

So You're Going to Design a BACnet Device?

A short guide for new developers

Bill Swan

Probably every implementer new to BACnet, flipping through a copy of the BACnet standard for the first time, finds himself at a bit of a loss on how to proceed. Years ago, staring at this brand-new 500 page tome of technical legalese I held in my hands, I know I was. Flipping through thousands of numbered clauses describing layers, objects, properties, services, encodings, productions, procedures and more, and the huge list of acronyms including MS/TP, APDU, TSM, LLC, PICS, ASN.1, VT, BIBBs and so on, it was hard to see just how to get from there to an actual BACnet-speaking device on the workbench.

It is most likely that the new implementer will next do exactly what I did. Starting at Chapter One ("Purpose"), I read my grim and resolute way to the end ("References") and on beyond through the Annexes, with only brief flashes of understanding appearing through the clouds of confusion, soon to disappear again. Okay. Back to the beginning and try again...

This is *not* a good way to learn BACnet! There is a structure to the standard, a procedure for deciding what features to implement, guidelines to help avoid earlier implementers' mistakes, tools to help test implementations and multiple sources of information. This article presents information to help you get started faster.

The global view

The first step in understanding BACnet is to learn your way around the standard. In the BACnet for developer classes I've conducted I suggest reading the standard in this unintuitive order:

- Clause 12: Objects and properties (I/O and data models)
- Clauses 13 through 18: BACnet services (messages)
- Clauses 20 and 21: Encoding messages
- Clause 5: Transaction machines (guaranteed delivery across networks)
- Clause 6: Routing messages across networks
- Clauses 7 through 11: Datalinks (various network LANs)
- Annex J: Using UDP/IP as a datalink

This order presents the structure of the BACnet protocol stack starting at the top with modeling the device's I/O, data and operations; down to sending message packets across a datalink (LAN) to another device. One could implement a BACnet device on this information alone, but there are some aids to guide the implementer.

- Clause 19: Special procedures in BACnet (e.g. backup & restore)
- Clause 23: Proprietary extensions (implement features not in BACnet)
- Annexes D through F: Examples of objects and message encoding

But before you nail down the BACnet features of your device, you need to understand "device profiles" and the requirements they make on BACnet devices. If you don't do this first, you'll be doing it later – and probably changing your implementation as you go:

- Annexes L and K: Device profiles and BIBBs

No, those aren't backwards; it's better to read them in that order and you'll save yourself a bit of a headache in the process. But we'll return to this topic in a little more detail in the section "Defining your device."

There are a few more annexes that may be of help:

Annex C:	Formal descriptions of BACnet objects
Annex G:	How to perform BACnet's CRC calculations
Annex I:	How to implement "minimum on/off time" features in BACnet

Finally, be aware that BACnet is not a static standard. New capabilities and extensions are added from time to time, and are published on the ASHRAE website. These can be found at:

<http://www.ashrae.org/technology/page/132>

And because errors do creep in, corrections are posted at:

<http://www.ashrae.org/technology/page/120>

These are also posted on the BACnet website <http://www.bacnet.org/>.

BACnet Objects and Properties

BACnet objects represent various elements of building automation devices including inputs, outputs, data storage, control values, control loops, metering, scheduled operations or modes, program execution and data-logging ("trend-logging").

BACnet objects contain "properties," each of which represents a single data value, or in a few cases a group of values, related to the object's function. In many objects, particularly the I/O objects, there is a "central" property (usually called Present_Value) representing the central element of the object's function, such as the Present_Value property of an Analog Input representing a temperature read by a sensor. There are often subsidiary properties, such as the Units property of an Analog Input.

Properties defined in the standard fall into two basic categories: those required to be present because they represent the basic functionality of the object, and those defined as optional because they extend the capability of the object -- such as adding out-of-range alarms to a temperature sensor.

There are four components of every property of every BACnet object. These are:

1. The property's name (linked to a numeric code identifying the property in messages).
2. Its datatype, which is either "simple" (a.k.a. "primitive"), such as Integer or Character String, or "complex" (a.k.a. "constructed") data structures.
3. Its "conformance code" -- whether it is **R**equired, **O**ptional, or (required to be present and) **W**ritable.
4. A clause containing a text description of the property, its use, functionality and requirements.

Required properties, as noted, are *always* required to be present in the object (it is a violation of the standard to leave these properties out). I tend to group **R**equired properties into three categories:

1. The "identification group" (my term); these are the Object_Identifier, Object_Name and Object_Type properties, which are required in all BACnet objects.
2. The "basic functionality group": property or properties that provide the basic functionality of the object. In the case of the Calendar object there are two such properties: the Date_List property which holds a list of dates on which the Calendar is "active" (perhaps indicating holidays), and the Present_Value property which tells whether this Calendar objects is "active" today.

3. The "required functionality group": properties with additional functionality required by the standard. The Status_Flags property, for example, contains four binary flags to indicate whether the output is in an alarm state, in a fault condition, overridden by some process, or out of service.

Optional properties are either independently present or absent, or may be required to be present in groups. Independent optional properties may be related, such as Min_Pres_Value (minimum Present_Value), the lowest or most negative value that can appear on this particular output and Max_Pres_Value, but consider that an Analog Input object that presents a value ranging from say, one to infinity might contain a Min_Pres_Val property (value 1.0) but not a Max_Pres_Val.

Grouped optional properties typically provide some additional capability to the object, such as the capability to detect and issue alarms. These groups are identified by footnotes to the "Properties of the [Name] Object Type" property tables found in the Clause 12 object definitions.

Other optional properties are sometimes required by some other aspect of the implementation. Such conditions are also identified in the property tables.

More discussion of objects, properties and services is available in "The Language of BACnet," online at: <http://www.bacnet.org/Bibliography/ES-7-96/ES-7-96.htm> (The "Conformance Classes" and "Functional Groups" discussions in this 1996 article are now out of date and should be ignored.) Other articles posted at <http://www.bacnet.org/Bibliography/index.html> present introductions to other elements of BACnet, such as networking.

Formal definitions and examples of BACnet objects are presented in Annexes D and E of the standard.

BACnet Services and Networking

BACnet "services" are the communications or messages exchanged between BACnet devices. These are defined as either "Requests" or "Acknowledgements" ("ACKs" for short).

There are several kinds of services. Some are used to find particular devices or objects that reside somewhere on the BACnet network, for example, "Who is Device #54"? Other services are used to read or write the data values in properties. And yet others are used to issue, query about, and cancel alarms. The services are defined in categories as follows:

Alarm and Event	Clause 13
File Access	Clause 14
Object Access	Clause 15 (includes reading and writing properties)
Remote Device Management	Clause 16
Virtual Terminal	Clause 17
Network Security	Clause 24 (these are out of date and will be replaced)

The selection of services to be implemented, whether to be "initiated" (requests sent) or "executed" (requests received, a.k.a. "supported") is not completely arbitrary, as will be seen in **Defining your device**. Other requirements for BACnet devices have been added to the BACnet standard by Addendum *d-5* to BACnet-2004, available online at:

<http://www.bacnet.org/Addenda/Add-2004-135d.pdf>

Again, more discussion of services is available online at:

<http://www.bacnet.org/Bibliography/ES-7-96/ES-7-96.htm>

A formal explanation of how to encode BACnet services for transmission (in Application Protocol Data Units, or APDUs) is beyond the scope of this article, but examples of services and their corresponding encodings are provided in Annexes E and F. Two documents online may help further:

<http://www.bacnet.org/Tutorial/Encoding.doc>,

<http://www.bacnet.org/Bibliography/BACnet-Today-05/27060Ocraven.pdf>

Clause 6 provides a straightforward description of the Network layer's operation. Additionally an overview of BACnet networking is online at:

<http://www.bacnet.org/Bibliography/ES-1-97/ES-1-97.htm>

The Clause 5 Transaction State Machines (as well as the Clause 9 MS/TP and Clause 10 PTP machines) should be implemented *exactly* as presented. Shortcuts inevitably fail.

Getting Started

One of the first things you should do if you are going to develop a BACnet device is to get a BACnet Vendor ID! This is a unique number issued by ASHRAE to BACnet manufacturers, and will appear in your device. The process for obtaining one (it's free) is defined here, along with the list of those already assigned: <http://www.bacnet.org/VendorID/index.html>

Defining your device

As I noted earlier, one of the things you should do when initially defining your device is to choose its "device profile," because this is likely to affect the implementation.

The BACnet device profiles, defined in Annex L, define devices generally categorized as controllers or workstations, and usually ranked in capability or "power."¹ For example, controllers range from very simple (B-SS or BACnet Smart Sensor, a small I/O device) to very complex (B-BC, or BACnet Building Controller, a large device which often has no I/O but which monitors and controls other controllers).

At the end of Annex L is a table with column headings that are the device profiles, and in the columns below each device profile you'll find jumbles of letters separated by dashes. Those jumbles of letters, less cryptic than they appear initially, are known as BIBBs (BACnet Interoperability Building Blocks), and the appearance of a BIBB in a device profile column says that in order to claim that profile, the device is required to support that BIBB.

Each BIBB, defined in Annex K, specifies a functional area within BACnet and defines certain capabilities a device must have in order to claim it supports that BIBB. BIBBs are usually divided into two parts; for example, the DM-BR-A BIBB ("device management – backup and restore – A") defines certain requirements for a device (usually a workstation) that can backup and restore another device's configurations per a specified set of procedures in Clause 19.1, while the DM-BR-B BIBB defines the requirements for a device whose configuration can be backed up and restored by the Clause 19.1 procedures..

¹ As of 2007 only one workstation profile is defined, but a range of profiles is in currently in draft.

When selecting a device profile you generally want the "lowest" one that covers the capability of your device. Moving up to a "higher" profile in order to have a "better" specification may or may not be worth the additional expense of implementation and possibly hardware.

For example, if I were to design a BACnet light switch, the obvious choice is the "lowest" controller category, a BACnet Smart Sensor, whose only requirements are the DS-RP-B BIBB (other devices can read this device's property values, the property of interest here is obviously the one showing the light switch's position), plus the DM-DDB-B and DM-DOB-B BIBBs added in the aforementioned Addendum *d-5* (<http://www.bacnet.org/Addenda/Add-2004-135d.pdf>).

But I might also prefer to support the DS-COV-B ("data sharing – change of value – B") which would allow my BACnet light switch device to immediately inform other devices when the light switch was flipped on or off. This is acceptable – a device can claim additional BIBBs.

Optionality is the Enemy of Interoperability, unless you follow the rules

One of the strengths of BACnet lies in the flexibility it provides the implementer. Unfortunately, that very flexibility can also be its weakness. To illustrate, during a recent committee meeting I remarked to a new liaison that his PowerPoint was "sparse" by comparison to his predecessor's (which were spectacularly overloaded *objets d'art*). He didn't understand at all until I clarified: "more megabytes, more megabytes!" We both spoke English but I used an option, a definition, he didn't understand.

Situations similar to this had been seen in the early days of BACnet, where implementers had chosen different options, blithely assuming other devices would also support those options. Sometimes they didn't, and things didn't work..

For this reason, the BACnet Testing Labs Working Group (BTL-WG) was formed in January, 2000. Its missions: develop test procedures for verifying devices' conformance to BACnet, and to research and define on a case-by-case basis what was required to ensure interoperability between BACnet devices. The goal is to maintain the flexibility of the BACnet standard as much as possible, rather than reducing it to a Procrustean "one size fits all" approach.

Since its formation the BTL-WG has studied innumerable interoperability cases, starting with the simple ones presented by the simplest controllers, working up presently to the very complex issues that come with operator workstations. They have made determinations in each case as to what is needed for interoperability.

The end result of this work can be found in two documents published by the BTL-WG. The "Implementer's Guide," (found at <http://www.bacnetinternational.org/btl/resources.php>) lays out a number of basic requirements, suggestions and mistakes commonly made by implementers. The Functionality Checklist (<http://www.bacnetinternational.org/btl/documentation.php>) in the BTL Lab's test submission documents lays out further requirements and recommendations to achieve interoperability.

Whether or not you intend to have your device tested by the BACnet Testing Lab or the European WSPLab (I strongly recommend you do), conforming to these requirements and recommendations beforehand will save you many headaches later on. The following are links to the two organizations' testing:

<http://www.bacnetinternational.org/btl/>

<http://www.big-eu.org/eng/conformance/index.php>

I will note also that both organizations conduct Interoperability Workshops (a.k.a. "Plugfests"), several days during which teams from multiple manufacturers bring their devices, completed or not, and test them together. Although the paired teams are often from competitors, the spirit is one of cooperation, aided by the fact that the public is not permitted in these sessions – the results are private. The BACnet news website (<http://www.bacnet.org/>) provides after-the fact summaries of these events, along with a photo.

BACnet Tools

There are a number of tools, such as protocol analyzers ("sniffers" able to decode BACnet messages), on the market that are of considerable value when developing a new BACnet device.

One especially valuable tool is the BACnet "Visual Test Shell" (VTS), downloadable from SourceForge (<http://sourceforge.net/projects/vts/>), which runs on Windows NT, 2000 and XP. VTS allows one to create arbitrary requests, send them to a device and see the response. It also has capabilities to support BACnet clients, and even provides a means for taking a request and changing it to make it malformed (an incorrect tag, a dropped octet, etc.) to see how the device handles malformed requests.

Summing Up

There are a lot of resources available to new BACnet developers today, resources to make it easier to correctly implement BACnet devices. The early adopters blazed the trail, you now have the roadmap. Good luck – and consider attending the BACnet committee meetings announced on the BACnet web site. They're open to all interested parties, no fee required, and you'll learn a lot.

Bill Swan is Building Standards Initiatives Leader for Alerton & Honeywell, BACnet Committee Chair 2004-2008, a member of ISO Technical Committee 205 which adopted BACnet as an international standard, a member of the BACnet Testing Labs working group, and an executive board member of the BACnet Interest Group – Europe.