INTERPRETATION IC 135-2012-5 OF
ANSI/ASHRAE STANDARD 135-2012 BACnet® -
A Data Communication Protocol for Building
Automation and Control Networks

Approval Date: June 22, 2013

Note: Transferred from the 2010 edition of Standard 135 (IC 135-2010-14).

Request from: Gerhard Bahr (gerhard.bahr@honeywell.com), Honeywell GmbH, Böblinger Str. 17, Schönaich 71101.

Reference: This request for interpretation refers to ANSI/ASHRAE Standard 135-2012, Clauses 12.4.9, 12.8.9 and 12.20.9, regarding the Out_Of_Service property.

Background: In Clauses 12.4.9 / 12.8.9 / 12.20.9 of the Standard, the meaning of the third sentence is unclear. Either it does not belong to this section or it is meant to describe an alternative to the previous sentences. In addition to that the language for the newer Value objects differs from the above mentioned. Depending on the way the Value object is used, the desired behavior of the object could be different. Also it is not clear, what is really meant by "software local to the device". Applications may be restructured in a way that a producer or consumer of the value is moved from inside a device to another device. Is the behavior expected to change then?

Language of the standard:

(1) The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the Present_Value of the …. Value object is prevented from being modified by software local to the BACnet device in which the object resides.

(2) When Out_Of_Service is TRUE, the Present_Value property may be written to freely.

(3) If the Priority_Array and Relinquish_Default properties are present, then writing to the Present_Value property shall be controlled by the BACnet command prioritization mechanism. See Clause 19.

The language of Clauses 12.37.9 / 12.38.9 / 12.39.9 / 12.40.10 / 12.41.9 / 12.42.9 / 12.43.9 / 12.44.9 / 12.45.9 / 12.46.9 / 12.47.9 (other value objects) differs from the above mentioned clauses.

(1) The optional property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the Present_Value of the …. Value object is decoupled from software local to the BACnet device in which the object resides that normally produces the Present_Value as an output or consumes it as an input.

(2) When Out_Of_Service is TRUE, the Present_Value property may be written to freely.

If an external Schedule or Command is writing to the Present_Value, then the only way to decouple this external data flow is to write with a higher priority, or to perform a local override within the device (i.e. the Overridden bit is TRUE in the Status_Flags property).

Following are a couple of pertinent emails on the topic from BACnet-L:

Regrettably the original idea behind Out_Of_Service has become obscure over the years by several well-meaning, but erroneous, "clarifications" to the standard. There has been an effort in the past several years to craft language that again clarifies the exact meaning in several new addenda.

Value objects (AV, BV, MSV etc.) have the unique characteristic that they can represent values known to the device (sort of like "inputs") or values that we give to the device (sort of like "outputs"). Picture the value object's Present_Value as a bucket that contains some value. When the value object is acting like an input, some process in the device periodically updates the bucket with a new value, maybe every 10ms or every second, however often it needs to. When the value is acting like an output, other BACnet devices or humans fill the bucket by writing to the Present_Value property through BACnet. Again some process periodically (as often as it wants to) gets the value from the bucket and does something with the value.

The operative term is "some process" which is periodically reading from or writing to the bucket. Keep in mind that the "bucket" is essentially the Present_Value property.

The idea of Out_Of_Service is similar for inputs and outputs. If the object is "in service" (Out_Of_Service=FALSE) then the bucket may be filled by the internal process if the value object is an input, or the process may use the bucket value if the value object is an output. If the object is "out of service" (Out_Of_Service=TRUE) then the process may NOT fill the bucket (if it's an input) or may NOT use the bucket value (if it's an output). So Out_Of_Service is like a switch that connects the bucket to the internal process.

On the BACnet side however we read and write to the bucket only. The prioritization mechanism for commandable values only writes to the bucket.

If you have an input object you would normally never write to the Present_Value because a few milliseconds later the internal process is going to overwrite your value anyway. If Out_Of_Service is TRUE then this internal refreshing of the bucket is prevented, so you could theoretically write to Present_Value and your value would stay there. In the case of an output when you write to the Present_Value AND it's Out_Of_Service is TRUE, the internal process is not allowed to look at the bucket so it doesn't change its behavior and just continues to use the last value it got before Out_Of_Service became TRUE.

So the Clause 12.8.9 language (and similar clauses) is overly broad when it says:

> *"… the Present_Value of the Binary Value object is prevented from being modified by software local to the BACnet device in which the object resides."*

What it should really say is something like:

> *"… for input-style Binary Values, the Present_Value of the Binary Value object is prevented from being modified by the internal process which normally updates the*

Present_Value in the BACnet device in which the object resides. For output-style Binary Values, any value written to Present_Value is prevented from being used by the internal process which normally uses Present_Value to perform the output function."

In your example, the DDC program is separate from the Binary Value itself. This is no different from a human operator, or control program in another device, that wants to use say WriteProperty to write to that BV Present_Value. All of these are allowed and should have no knowledge of the state of Out_Of_Service. Only the internal mechanism of the BV object cares about Out_Of_Service.

David Fisher
Polar*Soft*® Inc.

I agree with everything that David Fisher wrote about Out_Of_Service. I happened to be answering a question about Out_Of_Service, Reliability, Present_Value, and error codes yesterday, and I would like to inject some of that here, as in some ways it goes beyond David Fisher's statements, in case people gain greater understanding from seeing it expressed a couple different ways.

A writable Out_Of_Service property is always recommended, and never required. It is for purposes of testing only. Several scenarios of behavior are all-but-impossible to bring about "on demand" in a working device, such as seeing what the system does while Present_Value is a particular, seldom encountered value. It is nice, for purposes of testing only, to make them easy to bring about, while Out_Of_Service property is True.

The main one is to make Reliability property accepts write requests while Out_Of_Service property is True, but there are also some scenarios regarding values in Present_Value property as well. Some values you maybe want to return VALUE_OUT_OF_RANGE or WRITE_ACCESS_DENIED to Write requests if they occur in a working device. But testing what happens if that *does* become the value can still be a desirable testing scenario. While Out_Of_Service property is True, is when to let Present_Value take on those "tricky" values for testing purposes. By doing it while Out_Of_Service property is True, so that Present_Value is disconnected from being continuously written by any internal process and from any signals to the outside world, the behavior of the BACnet side of things can be observed, for testing purposes, while the rest of the system is disconnected from the consequences of those "tricky" values. Writing Out_Of_Service property as True, is the standard prescribed means to prevent the device from continuous overwriting those "tricky" values, so there is time to see the effects.

Everything about Priority_Array prioritization and also effects from and upon other BACnet properties (i.e. Relinquish_Default, Alarm_Values, Event_State) is expected to be just the same while Out_Of_Service property is True as when Out_Of_Service property is False. If Present_Value is written with a value of 4 at priority 12 in an object that contains a Priority_Array, then even while Out_Of_Service property is True, array index 12 of the Priority_Array becomes 4. If there is a value in Priority_Array at a priority superior to 12, then Present_Value takes on the value from the superior priority, not the value which was just written. The 3. statement which Gerhard originally posted is the attempt in the standard to state that in an object that has clause 19 command prioritization, the clause 19 command prioritization

still applies and acts on all affected BACnet properties, even while Out_Of_Service property is True.

More Write requests, but not necessarily all Write requests, should succeed while Out_Of_Service property is True. It is fine not to accept values that are always harmful and never need to be tested. You can still return VALUE_OUT_OF_RANGE to a Write request while Out_Of_Service property is True, whenever in your device that is the prudent choice. It is best to accept those Write requests bearing values that are useful for testing purposes, while Out_Of_Service property is True, but not necessary that all Write requests should succeed.

- Duffy O'Craven
BTL Manager

Interpretation No.1: The third sentence in Clauses 12.4.9 / 12.8.9 / 12.20.9 should be added to the Out_Of_Service definitions in other value objects as well.

Question No.1: Is this interpretation correct?

Answer No.1: Yes.

Comments: Not including this sentence in the description of all value objects that are commandable was an oversight.

Interpretation No.2: BACnet objects within the local device are considered to be local programming and as such cannot write / command the Present_Value of an object in which Out_Of_Service is TRUE.

Question No.2: Is this interpretation correct?

Answer No.2: Yes.

Comments:

Interpretation No.3: A DDC program in the local device is always considered to be local programming no matter if it is related to any BACnet object or not.

Question No.3: Is this interpretation correct?

Answer No.3: Yes.

Comments:

Interpretation No.4: It is a local matter whether, or not, user commands from a directly connected HMI are allowed when Out_Of_Service is TRUE.

Question No.4: Is this interpretation correct?

Answer No.4: No.

Comments: It is not a local matter. User commands from an HMI local to the device are not affected by Out_Of_Service.