReinitializedStateofDevice Definition

Attribute		Value			
BrowseName		BACnetReinitializedStateofDevice			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.28 BACnetReliability

This DataType is an enumeration that defines the various fault states that may be supported by an object. Its values are defined in Table 158.

Table 158 – BACnetReliability Values

Name	Value	Description
NoFaultDetected	0	The present value is reliable; that is, no other fault has been detected.
NoSensor	1	No sensor is connected to the Input object.
OverRange	2	The sensor connected to the Input is reading a value higher than the normal operating range. If the object is a Binary Input, this is possible when the Binary state is derived from an analog sensor or a binary input equipped with electrical loop supervision circuits.
UnderRange	3	The sensor connected to the Input is reading a value lower than the normal operating range. If the object is a Binary Input, this is possible when the Binary Input is actually a binary state calculated from an analog sensor.
OpenLoop	4	The connection between the defined object and the physical device is providing a value indicating an open circuit condition.
ShortedLoop	5	The connection between the defined object and the physical device is providing a value indicating a short circuit condition.
NoOutput	6	No physical device is connected to the Output object.
UnreliableOther	7	The controller has detected that the present value is unreliable, but none of the other conditions describe the nature of the problem. A generic fault other than those listed above has been detected, e.g., a Binary Input is not cycling as expected.
ProcessError	8	A processing error was encountered.
MultiStateFault	9	The FAULT_STATE, FAULT_LIFE_SAFETY or FAULT_CHARACTERSTRING fault algorithm has evaluated a fault condition.
ConfigurationError	10	The object's properties are not in a consistent state.
CommunicationFailure	12	Proper operation of the object is dependent on communication with a remote sensor or device and communication with the remote sensor or device has been lost.
MemberFault	13	Indicates that the set of referenced member objects includes one or more Status_Flags properties whose FAULT flag value is equal to TRUE.
MONITORED_OBJECT_FAULT	14	Indicates that the monitored object is in fault.
TRIPPED	15	The end device, such as an actuator, is not responding to commands, prevented by a tripped condition or by being mechanically held open.

Its representation in the AddressSpace is defined in Table 159.

Table 159 – BACnetReliability Definition

Attribute		Value			
BrowseName		BACnetReliability			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumValues	0: EnumValueType[]	0:PropertyType	

10.4.29 BACnetRestartReason

This *DataType* is an enumeration that represents a BACnet restart reason. Its values are defined in Table 160.

Table 160 – BACnetRestartReason Values

Name	Value	Description
unknown	0	The device cannot determine the cause of the last reset.
coldstart	1	A ReinitializeDevice request was received with a 'Reinitialized State of Device' of COLDSTART or the device was made to COLDSTART by some other means.
warmstart	2	A ReinitializeDevice request was received with a 'Reinitialized State of Device' of WARMSTART or the device was made to WARMSTART by some other means.
detected_power_lost	3	The device detected that incoming power was lost.
detected_powered_off	4	The device detected that its power switch was turned off.
hardware_watchdog	5	The hardware watchdog timer reset the device.
software_watchdog	6	The software watchdog timer reset the device.
suspended	7	The device was suspended. How the device was suspended or what it means to be suspended is a local matter.

Its representation in the AddressSpace is defined in Table 161.

Table 161 – BACnetRestartReason Definition

Attribute		Value			
BrowseName		BACnetRestartReason			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.4.30 BACnetSegmentation

This DataType is an enumeration that defines the segmentation of a transmission. Its values are defined in Table 162.

Table 162 – BACnetSegmentation Values

Name	Value	Description
segmented-both	0	
segmented-transmit	1	
segmented-receive	2	
no-segmentation	3	

Its representation in the AddressSpace is defined in Table 163.

Table 163 – BACnetSegmentation Definition

Attribute		Value			
BrowseName		BACnetSegmentation			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	

10.5 OPC UA Structure DataTypes

10.5.1 General

BACnet SEQUENCE definitions are mapped to OPC UA *Structure DataTypes*.

BACnet CHOICE definitions are mapped to OPC UA *Union DataTypes*. These types are defined in 10.6.

10.5.2 BACnetAddress

This DataType is a structure that represents a BACnet address. Its composition is formally defined in Table 164.

Table 164 – BACnetAddress Structure

Name	Type	Description
BACnetAddress	structure	
NetworkNumber	0:UInt16	A value 0 indicates the local network
MacAddress	0:ByteString	A string of length 0 indicates a broadcast

Its representation in the AddressSpace is defined in Table 165.

Table 165 – BACnetAddress Definition

Attribute	Value				
BrowseName	BACnetAddress				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.3 BACnetAddressBinding

This DataType is a structure that defines the network address binding of a BACnet object. Its composition is formally defined in Table 166.

Table 166 – BACnetAddressBinding Structure

Name	Type	Description
BACnetAddressBinding	structure	
DeviceObjectIdentifier	BACnetObjectIdentifier	The *DataType BACnetObjectIdentifier* is defined in 10.2.1.
DeviceAddress	BACnetAddress	The *DataType BACnetAddress* is defined in 10.5.2.

Its representation in the AddressSpace is defined in Table 167

Table 167 – BACnetAddressBinding Definition

Attribute	Value				
BrowseName	BACnetAddressBinding				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.4 BACnetCOVSubscription

This DataType is a structure that defines the change-of-value subscription for a BACnet object. Its composition is formally defined in Table 168.

Table 168 – BACnetCOVSubscription Structure

Name	Type	Description	Optional
BACnetCOVSubscription	structure		
Recipient	BACnetRecipientProcess	The *DataType BACnetRecipientProcess* is defined in 10.5.30.	False
MonitoredProperty Reference	BACnetDeviceObjectPropertyReference	The *DataType BACnetDeviceObjectPropertyReference* is defined in 10.5.10.	False
IssueConfirmed Notifications	0:Boolean		False
TimeRemaining	UInt32		False
CovIncrement	0:Float		True

Its representation in the AddressSpace is defined in Table 169.

Table 169 – BACnetCOVSubscription Definition

Attribute	Value				
BrowseName	BACnetCOVSubscription				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.5 BACnetDailySchedule

This DataType is a structure that defines a sequence of BACnetTimeValue structures. Each element in the sequence defines a time/value pair that describes the state of the object at a given point in the day. Its composition is formally defined in Table 170.

Table 170 – BACnetDailySchedule Structure

Name	Type	Description
BACnetDailySchedule	structure	
Day-schedule	BACnetTimeValue []	The *DataType BACnetTimeValue* is defined in 10.5.33

Its representation in the AddressSpace is defined in Table 171.

Table 171 – BACnetDailySchedule Definition

Attribute	Value				
BrowseName	BACnetDailySchedule				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.6 BACnetDate

This DataType is a structure that defines a calendar date. Its composition is formally defined in Table 172.

Table 172 – BACnetDate Structure

Name	Type	Description
BACnetDate	structure	
Year	BACnetYear	The *DataType BACnetYear* is defined in 10.2.2
Month	BACnetMonth	The *DataType BACnetMonth* is defined in 10.4.18
DayOfMonth	BACnetDayOfMonth	The *DataType BACnetDayOfMonth* is defined in 10.4.6
DayOfWeek	BACnetDayOfWeek	The *DataType BACnetDayOWeek* is defined in 10.4.7

Its representation in the AddressSpace is defined in Table 173.

Table 173 – BACnetDate Definition

Attribute	Value				
BrowseName	BACnetDate				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.7 BACnetDateRange

This DataType is a structure that defines a time span, with absolute start and end times. Its composition is formally defined in Table 174.

Table 174 – BACnetDateRange Structure

Name	Type	Description
BACnetDateRange	structure	
StartDate	BACnetDate	The *DataType BACnetDate* is defined in 10.5.6.
EndTime	BACnetDate	The *DataType BACnetDate* is defined in 10.5.6.

Its representation in the AddressSpace is defined in Table 175.

Table 175 – BACnetDateRange Definition

Attribute	Value				
BrowseName	BACnetDateRange				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.8 BACnetDateTime

This DataType is a structure that defines a calendar date and an absolute time. Its composition is formally defined in Table 176.

Table 176 – BACnetDateTime Structure

Name	Type	Description
BACnetDateTime	structure	
Date	BACnetDate	The *DataType BACnetDate* is defined in 10.5.6.
Time	BACnetTime	The *DataType BACnetDate* is defined in 10.5.32.

Its representation in the AddressSpace is defined in Table 177.

Table 177 – BACnetDateTime Definition

Attribute	Value				
BrowseName	BACnetDateTime				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.9 BACnetDestination

This DataType is a structure that represents a BACnet destination. Its composition is formally defined in Table 178.

Table 178 – BACnetDestination Structure

Name	Type	Description
BACnetDestination	structure	
ValidDays	BACnetDaysOfWeek	The *DataType BACnetDaysOfWeek* is defined in 10.3.2
FromTime	BACnetTime	The *DataType BACnetTime* is defined in 10.5.32
ToTime	BACnetTime	The *DataType BACnetTime* is defined in 10.5.32
Recipient	BACnetRecipient	The *DataType BACnetRecipient* is defined in 10.6.9
ProcessIdentifier	UInt32	
IssueConfirmedNotifications	0:Boolean	
Transitions	BACnetEventTransition Bits	The *DataType BACnetEventTransitionBits* is defined in 10.3.3

Its representation in the AddressSpace is defined in Table 179.

Table 179 – BACnetDestination Definition

Attribute	Value				
BrowseName	BACnetDestination				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.10 BACnetDeviceObjectPropertyReference

This OPC UA *DataType* also covers the superset of the BACnet data types BACnetDeviceObjectPropertyReference, BACnetDeviceObjectReference and

BACnetObjectPropertyReference. This is used to simplify the view from the OPC UA client side. It covers also changes in BACnet properties where the included type was changed to BACnetDeviceObjectPropertyReference.

This *DataType* is a structure that defines a reference to a 0:PropertyType of a BACnet object. Its composition is formally defined in Table 180.

Table 180 – BACnetDeviceObjectPropertyReference Structure

Name	Type	Description	Optional
BACnetDeviceObject PropertyReference	structure		
ObjectIdentifier	BACnetObjectIdentifier	Object properties that contain BACnetObjectIdentifiers may use 4194303 to indicate that the 0:PropertyType is not initialized.	False
PropertyIdentifier	BACnetPropertyIdentifier	The propertyIdentifier is not specified if the structure represents a BACnetDeviceObjectReference.	True
PropertyArrayIndex	UInt32	Used only with array datatype. If omitted with an array then the entire array is referenced.	True
DeviceIdentifier	BACnetObjectIdentifier	The deviceIdentifier is not specified if the structure represents a BACnetObjectPropertyReference.	True

Its representation in the AddressSpace is defined in Table 181.

Table 181 – BACnetDeviceObjectPropertyReference Definition

Attribute	Value				
BrowseName	BACnetDeviceObjectPropertyReference				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.11 BACnetEventParameterBufferReady

This DataType is a structure that represents a buffer ready event parameter. Its composition is formally defined in Table 182.

Table 182 – BACnetEventParameterBufferReady Structure

Name	Type	Description
BACnetEventParameterBufferReady	structure	
notification-threshold	0:UInt32	
Previous-notification-count	0:UInt32	

Its representation in the AddressSpace is defined in Table 183.

Table 183 – BACnetEventParameterBufferReady Definition

Attribute	Value				
BrowseName	BACnetEventParameterBufferReady				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.12 BACnetEventParameterChangeOfBitstring

This DataType is a structure that represents a change of bitstring event parameter. Its composition is formally defined in Table 184.

Table 184 – BACnetEventParameterChangeOfBitstring Structure

Name	Type	Description	Allow Subtypes
BACnetEventParameterChangeOfBitstring	structure		
Time-delay	0:UInt32		False
bitmask	0:OptionSet		True
List-of-bitstring-values	0:OptionSet[]		True

Its representation in the AddressSpace is defined in Table 185.

Table 185 – BACnetEventParameterChangeOfBitstring Definition

Attribute	Value				
BrowseName	BACnetEventParameterChangeOfBitstring				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.13 BACnetEventParameterChangeOfCharacterString

This DataType is a structure that represents a change of char string event parameter. Its composition is formally defined in Table 186.

Table 186 – BACnetEventParameterChangeOfCharacterString Structure

Name	Type	Description
BACnetEventParameterChangeOfCharacterString	structure	
Time-delay	0:UInt32	
AlarmValues	0:String[]	

Its representation in the AddressSpace is defined in Table 187.

Table 187 – BACnetEventParameterChangeOfCharacterString Definition

Attribute	Value				
BrowseName	BACnetEventParameterChangeOfCharacterString				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.14 BACnetEventParameterChangeOfLifeSafety

This DataType is a structure that represents a change of char string event parameter. Its composition is formally defined in Table 188.

Table 188 – BACnetEventParameterChangeOfLifeSafety Structure

Name	Type	Description
BACnetEventParameterChangeOfLifeSafety	structure	
NewState	BACnetLifeSafetyState	
NewMode	BACnetLifeSafetyMode	
OperationExtended	BACnetLifeSafetyOperation	

Its representation in the AddressSpace is defined in Table 189.

Table 189 – BACnetEventParameterChangeOfLifeSafety Definition

Attribute		Value			
BrowseName		BACnetEventParameterChangeOfLifeSafety			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.15 BACnetEventParameterChangeOfState

This DataType is a structure that represents a change of state event parameter. Its composition is formally defined in Table 190.

Table 190 – BACnetEventParameterChangeOfState Structure

Name	Type	Description
BACnetEventParameterChangeOfState	structure	
Time-delay	0:UInt32	
List-of -values	BACnetPropertyStates[]	

Its representation in the AddressSpace is defined in Table 191.

Table 191 – BACnetEventParameterChangeOfState Definition

Attribute		Value			
BrowseName		BACnetEventParameterChangeOfState			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.16 BACnetEventParameterChangeOfValue

This DataType is a structure that represents a change of value event parameter. Its composition is formally defined in Table 192.

Table 192 – BACnetEventParameterChangeOfValue Structure

Name	Type	Description	Allow Subtypes
BACnetEventParameterChangeOfValue	structure		
Time-delay	0:UInt32		False
Cov-criteria-bitmask	0:OptionSet		True
Cov-criteria-referenced-property-increment	0:Float		False

Its representation in the AddressSpace is defined in Table 193.

Table 193 – BACnetEventParameterChangeOfValue Definition

Attribute		Value			
BrowseName		BACnetEventParameterChangeOfValue			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.17 BACnetEventParameterCommandFailure

This DataType is a structure that represents a command failure event parameter. Its composition is formally defined in Table 194.

Table 194 – BACnetEventParameterCommandFailure Structure

Name	Type	Description
BACnetEventParameterCommandFailure	structure	
Time-delay	0:UInt32	
Feedback-property-reference	BACnetDeviceObjectPropertyReference	

Its representation in the AddressSpace is defined in Table 195.

Table 195 – BACnetEventParameterCommandFailure Definition

Attribute	Value				
BrowseName	BACnetEventParameterCommandFailure				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.18 BACnetEventParameteDoubleOutOfRange

This DataType is a structure that represents a double parameter out of range event parameter. Its composition is formally defined in Table 196.

Table 196 – BACnetEventParameterDoubleOutOfRange Structure

Name	Type	Description
BACnetEventParameterDoubleOutOfRange	structure	
Time-delay	0:UInt32	
Low-limit	0:Double	
High-limit	0:Double	
deadband	0:Double	

Its representation in the AddressSpace is defined in Table 197.

Table 197 – BACnetEventParameterDoubleOutOfRange Definition

Attribute	Value				
BrowseName	BACnetEventParameterDoubleOutOfRange				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.19 BACnetEventParameterFloatingLimit

This DataType is a structure that represents a floating limit event parameter. Its composition is formally defined in Table 198.

Table 198 – BACnetEventParameterFloatingLimit Structure

Name	Type	Description
BACnetEventParameterFloatingLimit	structure	
Time-delay	0:UInt32	
Setpoint-reference	BACnetDeviceObjectPropertyReference	
Low-diff-limit	0:Double	
High-diff-limit	0:Double	
deadband	0:Double	

Its representation in the AddressSpace is defined in Table 199.

Table 199 – BACnetEventParameterFloatingLimit Definition

Attribute		Value			
BrowseName		BACnetEventParameterFloatingLimit			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.20 BACnetEventParameterOutOfRange

This DataType is a structure that represents a parameter out of range event parameter. Its composition is formally defined in Table 200.

Table 200 – BACnetEventParameterOutOfRange Structure

Name	Type	Description
BACnetEventParameterOutOfRange	structure	
Time-delay	0:UInt32	
Low-limit	0:Double	
High-limit	0:Double	
Deadband	0:Double	

Its representation in the AddressSpace is defined in Table 201.

Table 201 – BACnetEventParameterOutOfRange Definition

Attribute		Value			
BrowseName		BACnetEventParameterOutOfRange			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.21 BACnetEventParameterSignedOutOfRange

This DataType is a structure that represents a signed parameter out of range event parameter. Its composition is formally defined in Table 202.

Table 202 – BACnetEventParameterSignedOutOfRange Structure

Name	Type	Description
BACnetEventParameterSignedOutOfRange	structure	
Time-delay	0:UInt32	
Low-limit	0:Int32	
High-limit	0:Int32	
deadband	0:UInt32	

Its representation in the AddressSpace is defined in Table 203.

Table 203 – BACnetEventParameterSignedOutOfRange Definition

Attribute		Value			
BrowseName		BACnetEventParameterSignedOutOfRange			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.22 BACnetEventParameterUnsignedOutOfRange

This DataType is a structure that represents an unsigned parameter out of range event parameter. Its composition is formally defined in Table 204.

Table 204 – BACnetEventParameterUnsignedOutOfRange Structure

Name	Type	Description
BACnetEventParameterUnsignedOutOfRange	structure	
Time-delay	0:UInt32	
Low-limit	0:UInt32	
High-limit	0:UInt32	
deadband	0:UInt32	

Its representation in the AddressSpace is defined in Table 205.

Table 205 – BACnetEventParameterUnsignedOutOfRange Definition

Attribute	Value				
BrowseName	BACnetEventParameterUnsignedOutOfRange				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.23 BACnetEventFaultParameterExtended

This DataType is a structure that represents a parameter extended event parameter. Its composition is formally defined in Table 206.

Table 206 – BACnetEventParameterExtended Structure

Name	Type	Description	Allow Subtypes
BACnetEventFaultParameterExtended	structure		
vendorId	0:UInt16		False
Extended-fault-type	0:UInteger		True
parameters	BACnetEventParameterExtendedParameters[]		False

Its representation in the AddressSpace is defined in Table 207.

Table 207 – BACnetEventFaultParameterExtended Definition

Attribute	Value				
BrowseName	BACnetEventFaultParameterExtended				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.24 BACnetEventParameterUnsignedRange

This DataType is a structure that represents an unsigned parameter out of range event parameter. Its composition is formally defined in Table 208.

Table 208 – BACnetEventParameterUnsignedRange Structure

Name	Type	Description
BACnetEventParameterUnsignedRange	structure	
Time-delay	0:UInt32	
Low-limit	0:UInt32	
High-limit	0:UInt32	

Its representation in the AddressSpace is defined in Table 209.

Table 209 – BACnetEventParameterUnsignedRange Definition

Attribute		Value			
BrowseName		BACnetEventParameterUnsignedRange			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.25 BACnetFaultParameterFaultCharacterstring

This DataType is a structure that represents char string fault parameter. Its composition is formally defined in Table 210.

Table 210 – BACnetFaultParameterFaultCharacterstring Structure

Name	Type	Description
BACnetFaultParameterFaultCharacterstring	structure	
Fault-characterstring	0:String	

Its representation in the AddressSpace is defined in Table 211.

Table 211 – BACnetFaultParameterFaultCharacterstring Definition

Attribute		Value			
BrowseName		BACnetFaultParameterFaultCharacterstring			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.26 BACnetFaultParameterFaultLifeSafety

This DataType is a structure that represents fault safety fault parameter. Its composition is formally defined in Table 212.

Table 212 – BACnetFaultParameterFaultLifeSafety Structure

Name	Type	Description
BACnetFaultParameterFaultLifeSafety	structure	
List-of-fault-values	BACnetLifeSafetyState[]	
Mode-property-reference	BACnetDeviceObjectPropertyReference	

Its representation in the AddressSpace is defined in Table 213.

Table 213 – BACnetFaultParameterFaultLifeSafety Definition

Attribute		Value			
BrowseName		BACnetFaultParameterFaultLifeSafety			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.27 BACnetFaultParameterFaultState

This DataType is a structure that represents fault safety fault parameter. Its composition is formally defined in Table 214.

Table 214 – BACnetFaultParameterFaultState Structure

Name	Type	Description
BACnetFaultParameterFaultState	structure	
List-of-fault-values	BACnetProgramStates[]	

Its representation in the AddressSpace is defined in Table 215.

Table 215 – BACnetFaultParameterFaultState Definition

Attribute		Value			
BrowseName		BACnetFaultParameterFaultState			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.28 BACnetFaultParameterFaultStatusFlags

This DataType is a structure that represents fault status flags fault parameter. Its composition is formally defined in Table 216.

Table 216 – BACnetFaultParameterFaultStatusFlags Structure

Name	Type	Description
BACnetFaultParameterFaultStatusFlags	structure	
Status-flags-reference	BACnetDeviceObjectPropertyReference []	

Its representation in the AddressSpace is defined in Table 217.

Table 217 – BACnetFaultParameterFaultStatusFlags Definition

Attribute		Value			
BrowseName		BACnetFaultParameterFaultStatusFlags			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.29 BACnetPropertyStates

This DataType is a structure that represents property states. Its composition is formally defined in Table 218.

Table 218 – BACnetPropertyStates Structure

Name	Type	Description
BACnetPropertyStates	structure	
BooleanValue	0:Boolean	
BinaryValue	BACnetBinaryPV	
EventType	BACnetEventEnumType	
Polarity	BACnetPolarity	
ProgramChange	BACnetProgramRequest	
ProgramState	BACnetProgramStates	
ProgramError	BACnetProgramError	
Reliability	BACnetReliability	
State	BACnetEventState	
SystemStatus	BACnetDeviceStatus	
Units	0:EUInformation	
UnsignedValue	0:UInt32	
LifeSafetyMode	BACnetLifeSafetyMode	
LifeSafetyState	BACnetLifeSafetyState	

Its representation in the AddressSpace is defined in Table 219.

Table 219 – BACnetPropertyStates Definition

Attribute	Value				
BrowseName	BACnetPropertyStates				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.30 BACnetRecipientProcess

This DataType is a structure that represents the recipient process. Its composition is formally defined in Table 220.

Table 220 – BACnetRecipientProcess Structure

Name	Type	Description
BACnetRecipientProcess	structure	
Recipient	BACnetRecipient	The *DataType BACnetRecipient* is defined in 10.6.9
ProcessIdentifier	UInt32	

Its representation in the AddressSpace is defined in Table 221.

Table 221 – BACnetRecipientProcess Definition

Attribute	Value				
BrowseName	BACnetRecipientProcess				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.31 BACnetSpecialEvent

This DataType is a structure that defines a period, a list of time values, and a priority. It is a means to identify moments in time over one or more days. Its composition is formally defined in Table 222.

Table 222 – BACnetSpecialEvent Structure

Name	Type	Description
BACnetSpecialEvent	structure	
Period	BACnetSpecialEventPeriod	The *DataType BACnetSpecialEventPeriod* is defined in
ListOfTimeValues	BACnetTimeValue[]	The *DataType BACnetTimeValue* is defined in 10.5.33
EventPriority	Byte	

Its representation in the AddressSpace is defined in Table 223.

Table 223 – BACnetSpecialEvent Definition

Attribute	Value				
BrowseName	BACnetSpecialEvent				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.32 BACnetTime

This DataType is a structure that represents a time. Its composition is formally defined in Table 224.

Table 224 – BACnetTime Structure

Name	Type	Description
BACnetTime	structure	
Hour	Byte	
Minute	Byte	
Second	Byte	
Hundredths	Byte	

Its representation in the AddressSpace is defined in Table 225.

Table 225 – BACnetTime Definition

Attribute	Value				
BrowseName	BACnetTime				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.33 BACnetTimeValue

This DataType is a structure that represents a BACnet time value. Its composition is formally defined in Table 226.

Table 226 – BACnetTimeValue Structure

Name	Type	Description
BACnetTimeValue	structure	
Time	BACnetTime	The *DataType BACnetTime* is defined in 10.5.32
Value	BACnetTimeValueValue	The *DataType BACnetTimeValueValue* is defined in 10.5.34

Its representation in the AddressSpace is defined in Table 227.

Table 227 – BACnetTimeValue Definition

Attribute	Value				
BrowseName	BACnetTimeValue				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.34 BACnetTimeValueValue

This DataType is a structure that represents a BACnet value used in BACnet time value. Its composition is formally defined in Table 228. If the BACnet value is NULL, the OPC UA value shall be set to NULL.

Table 228 – BACnetTimeValueValue Structure

Name	Type	Description	Allow Subtypes
BACnetTimeValueValue	structure		
BooleanValue	0:Boolean		False
UnsignedValue	0:UInteger		True
SignedValue	0:Integer		True
OctedStringValue	0:ByteString		False
CharStringValue	0:String		False
ObjectIdentifierValue	BACnetObjectIdentifier		False
EnumerationValue	0:Int32		False
BitStringValue	0:OptionSet		True

Its representation in the AddressSpace is defined in Table 229.

Table 229 – BACnetTimeValueValue Definition

Attribute		Value			
BrowseName		BACnetTimeValueValue			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.5.35 BACnetWeekNDay

This DataType is a structure that defines a combination of day and month using numeric codes. Its composition is formally defined in Table 230.

Table 230 – BACnetWeekNDay Structure

Name	Type	Description
BACnetWeekNDay	structure	
Month	BACnetMonth	The *DataType BACnetMonth* is defined in 10.4.18.
Day	BACnetDay	The *DataType BACnetDay* is defined in 10.4.5.
DayOfWeek	BACnetDayOfWeek	The *DataType BACnetDay* is defined in 10.4.7.

Its representation in the AddressSpace is defined in Table 231.

Table 231 – BACnetWeekNDay Definition

Attribute		Value			
BrowseName		BACnetWeekNDay			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:Structure					

10.6 OPC UA Union DataTypes

10.6.1 General

BACnet CHOICE definitions are mapped to OPC UA *Structure DataTypes* derived from the *Union DataType* defined in OPC 10000-3.

10.6.2 BACnetCalendarEntry

This DataType is a union that defines various calendar date values. Its composition is formally defined in Table 232.

Table 232 – BACnetCalendarEntry Union

Name	Type	Description
BACnetCalendarEntry	union	
Date	BACnetDate	The *DataType BACnetDate* is defined in 10.5.6.
DateRange	BACnetDateRange	The *DataType BACnetDateRange* is defined in 10.5.7.
WeekNDay	BACnetWeekNDay	The *DataType BACnetWeekNDay* is defined in 10.5.35.

Its representation in the AddressSpace is defined in Table 233.

Table 233 – BACnetCalendarEntry Definition

Attributes		Value			
BrowseName		BACnetCalendarEntry			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.3 BACnetClientCOV

This DataType is a union that represents the client COV. Its composition is formally defined in Table 234. If the Default-increment shall be used, the value is set to NULL.

Table 234 – BACnetClientCOV Structure

Name	Type	Description
BACnetClientCOV	union	
Real-increment	0:Float	

Its representation in the AddressSpace is defined in Table 235.

Table 235 – BACnetClientCOV Definition

Attributes			Value		
BrowseName			BACnetClientCOV		
IsAbstract			False		
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.4 BACnetEventParameter

This DataType is a union that represents the event parameter. Its composition is formally defined in Table 236. If the event parameter is None-event, the value is set to NULL. Event state transitions are also indicated if the value of the mode parameter changed since the last transition indicated. In this case, any time delays are overridden and the transition is indicated immediately.

Table 236 – BACnetEventParameter Structure

Name	Type	Description
BACnetEventParameter	union	
Change-of-bitstring	BACnetEventParameterChangeOfBitstring	The CHANGE_OF_BITSTRING event algorithm detects whether the monitored value of type BIT STRING equals a value that is listed as an alarm value, after applying a bitmask.
Change-of-state	BACnetEventParameterChangeOfState	The CHANGE_OF_STATE event algorithm detects whether the monitored value equals a value that is listed as an alarm value. The monitored value may be of any discrete or enumerated data type, including Boolean.
Change-of-value	BACnetEventParameterChangeOfValue	The CHANGE_OF_VALUE event algorithm, for monitored values of datatype REAL, detects whether the absolute value of the monitored value changes by an amount equal to or greater than a positive REAL increment.
Command-failure	BACnetEventParameterCommandFailure	The COMMAND_FAILURE event algorithm detects whether the monitored value and the feedback value disagree for a time period. It may be used, for example, to verify that a process change has occurred after writing a property.
Floating-limit	BACnetEventParameterFloatingLimit	The FLOATING_LIMIT event algorithm detects whether the monitored value exceeds a range defined by a setpoint, a high difference limit, a low difference limit and a deadband.
Out-of-range	BACnetEventParameterOutOfRange	The OUT_OF_RANGE event algorithm detects whether the monitored value exceeds a range defined by a high limit and a low limit. Each of these limits may be enabled or disabled. If disabled, the normal range has no higher limit or no lower limit. In order to reduce jitter of the resulting event state, a deadband is applied when the value is in the process of returning to the normal range.
Extended	BACnetEventFaultParameterExtended	The EXTENDED event algorithm detects event conditions based on a proprietary event algorithm. The proprietary event algorithm uses parameters and conditions defined by the vendor. The algorithm is identified by a vendor-specific event type that is in the scope of the vendor's vendor identification code. The algorithm may, at the vendor's discretion, indicate a new event state, a transition to the same event state, or no transition to the Event-State-Detection. The indicated new event states may be NORMAL, and any OffNormal event state. FAULT event state may not be indicated by this algorithm. For the

		purpose of proprietary evaluation of unreliability conditions that may result in FAULT event state, a FAULT_EXTENDED fault algorithm shall be used.
Buffer-ready	BACnetEventParameterBufferReady	The BUFFER_READY event algorithm detects whether a defined number of records have been added to a log buffer since start of operation or the previous notification, whichever is most recent.
Unsigned-range	BACnetEventParameterUnsignedRange	The UNSIGNED_RANGE event algorithm detects whether the monitored value exceeds a range defined by a high limit and a low limit.
Double-out-of-range	BACnetEventParameterDoubleOutOfRange	The DOUBLE_OUT_OF_RANGE event algorithm detects whether the monitored value exceeds a range defined by a high limit and a low limit. Each of these limits may be enabled or disabled. If disabled, the normal range has no lower limit or no higher limit respectively. In order to reduce jitter of the resulting event state, a deadband is applied when the value is in the process of returning to the normal range.
Signed-out-of-range	BACnetEventParameterSignedOutOfRange	The SIGNED_OUT_OF_RANGE event algorithm detects whether the monitored value exceeds a range defined by a high limit and a low limit. Each of these limits may be enabled or disabled. If disabled, the normal range has no lower limit or no higher limit respectively. In order to reduce jitter of the resulting event state, a deadband is applied when the value is in the process of returning to the normal range.
Unsigned-out-of-range	BACnetEventParameterUnsignedOutOfRange	The UNSIGNED_OUT_OF_RANGE event algorithm detects whether the monitored value exceeds a range defined by a high limit and a low limit. Each of these limits may be enabled or disabled. If disabled, the normal range has no lower limit or no higher limit respectively. In order to reduce jitter of the resulting event state, a deadband is applied when the value is in the process of returning to the normal range.
Change-of-characterstring	BACnetEventParameterChangeOfCharacterString	The CHANGE_OF_CHARACTERSTRING event algorithm detects whether the monitored value matches a character string that is listed as an alarm value. Alarm values are of type BACnetOptionalCharacterString, and may also be NULL or an empty character string.
Change-of-life-safety	BACnetEventParameterChangeOfLifeSafety	The CHANGE_OF_LIFE_SAFETY event algorithm detects whether the monitored value equals a value that is listed as an alarm value or life safety alarm value. Event state transitions are also indicated if the value of the mode parameter changed since the last transition indicated. In this case, any time delays are overridden and the transition is indicated immediately.

Its representation in the AddressSpace is defined in Table 237.

Table 237 – BACnetEventParameter Definition

Attributes		**Value**			
BrowseName		BACnetEventParameter			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.5 BACnetEventParameterExtendedParameters

This DataType is a union that represents the event parameter values. Its composition is formally defined in Table 238

Table 238 – BACnetEventParameterExtendedParameters Structure

Name	**Type**	**Description**	Allow Subtypes
BACnetEventParameterExtendedParameters	union		
real	0:Double		False
unsigned	0:UInt32		False
boolean	0:Boolean		False
double	0:Double		False
octed	0:Byte[]		False

characterString	0:String		False
bitString	0:OptionSet		True
enum	0:UInt32		False
date	BACnetDate		False
time	BACnetTime		False
objectIdentifier	BACnetObjectIdentifier		False
reference	BACnetDeviceObjectPropertyReference		False
integer	0:Int32		False

Its representation in the AddressSpace is defined in Table 239.

Table 239 – BACnetEventParameterExtendedParameters Definition

Attributes		**Value**			
BrowseName		BACnetEventParameterExtendedParameters			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.6 BACnetFaultParameter

This DataType is a union that represents fault parameter. Its composition is formally defined in Table 240. If the value is set to NULL, “NONE” is selected. The NONE fault algorithm is a placeholder for the case where no fault algorithm is applied by the object. This fault algorithm has no parameters, no conditions, and does not indicate any transitions of reliability.

Table 240 – BACnetFaultParameter Structure

Name	**Type**	**Description**
BACnetFaultParameter	union	
Fault-characterstring	BACnetFaultParameterFaultCharacterstring	The FAULT_CHRACTERSTRING event algorithm detects whether the monitored value matches a character string that is listed as a fault value. Fault values are of type BACnetOptionalCharacterString and may also be NULL or an empty character string.
Fault-life-safety	BACnetFaultParameterFaultLifeSafety	The FAULT_LIFE_SAFETY fault algorithm detects whether the monitored value equals a value that is listed as a fault value. The monitored value is of type BACnetLifeSafetyState. If internal operational reliability is unreliable, then the internal reliability takes precedence over evaluation of the monitored value.
Fault-state	BACnetFaultParameterFaultState	The FAULT_STATE fault algorithm detects whether the monitored value equals a value that is listed as a fault value. The monitored value may be of any discrete or enumerated data type, including Boolean. If internal operational reliability is unreliable, then the internal reliability takes precedence over evaluation of the monitored value.
Fault-status-flags	BACnetFaultParameterFaultStatusFlags	The FAULT_STATUS_FLAGS fault algorithm detects whether the monitored status flags are indicating a fault condition.
Fault-extended	BACnetEventFaultParameterExtended	The FAULT_EXTENDED fault algorithm detects fault conditions based on a proprietary fault algorithm. The proprietary fault algorithm uses parameters and conditions defined by the vendor. The algorithm is identified by a vendor-specific fault type that is in the scope of the vendor's vendor identification code. The algorithm may, at the vendor's discretion, indicate a new reliability, a transition to the same reliability, or no transition to the reliability-evaluation process.

Its representation in the AddressSpace is defined in Table 241.

Table 241 – BACnetFaultParameter Definition

Attributes		Value			
BrowseName		BACnetFaultParameter			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.7 BACnetMessageClass

This DataType is a union that defines a message class. Its composition is formally defined in Table 242.

Table 242 – BACnetMessageClass Union

Name	Type	Description	Allow Subtypes
BACnetMessageClass	union		
Unsigned	0:UInteger		True
String	0:String		False

Its representation in the AddressSpace is defined in Table 243.

Table 243 – BACnetMessageClass Definition

Attributes		Value			
BrowseName		BACnetMessageClass			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.8 BACnetPriorityValue

This DataType is a union that defines a context specific priority, where the context is based on data type (character string, 0:Boolean, and so on). Its composition is formally defined in Table 244. If there is no PriorityValue (None), the value is NULL.

Table 244 – BACnetPriorityValue Union

Name	Type	Description	Allow Subtypes
BACnetPriorityValue	union		
Real	0:Float		False
Enumerated	0:Int32		False
Unsigned	0:UInteger		True
Boolean	0:Boolean		False
Signed	0:Integer		True
Double	0:Double		False

Its representation in the AddressSpace is defined in Table 245.

Table 245 – BACnetPriorityValue Definition

Attributes		Value			
BrowseName		BACnetPriorityValue			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.9 BACnetRecipient

This DataType is a union that represents a recipient. Its composition is formally defined in Table 246.

Table 246 – BACnetRecipient Structure

Name	Type	Description
BACnetRecipient	union	
Device	BACnetObjectIdentifier	The *DataType BACnetObjectIdentifier* is defined in 10.2.1
Address	BACnetAddress	The *DataType BACnetAddress* is defined in 10.5.2

Its representation in the AddressSpace is defined in Table 247.

Table 247 – BACnetRecipient Definition

Attributes	Value				
BrowseName	BACnetRecipient				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.10 BACnetSpecialEventPeriod

This DataType is a union that represents a period for a special event. Its composition is formally defined in Table 248.

Table 248 – BACnetSpecialEventPeriod Structure

Name	Type	Description
BACnetSpecialEventPeriod	union	
CalendarEntry	BACnetCalendarEntry	The *DataType BACnetCalendarEntry* is defined in 10.6.2
CalendarReference	BACnetObjectIdentifier	The *DataType BACnetObjectIdentifier* is defined in 10.2.1

Its representation in the AddressSpace is defined in Table 249.

Table 249 – BACnetSpecialEventPeriod Definition

Attributes	Value				
BrowseName	BACnetSpecialEventPeriod				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

10.6.11 BACnetTimeStamp

This DataType is a union that represents a time stamp. Its composition is formally defined in Table 250.

Table 250 – BACnetTimeStamp Structure

Name	Type	Description
BACnetTimeStamp	union	
Time	BACnetTime	The *DataType BACnetTime* is defined in 10.5.32
SequenceNumber	0:UInt16	
DateTime	BACnetDateTime	The *DataType BACnetDateTime* is defined in 10.5.8

Its representation in the AddressSpace is defined in Table 251.

Table 251 – BACnetTimeStamp Definition

Attributes		Value			
BrowseName		BACnetTimeStamp			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of Union defined in OPC 10000-5.					

11 Mapping of Engineering Units

Mapping the engineering units used OPC UA is not quite easy. Table 252 shows those units which are identical in both standards and thus can be easily mapped.

Rows marked in yellow, not having an OPC UA UnitId, are handled differently. For those, an EngineeringUnits shall be used having “http://opcfoundation.org/UA/BACnet_V2/” as NamespaceUri, the “BACnetEngineeringUnits enumeration value” as UnitId, the OPC UA DisplayName of the table as DisplayName and the OPC UA Description of the table as Description.

Table 252 – Mapping of BACnet EngineeringUnits to OPC UA UnitIds

BACnetEngineeringUnits enumeration value	OPC UA UnitId	OPC UA DisplayName	OPC UA Description
0	5067851	m²	square metre
1	4609099	ft²	square foot
2	13387	mA	milliampere
3	4279632	A	ampere
4	5195853	Ω	ohm
5	5655636	V	volt
6	4937300	kV	kilovolt
7	4339512	MV	megavolt
8	4469814	V·A	volt - ampere
9	4937281	kV·A	kilovolt - ampere
10	5068353	MV·A	megavolt - ampere
11			volt-amperes-reactive
12	19253	kvar	kilovolt ampere (reactive)
13			megavolt-amperes-reactive
14			degrees-phase
15			power-factor
16	4869973	J	joule
17	4934223	kJ	kilojoule
18	5720146	W·h	watt hour
19	4937544	kW·h	kilowatt hour
20	4862777	Btu	British thermal unit (mean)
21	5125938	thm (US)	therm (U.S.)
22	13399	ton (US) /h	ton (US) per hour
23			joules-per-kilogram-dry-air
24			btus-per-pound-dry-air
25			cycles-per-hour
26			cycles-per-minute
27	4740186	Hz	hertz
28			grams-of-water-per-kilogram-dry-air
29			percent-relative-humidity
30	5066068	mm	millimetre
31	5067858	m	metre
32	4804168	in	inch
33	4607828	ft	foot
34			watts-per-square-foot
35			watts-per-square-meter
36	5002573	lm	lumen
37	5002584	lx	lux
38	5255735	ftc	footcandle
39	4933453	kg	kilogram
40			pounds-mass
41			tons
42	4933459	kg/s	kilogram per second
43	4600625	kg/min	kilogram per minute
44	4536627	kg/h	kilogram per hour

45			pounds-mass-per-minute
46			pounds-mass-per-hour
47	5723220	W	watt
48	4937556	kW	kilowatt
49	5062999	MW	megawatt
50	4863031	Btuth/h	British thermal unit (thermochemical) per hour
51			horsepower
52			tons-refrigeration
53	5259596	Pa	pascal
54	4935745	kPa	kilopascal
55	4342098	bar	bar [unit of pressure]
56	20563	lbf/in²	pound-force per square inch
57			centimeters-of-water
58	4601656	inH2O	inch of water
59			millimeters-of-mercury
60	4864057	cm Hg	centimetre of mercury
61	4601657	inHg	inch of mercury
62	4408652	°C	degree Celsius
63	4932940	K	kelvin
64	4604232	°F	degree Fahrenheit
65			degree-days-Celsius
66			degree-days-Fahrenheit
67			years
68	5066574	mo	month
69	5719365	wk	week
70	4473177	d	day
71	4740434	h	hour
72	5065038	min	minute [unit of time]
73	5457219	s	second [unit of time]
74	5067859	m/s	metre per second
75	4934984	km/h	kilometre per hour
76	18003	ft/s	foot per second
77	18002	ft/min	foot per minute
78	18509	mile/h	mile per hour (statute mile)
79	4609105	ft³	cubic foot
80	5067857	m³	cubic metre
81			imperial-gallons
82	5002322	l	litre
83	4672588	gal (US)	gallon (US)
84	12876	ft³/min	cubic foot per minute
85	5067091	m³/s	cubic metre per second
86	18227	gal (UK) /min	Imperial gallon per minute
87	4666673	l/s	litre per second
88	19506	l/min	litre per minute
89	18226	gal (US) /min	US gallon per minute
90	17476	°	degree [unit of angle]
91	4731186	°C/h	degree Celsius per hour
92	4731187	°C/min	degree Celsius per minute
93	4862515	°F/h	degree Fahrenheit per hour
94	4862516	°F/min	degree Fahrenheit per minute
95			no-units
96			parts-per-million
97			parts-per-billion
98	20529	%	percent
99			percent-per-second
100			per-minute
101			per-second
102			psi-per-degree-Fahrenheit
103			radians
104	5394509	r/min	revolutions per minute
105			currency1
106			currency2
107			currency3
108			currency4
109			currency5
110			currency6

111			currency7
112			currency8
113			currency9
114			currency10
115	4804171	in²	square inch
116	4410699	cm²	square centimetre
117	16730	BtuIT/lb	British thermal unit (international table) per pound
118	4410708	cm	centimetre
119			pounds-mass-per-second
120			delta-degrees-Fahrenheit
121			delta-degrees-Kelvin
122	4338745	kΩ	kiloohm
123	4339509	MΩ	megaohm
124	12890	mV	millivolt
125	4338738	kJ/kg	kilojoule per kilogram
126	13122	MJ	megajoule
127			joules-per-degree-Kelvin
128			joules-per-kilogram-degree-Kelvin
129	5064794	kHz	kilohertz
130	4933722	MHz	megahertz
131			per-hour
132	4404017	mW	milliwatt
133	4274487	hPa	hectopascal
134	5063250	mbar	millibar
135	5067080	m³/h	cubic metre per hour
136			liters-per-hour
137			kilowatt-hours-per-square-meter
138			kilowatt-hours-per-square-foot
139			megajoules-per-square-meter
140			megajoules-per-square-foot
141			watts-per-square-meter-degree-kelvin
142			cubic-feet-per-second
143			percent-obscuration-per-foot
144			percent-obscuration-per-meter
145	4535349	mΩ	milliohm
146	5068616	MW·h	megawatt hour (1000 kW.h)
147			kilo-btus
148			mega-btus
149			kilojoules-per-kilogram-dry-air
150			megajoules-per-kilogram-dry-air
151			kilojoules-per-degree-Kelvin
152			megajoules-per-degree-Kelvin
153	5129559	N	newton
154	4600377	g/s	gram per second
155	4600376	g/min	gram per minute
156	4534584	t/h	tonne per hour
157			kilo-btus-per-hour
158			hundredths-seconds
159	4403766	ms	millisecond
160	20053	N·m	newton metre
161	4403510	mm/s	millimetre per second
162	4732977	mm/min	millimetre per minute
163	12888	m/min	metre per minute
164	5060144	m/h	metre per hour
165	4666675	m³/min	cubic metre per minute
166	5067595	m/s²	metre per second squared
167	16709	A/m	ampere per metre
168	4273201	A/m²	ampere per square metre
169	16693	A·m²	ampere square metre
170	4604242	F	farad
171	14385	H	henry
172	4404785	Ω·m	ohm metre
173	5458245	S	siemens
174	4469040	S/m	siemens per metre
175	4469555	T	tesla
176	4469816	V/K	volt per kelvin

177	4470064	V/m	volt per metre
178	5719362	Wb	weber
179	4408396	cd	candela
180	4272692	cd/m²	candela per square metre
181	4600112	K/h	kelvin per hour
182	4600113	K/min	kelvin per minute
183	4337976	J·s	joule second
184	12865	rad/s	radian per second
185	4732217	m²/N	square metre per newton
186	4934993	kg/m³	kilogram per cubic metre
187	4404535	N·s	newton second
188	13392	N/m	newton per metre
189	4470067	W/(m·K)	watt per metre kelvin
190	4340025	µS	microsiemens
191	12875	ft³/h	cubic foot per hour
192			us-gallons-per-hour
193	4934996	km	kilometre
194	13384	µm	micrometre (micron)
195	4674125	g	gram
196	5064525	mg	milligram
197	5065812	ml	millilitre
198	13360	ml/s	millilitre per second
199	12878	dB	decibel
200			decibels-millivolt
201			decibels-volt
202	4403767	mS	millisiemens
203			watt-hours-reactive
204			kilowatt-hours-reactive
205			megawatt-hours-reactive
206			millimeters-of-water
207			per-mille
208			grams-per-gram
209	5059129	kg/kg	kilogram per kilogram
210			grams-per-kilogram
211	4732468	mg/g	milligram per gram
212	20033	mg/kg	milligram per kilogram
213	18250	g/ml	gram per millilitre
214	18252	g/l	gram per litre
215	19761	mg/l	milligram per litre
216	4731449	µg/l	microgram per litre
217	4274483	g/m³	gram per cubic metre
218	18256	mg/m³	milligram per cubic metre
219	18257	µg/m³	microgram per cubic metre
220			nanograms-per-cubic-meter
221	12851	g/cm³	gram per cubic centimetre
222	4346188	Bq	becquerel
223	12881	kBq	kilobecquerel
224	13390	MBq	megabecquerel
225	4274485	Gy	gray
226	4403507	mGy	milligray
227			microgray
228	4469043	Sv	sievert
229	4403768	mSv	millisievert
230			microsieverts
231	5257010	µSv/h	microsievert per hour
232			decibels-a
233			nephelometric-turbidity-unit
234	5321520	pH	pH (potential of Hydrogen)
235	18253	g/m²	gram per square metre
236			minutes-per-degree-kelvin

12 BACnet Profiles

The suggested minimum BACnet device profile supported by the mapping software implementing this specification is the B-OWS. On implementers option the B-AWS profile or other additional BIBBS may be supported.

The following tables contain the details of the BACnet client device profiles from ANSI/ASHRAE Standard 135-2012 annex L.

Table 253 – BACnet client device profiles

B-AWS	B-OWS
Data Sharing	
DS-RP-A	DS-RP-A
DS-RP-B	DS-RP-B
DS-RPM-A	DS-RPM-A
DS-WP-A	DS-WP-A
DS-WPM-A	DS-WPM-A
DS-AV-A	
DS-AM-A	
	DS-V-A
	DS-M-A
Alarm & Event Management	
AE-N-A	AE-N-A
AE-ACK-A	AE-ACK-A
AE-AS-A	AE-AS-A
	AE-VM-A
AE-AVM-A	
	AE-VN-A
AE-AVN-A	
AE-ELVM-A	
Scheduling	
	SCH-VM-A
SCH-AVM-A	
Trending	
	T-V-A
T-AVM-A	
Device & Network Management	
DM-DDB-A	DM-DDB-A
DM-DDB-B	DM-DDB-B
DM-ANM-A	
DM-ADM-A	
DM-DOB-B	DM-DOB-B
DM-DCC-A	
DM-MTS-A	DM-MTS-A
DM-OCD-A	
DM-RD-A	
DM-BR-A	

Table 254 – BIBBS List

Abbreviation	Description
DS-RP-A	Data Sharing - ReadProperty-A
DS-RP-B	Data Sharing - ReadProperty-B
DS-RPM-A	Data Sharing - ReadPropertyMultiple-A
DS-WP-A	Data Sharing - WriteProperty-A
DS-WPM-A	Data Sharing - WritePropertyMultiple-A
DS-AV-A	Data Sharing - Advanced View-A
DS-AM-A	Data Sharing - Advanced Modify-A
DS-V-A	Data Sharing - View-A
DS-M-A	Data Sharing - Modify-A
AE-N-A	Alarm and Event Management - Alarm and Event-Notification-A
AE-ACK-A	Alarm and Event Management - Alarm and Event-ACK-A
AE-AS-A	Alarm and Event Management - Alarm Summary View-A
AE-VM-A	Alarm and Event Management - View and Modify-A
AE-AVM-A	Alarm and Event Management - Advanced View and Modify-A

Abbreviation	Description
AE-VN-A	Alarm and Event Management - View Notifications-A
AE-AVN-A	Alarm and Event Management - Advanced View Notifications-A
AE-ELVM-A	Alarm and Event Management - Event Log View and Modify
SCH-VM-A	Scheduling - View Modify-A
SCH-AVM-A	Scheduling - Advanced View Modify-A
T-V-A	Trending - View-A
T-AVM-A	Trending - Advanced View and Modify-A
DM-DDB-A	Device Management - Dynamic Device Binding-A
DM-DDB-B	Device Management - Dynamic Device Binding-B
DM-ANM-A	Device Management - Automatic Network Mapping-A
DM-ADM-A	Device Management - Automatic Device Mapping-A
DM-DOB-B	Device Management - Dynamic Object Binding-B
DM-DCC-A	Device Management - DeviceCommunicationControl-A
DM-MTS-A	Device Management - Manual Time Synchronization-A
DM-OCD-A	Device Management - Object Creation and Deletion-A
DM-RD-A	Device Management - ReinitializeDevice-A
DM-BR-A	Device Management - Backup and Restore-A

13 Profiles and Conformance Units

This chapter defines the corresponding profiles and conformance units for the OPC UA Information Model for BACnet. Profiles are named groupings of conformance units. Facets are profiles that will be combined with other *Profiles* to define the complete functionality of an OPC UA *Server* or *Client.* The following tables specify the facets available for *Servers* that implement the BACnet Information Model companion standard.

13.1 Conformance Units

This chapter defines the corresponding *Conformance Units* for the OPC UA Information Model for BACnet.

Table 255 – BACnet Conformance Units Definition

Category	Title	Description
Server	BACnet Information Model	Supports all Nodes defined in this specification.
Server	BACmet mapping implementation	Provides all BACnet information of the accessed BACnet system as defined in this specification.

13.2 Profiles

13.2.1 Profile list

Table 256 lists all Profiles defined in this document and defines their URIs.

Table 256 – Profile URIs for BACnet

Profile	URI
BACnet Mapping Server Facet	http://opcfoundation.org/UA-Profile/BACnet/Server/BaseMapping

13.2.2 Server Facets

13.2.2.1 BACnet Mapping Server Facet

Table 257 defines a *Facet* of a server implementing the information model as defined in this specification and accesses a BACnet system.

Table 257 – BACnet Mapping Server Facet Definition

Group	Conformance Unit / Profile Title	Optional/ Mandatory
BACnet	BACnet Information Model	M
BACnet	BACmet mapping implementation	M

13.2.3 Client Facets

This specification does not define any client facets.

14 Namespaces

14.1 Namespace Metadata

Table 258 defines the namespace metadata for this document. The *Object* is used to provide version information for the namespace and an indication about static *Nodes*. Static *Nodes* are

identical for all *Attributes* in all *Servers*, including the *Value Attribute*. See OPC 10000-5 for more details.

The information is provided as *Object* of type *NamespaceMetadataType*. This *Object* is a component of the *Namespaces Object* that is part of the *Server Object*. The *NamespaceMetadataType ObjectType* and its *Properties* are defined in OPC 10000-5.

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in OPC 10000-6.

Table 258 – NamespaceMetadata Object for this Document

Attribute	**Value**	
BrowseName	http://opcfoundation.org/UA/BACnet_V2/	
Property	**DataType**	**Value**
NamespaceUri	String	http://opcfoundation.org/UA/BACnet_V2/
NamespaceVersion	String	2.00.1
NamespacePublicationDate	DateTime	2023-05-17
IsNamespaceSubset	0:Boolean	False
StaticNodeIdTypes	IdType []	0
StaticNumericNodeIdRange	NumericRange []	
StaticStringNodeIdPattern	String	

14.2 Handling of OPC UA Namespaces

Namespaces are used by OPC UA to create unique identifiers across different naming authorities. The *Attributes NodeId* and *BrowseName* are identifiers. A node in the UA *Address Space* is unambiguously identified using a *NodeId*. Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a node. Different nodes may have the same *BrowseName*. They are used to build a browse path between two nodes or to define a standard *Property*.

Servers may often choose to use the same namespace for the *NodeId* and the *BrowseName*. However, if they want to provide a standard *Property*, its *BrowseName* shall have the namespace of the standards body although the namespace of the *NodeId* reflects something else, for example the *EngineeringUnits Property*. All *NodeIds* of *Nodes* not defined in this specification shall not use the standard namespaces.

Table 259 provides a list of mandatory and optional namespaces used in a BACnet OPC UA *Server*.

Table 259 – Namespaces used in a BACnet Server

Namespace	**Description**
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowsNames* defined in the OPC UA specification. This namespace shall have namespace index 0.
Local Server URI	Namespace for nodes defined in the local server. This may include types and instances used in a device represented by the server. This namespace shall have namespace index 1.
http://opcfoundation.org/UA/BACnet_V2/	Namespace for *NodeIds* and *BrowsNames* defined in this specification. The namespace index is server specific.
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this companion specification in a vendor-specific namespace.
Vendor specific instances	A *Server* provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific namespace. It is recommended to separate vendor specific types and vendor specific instances into two or more namespaces.

Table 260 provides a list of namespaces and their index used for BrowseNames in this specification. The default namespace of this specification is not listed since all BrowseNames without prefix use this default namespace.

Table 260 – Namespaces used in this document

Namespace	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:EngineeringUnits

Annex A (normative): BACnet Namespace and Mappings

A.1 NodeSet and identifiers for BACnet Information Model

The BACnet *Information Model* is identified by the following URI:

http://opcfoundation.org/UA/BACnet_V2/

Documentation for the NamespaceUri can be found here.

The *NodeSet* associated with this version of specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/BACnet_V2/&v=2.00.1&i=1

The *NodeSet* associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/BACnet_V2/&i=1

Supplementary files for the BACnet *Information Model* can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/BACnet_V2/&v=2.00.1&i=2

The files associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/BACnet_V2/&i=2

Annex B (informative): BACnet Client Implementation

B.1 General

This annex is an informative best practice description of the behaviour of the *BACnetUaMapper* that implements this mapping document as an OPC UA server and BACnet client.

B.2 BACnet revisions

The minimum revision support required for the *BACnetUaMapper* is revision 7. The *BACnetUaMapper* should be prepared to communicate with BACnet servers supporting BACnet revision 4 or higher.

B.3 Timestamps and time synchronization

BACnet devices use local time for timestamps. OPC UA uses UTC time for timestamps. BACnetUaMapper should provide configuration options to handle the local time to UTC time conversion in the right way.

B.4 List handling

The methods defined in this specification for modification of BACnet list properties should be mapped to BACnet services AddListElement and RemoveListElement.

BACnet servers may not support the BACnet services AddListElement and RemoveListElement. An alternative method to update BACnet list elements could be:
Reading the entire list,
modify the content with the changes and
write back the list to the property.

B.5 Write with priority

The *BACnetUaMapper* should have a configuration option for the default priority used when a BACnet property is written through the OPC UA server. The default value for this configuration option should be the BACnet priority 8 which is “Manual Operator”.

B.6 Confirmation of confirmed event notifications

The *BACnetUaMapper* should automatically confirm (SimpleACK) the receipt of a confirmed event notification.