bacnet experts

ASHRAE BACnet®

Scheduling in BACnet

David Fisher

24-Mar-2021

TUTORIAL

Contents

Scheduling in BACnet

24-Mar-2021

David Fisher

Introduction

Within any building automation system (BAS) there is a need to make things happen at specific times on specific days of the year. In BACnet this is called *scheduling*. The goal is to allow devices to expose their internal schedules so that other BACnet devices can "see" them and understand them in an interoperable way. More importantly, most BACnet devices that implement schedules do so in a way that allows other devices to alter their schedules interoperably as well. The topic is complex so let's start by breaking down the parts.

As we know, BACnet represents input, output and programmatic/calculated points as *properties* of *objects*. Generally speaking, an object in a device is targeted by a numeric *object identifier* that identifies one particular object supported by that device. Similarly, each object exposes multiple properties that represent discrete values. Some properties are "inputs" to the object in the sense that they may be changed by a process internal to the device, or by being written by another BACnet device using WriteProperty or WritePropertyMultiple services. Some properties are "outputs" from the object in the sense that they represent some quantity known to the object, such as the temperature read from a temperature sensor. Some properties serve both roles.

As a rule, if we want a BACnet device to "do something" at a particular time, then when it becomes that time we would write a value to an object property. As a side effect of changing an object property value, some action occurs. For example, if we have a light that is represented by the Present_Value of a binary-output object and we write a 1 to that Present_Value, then the light is turned on, and similarly writing a 0 turns it off. If we have a valve motor represented by an analog-output then when we write 50 to its Present_Value the motor positions to 50%. Writing 100 to it positions the motor to 100% etc. If we want a schedule to "make things happen" it will ultimately mean writing a value or values to one or more object properties.

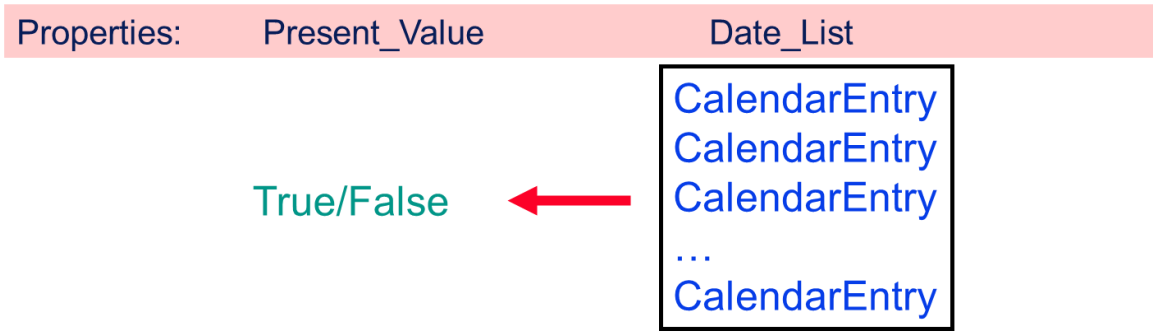
If a schedule is very simple it may need to do the same things at the same times every day regardless of the date. But it is much more common that we need to do different things at the same times on different days, based on the day of the week. In those cases, each day has its own schedule that may be different from the schedule for other days. BACnet calls these *weekly schedules*. A weekly schedule is used every week throughout the year and represents "normal" operation. Even when weekly schedules are used, it is also common to have special dates, or ranges of dates, that represent days when an alternative to the weekly shcdule must be used. BACnet calls these *special events*.

Although it's possible to represent scheduling for many points in a device by using a collection of schedules, there is a significant drawback to that. It is common to have special dates that apply to everything in a building or facility equally. In such cases if we need to change the date(s) associated with those special days, then we would need to locate and change every schedule that could be using those dates. At best that's a maintenance burden. BACnet has a kind of object called a Calendar object that is just for this purpose: to make in effect a list of special dates, on which something other than the weekly schedule is to be used.

Calendar Objects

The Calendar Object provides a network-visible representation of a collection of possible days out of a hypothetical year. The Calendar object is essentially a boolean variable that provides a true or false value on any particular day, based on whether that day is included within a list of possible dates.

The boolean value is provided by the Present_Value property of the Calendar object. The list of dates is provided by the Date_List property.



The Date_List property is composed of zero or more CalendarEntries. If any one of the CalendarEntries matches today's date, then the Present_Value is TRUE otherwise it is FALSE.

This feature is useful for describing "special" dates or ranges of dates that may affect the operation of schedules and other logic within BACnet devices. It also provides a standardized way of representing calendar information across different vendors.

Calendar objects are not required for scheduling, but they can be useful for controlling multiple activities that normally have different schedules, but share common dates for special events, holidays and so forth.

CalendarEntries

CalendarEntries are a special datatype that is used in both Calendar and Schedule objects. While it is possible, and common, to use a CalendarEntry to represent a single particular day of the year, the real power of CalendarEntries is in representing multiple dates.

A CalendarEntry can represent one of three types of specification for dates: Date, DateRange and WeekNDay. In all three forms, several parameters describe the particular date(s) for the CalendarEntry. A particularly powerful feature of these parameters is that any combination of them may be a "wildcard" or ANY value.

- Date month,day,year
- DateRange (from m,d,y)(to m,d,y)
- WeekNDay month,week,weekday

month	1..12 or ANY
day	1..31 or ANY
year	e.g. 2021 or ANY
week	1..5, 6=last week of month, or ANY
weekday	1..7 (1=Monday) or ANY

Date month,day,year

The simplest form is called a "Date." A Date has three parameters: month of the year, day of the month and year. Months may have any value from 1..12 representing January through December, or ANY meaning any month. Days may have any value from 1..31 that is appropriate for the specified Month. So, if the Month is January, then the Day may be 1..31, but if the Month is February then the Day may only be 1..28, except on Leap Years and so forth. The year is the absolute year, e.g. 1998. By using the ANY value in one or more parameters, you can easily create dates with broader meaning. For example (Month=ANY, Day=1, Year=ANY) represents the first day of any month of any year.

DateRange (from m,d,y)(to m,d,y)

The next form is called "DateRange." As you might expect, this represents a range of dates beginning with a starting or "from date" and ending on a "to date." The from/to dates are inclusive. The same rules that applied to Date parameters also apply to DateRange parameters.

One problem with DateRanges is: what does it mean when one or more parameters are ANY? There is a long and complex history behind DateRanges in Schedules. The short version is that the original 1995 language in the standard was not sufficiently clear about how DateRanges were intended to work, due to a lack of consensus about the wording. Regrettably, this caused several incompatible implementations to exist in the marketplace. When the SSPC attempted to readdress this issue in the 2004 standard, the compromise was to make the restriction of "all wildcard" or "no wildcard" as the only options for DateRange. This is too bad because there are many DateRange wild combinations that are useful.

The original idea (which was not articulated thoroughly in the standard) was that in DateRanges where there are any wildcards, each of the four ranges needs to be treated in a strict order of precedence: year, month, day-of-month, day-of-week. The test for something being "within the date range" is that it passes four tests in this order:

1. Is today's year within the range of years?
and
2. Is today's month within the range of months?
and
3. Is today's day-of-month within the range of days?
and
4. Is today's day-of-week within the range of days of week?

So for example:

startdate year=ANY, month=ANY, day=ANY, dayofweek=Monday
enddate year=ANY, month=April, day=ANY, dayofweek=ANY

To match this range a date must perform these steps:

- since start and end year=ANY it always matches year
- since start and end month is ANY through April this is the same as having said "January through April"
- since start and end day=ANY it always matches day
- since start and end dayofweek is Monday through ANY this is the same as having said "Monday through Sunday"

So, the *intended* logic of the example boils down to “is the month between January and April inclusive”?

The key idea, which a lot of people didn’t understand because the standard was not clear enough, was that the evaluation of range is different when there are any “ANYs” in the 8 fields of the range.

This is a moot point now, but that’s how we intended for it to work. Today's standard allows for only three cases:

DateRange where there are no "ANY"s

DateRange from (ANY,ANY,ANY,ANY) to (year,month,day,dayofweek) which means all days up to and including (year,month,day,dayofweek)

DateRange from (year,month,day,dayofweek) to (ANY,ANY,ANY,ANY) which means all days starting with (year,month,day,dayofweek).

WeekNDay month,week,weekday

The last form is called “WeekNDay.” The parameters are the month of the year, the week of the month, and the day of the week. The week of the month may be 1..6 or ANY. The week=6 is a special value meaning the “last week of this month.” This powerful format allows dates like (ANY,2,Tuesday) which would mean the 2nd Tuesday of any month.

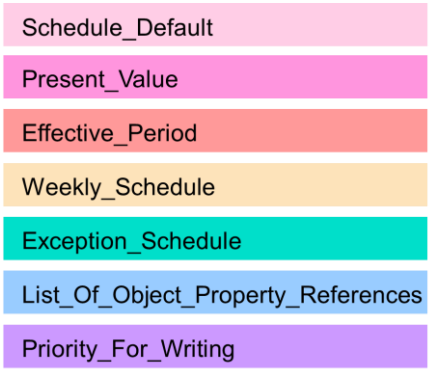
Schedule Object

A BACnet Schedule object provides a network-visible representation of a time-based series of events. The schedule object provides a collection of *TimeValues* that represent *any kind of value* that might be assigned to a property of an object at a particular time. The Present_Value of a schedule object is the most recently encountered TimeValue that was found to be applicable.

Using the Weekly_Schedule and Exception_Schedule properties, you can represent a rich collection of periodic and special dates, and collections of TimeValues for those dates.

Other programs and/or objects within a BACnet device can simply reference or "look at" the Present_Value of a schedule to find out what they should do. Normally however, it is the Schedule object itself that takes an action based on these schedules. The action is to write the appropriate TimeValue's value to one or more properties of BACnet objects, usually within the device containing the Schedule object.

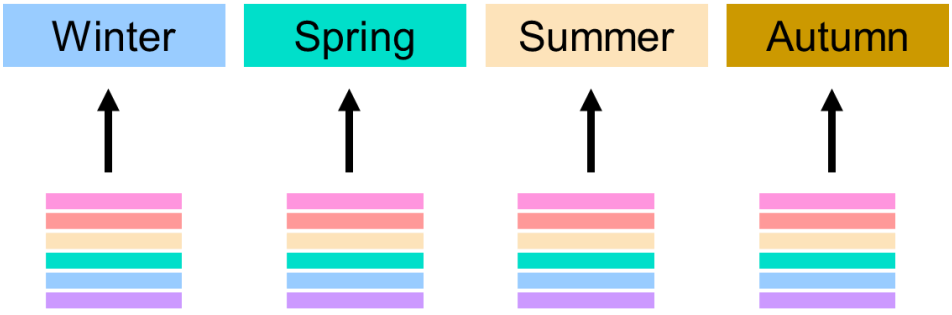
Since this behavior is potentially complex, we'll examine each of the components in greater detail. The Schedule object itself has seven important properties that we will show as a kind of "rainbow sandwich":



When the device is restarted, and at midnight, each schedule takes on a default value that is called the Schedule_Default. This value remains in effect until the first matching schedule occurs. The Present_Value property reflects the currently active scheduled value.

Effective_Period

The Effective_Period property is a DateRange that specifies when a particular schedule object is applicable. This can be useful, for example for seasonal scheduling using non-overlapping ranges and several schedule objects:



TimeValues

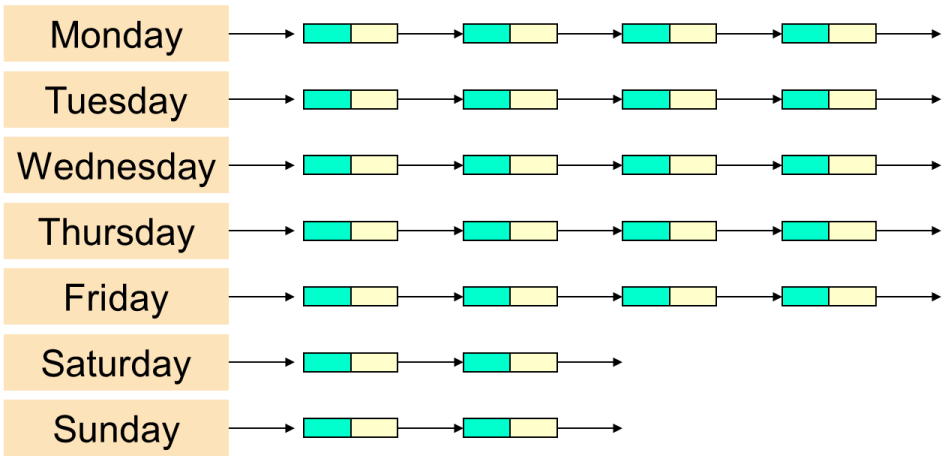
TimeValues are a datatype consisting of a time of day, and an arbitrary value. They are used in several places within schedule object components to specify a value to be written to the property of an object at a particular time of day.

time	*value*
07:00	On
17:00	Off

Weekly_Schedule

The Weekly_Schedule property is an array of seven lists of TimeValues that correspond to the seven days of the week. On a normal weekday, when a schedule is active, the corresponding list from the Weekly_Schedule array specifies those times at which values are to be written.

Note that a schedule object is not required to have a Weekly_Schedule if it has an Exception_Schedule.



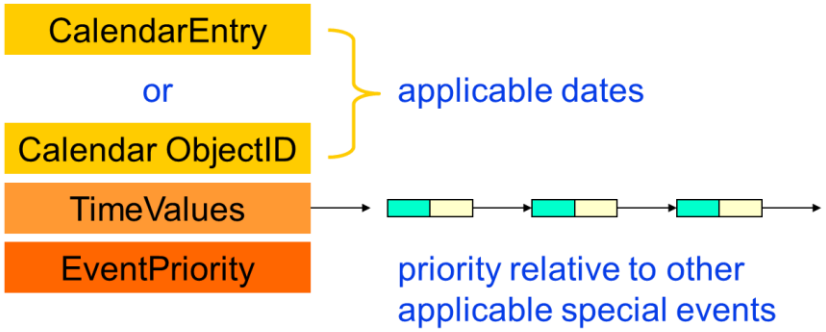
Remember that on any day between midnight and the first TimeValue's time, the value of the Schedule is the Schedule_Default. You can prevent this early day gap by including midnight in the first TimeValue. The TimeValues must be sorted in strictly increasing time order! It's not allowed to have multiple TimeValues with the same time.

SpecialEvents

SpecialEvents are a complex datatype that is used in the Exception_Schedule property of a schedule object. Like the Weekly_Schedule array, SpecialEvents are used to specify a list of TimeValues. However, SpecialEvents qualify the list of TimeValues in two important ways.

First, SpecialEvents qualify the TimeValue list by specifying a CalendarEntry that describes the date(s) upon which the SpecialEvent is valid, or alternatively by referencing a Calendar object. When referencing a Calendar object, the SpeicalEvent is only applicable on days when the Calendar object has a Present_Value of TRUE.

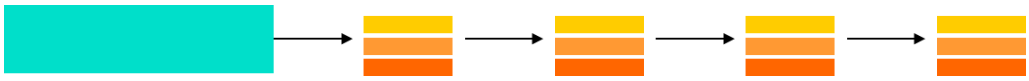
Second, each SpecialEvent includes an EventPriority from 1..16, with 1 being the highest or most important priority. This is used in situations where there are multiple SpecialEvents that are all applicable at the same time, for example because their calendar dates overlap. When this happens, the most important EventPriority SpecialEvent "wins."



Exception_Schedule

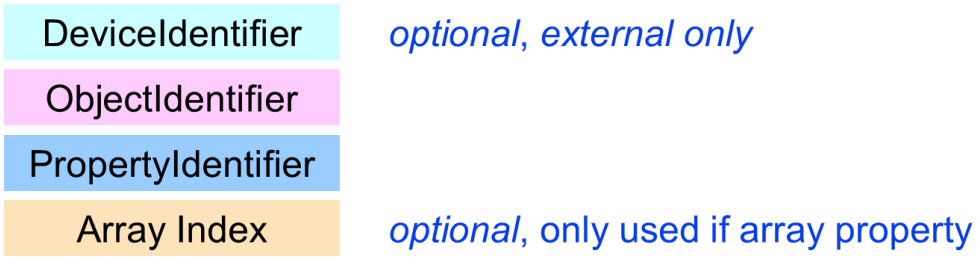
The Exception_Schedule property is an array of SpecialEvents that represent the collection of all dates and circumstances under which this schedule object overrides the Weekly_Schedule with a possibly different set of TimeValues.

Note that a schedule object is not required to have an Exception_Schedule if it has a Weekly_Schedule.



ObjectPropertyReferences

An ObjectPropertyReference is a special datatype that defines a property within an object that is to be written with a value. This complex datatype has four components, two of which are optional. The ObjectIdentifier represents the object containing the property to be written. The PropertyIdentifier is the property to be written. If the property is an array property, then the ArrayIndex must also be specified. This indicates which of the array slots is to be written.



BACnet allows schedule objects' ObjectPropertyReferences to optionally include a DeviceObjectIdentifier. This would be used in cases where it was desired to have a controller device maintaining the schedule object be able to write to object properties in other devices. To handle those cases, the ObjectIdentifier of the target device's Device Object would be included in the reference. In BACnet when we have such an "external" reference is is called a DeviceObjectPropertyReference.

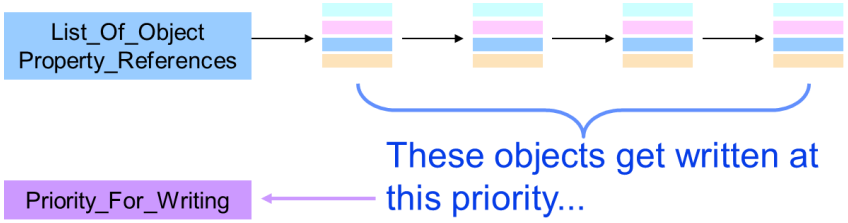
It is important to note that this feature (writing to external devices objects) may not be implemented in all devices that support Schedule objects.

List_Of_Object_Property_References

The schedule object's List_Of_Object_Property_References property specifies a list of DeviceObjectPropertyReferences. If non-empty, each referenced property is written with the value specified by the appropriate TimeValue at the time and on the date when that TimeValue becomes applicable, subject to arbitration among potentially competing SpecialEvents.

If and when writing occurs, the WriteProperty equivalent occurs at the priority specified by the Priority_For_Writing property. Don't confuse this priority, which is part of the Command Prioritization mechanism, with the EventPriority that arbitrates among simultaneously applicable SpecialEvents!

Schedules are not required to have any ObjectPropertyReferences in the list. Some types of BACnet devices may allow the list itself to be written using the WriteProperty or AddListElement services. Other devices may simply present the list as read-only.



Legal Stuff

Contact the Author

David Fisher

president, PolarSoft® Inc.
368 44th Street
Pittsburgh PA 15201-1761 USA
+1-412-683-2018 voice
dfisher@polarsoft.com
https://polarsoft.com