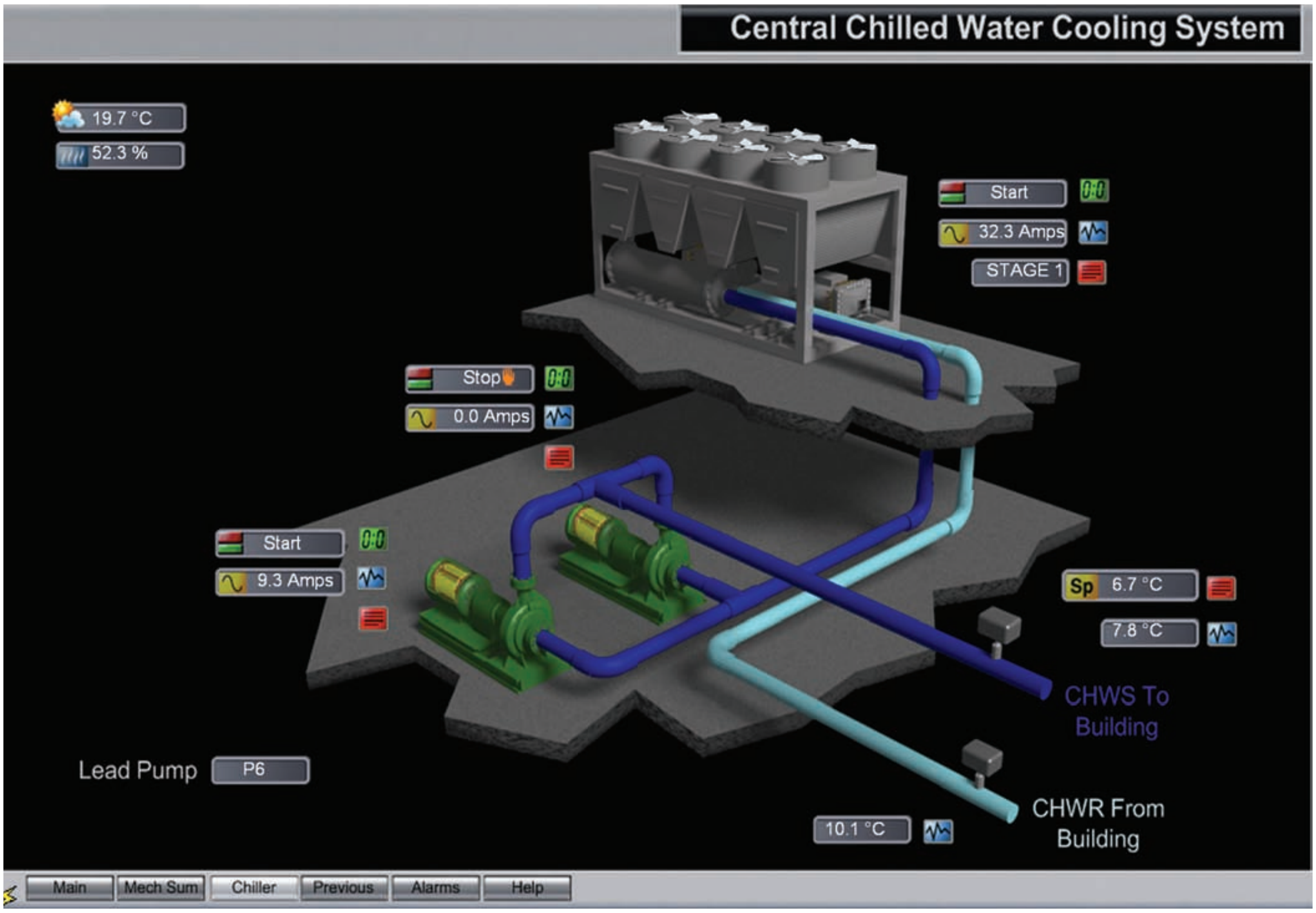

Figure 1: Operator workstations display system graphics populated with real-time data.

Visualizing BACnet®

By Roland Laird

What a BACnet system looks like and how it can be used generally depends on the capabilities of the system's BACnet controllers and the operator workstation (OWS). The user experience varies with each vendor's OWS as BACnet does not describe how user operations are performed. A BACnet OWS needs to perform every task, as much as possible, in an interoperable way so that the OWS can be used with any vendor's controllers. It is also desirable that actions performed at one vendor's workstation can be seen and reversed, if necessary, at another vendor's workstation.

This article provides the reader with a basic understanding of the standard capabilities of BACnet for performing eight typical building operations.

System Graphics

Commonly, the building operator uses the OWS for viewing graphic displays of either floor plans or mechanical systems populated with real-time data. The OWS is the primary window into the current operation of the entire building automation system (BAS). Animations and

About the Author

Roland Laird is president of Reliable Controls in Victoria, BC, Canada.

color quickly bring attention to any abnormal status. From the graphics, the user can override point values and troubleshoot or change other parts of the system.

Although BACnet does not specifically define interoperable graphic displays, it makes it possible to build graphics comprised of points from multiple vendors. BACnet's flexibility enables many possible ways for vendors to define graphics. The graphic may be defined entirely within the OWS software, entirely within a controller, or a hybrid with some aspects of the graphic definition in the controller and some in the OWS. Graphics for each vendor's OWS must be built specifically for that OWS.

The feature specification of an entire graphic system is beyond the scope of BACnet, but to date, two graphics-related BACnet standard objects exist, and one object is undergoing public review. These allow BACnet systems to group objects together that normally belong to individual graphics. However, none of these objects attempt to standardize display attributes and user operations for each point on the graphic. Standardization of graphical interfaces would not be wise because BACnet would lose the ability to enable OWS vendors to be adaptive while offering new and differentiating features.

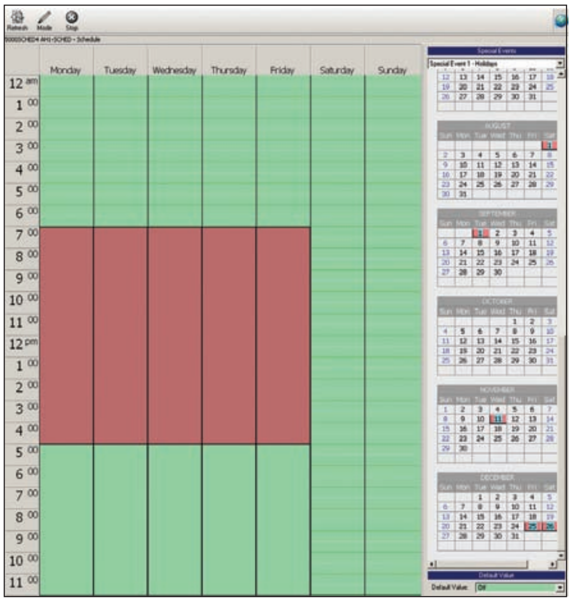

Figure 2: Typical BACnet weekly schedule.

The Group object introduced with the original BACnet standard provides for a list of objects (points) and their values. It is constrained to reference objects from a single device so that all values may be acquired simultaneously without networking delays. The Group object provides a convenient method to read all objects on the graphic and be assured that the values all were captured at the same time. Graphics that contain objects from multiple devices would need more than a single Group object to define the object set.

A different option would be using the Structured View object to define the objects contained in individual graphics. This object was designed to contain collections of objects that are linked to one another in some way. In addition, it has standardized attributes to indicate what type of view the collection is (mechanical system, area plan, etc.). The Structured View object supports a hierarchy of linked Structured View objects. The root Structured View of each hierarchical tree is maintained by a standard property in the Device object, which allows new workstations to learn the graphical architecture of the system. The Structured View object was adopted into the BACnet standard in 2006 but has yet to see much use in the BAS systems offered today.

Another option is the Global Group object, which is undergoing public review, which allows for references to any device in the system. This object has the unique capabilities of maintaining its member's values and sending change of value notifications and alarms to workstations.

Each vendor's BACnet implementation may or may not use any of these objects for containing the points on a graphic. Many systems maintain their graphics within the OWS or controllers in a proprietary format. This is definitely an area to watch for further developments as BACnet continues to evolve.

Weekly Schedules

Another common operation is making changes or exceptions to building schedules. The BACnet protocol defines a powerful Schedule object that is more than capable of handling most scheduling needs. The Schedule object can contain exceptions to the weekly schedule that may be specific dates, date ranges, repeating date patterns, or a link to a calendar. Overlapping exceptions are resolved by ranking the priority of each exception. BACnet scheduling can be as simple or as complex as the needs of the facility dictate.

BACnet Schedules are fully interoperable between vendors, but there will be differences in presentation.

Holidays

BACnet defines the Calendar object for containing a set of dates. As with the schedule object's exceptions, these dates may be specific dates, date ranges, or date patterns. Typically, calendars are used to override the normal weekly schedule so that when the building is unoccupied normal HVAC equipment operation does not occur. Schedules have the capability of being overridden by a Calendar object, but only when the Calendar object is in the same device. As a holiday calendar is common across many devices, vendors must provide a method for enabling the operator to make the holiday edits only once. Each of the following possible methods has its pros and cons in terms of maintenance and interoperability:

- Duplicating calendars in many devices;
- Controlling the equipment in the device's programming rather than directly from the Schedule object (including consideration of the holiday calendar); and
- A proprietary extension to BACnet Schedules to enable them to reference a master calendar in another device.

The first method is network visible and interoperable, but it's likely difficult to maintain duplication for every holiday edit. The second and third methods have interoperability concerns as third-party workstations are unable to change a device's programming or be easily able to use a proprietary extension.

Regardless of the method used, for the operator, the interface likely will display a calendar on which dates can be selected. But behind the scenes, any of the above methods, or others not discussed here, may be in place.

Manual Override

Probably the most frequent and basic task that an operator performs on the BAS is to override an object's value, such as overriding a failed sensor's value until the sensor is replaced, or changing a system setpoint. Often there is a need to quickly start or stop a piece of equipment that has not been controlled as expected by the automatic algorithm. This immediate action can resolve the situation until time can be devoted to correcting the problem. Manual override is typically done by the operator from the graphical displays, which also indicate the manual status of each value on the display.

Manual overrides are a simple task for the operator, but a BACnet system masks underlying complexity. This complexity is a result of the variation between objects and that not all objects have a dedicated property for manual overrides. The method by which the override is achieved in BACnet is different based on the type of object that is being overridden.

The Out_Of_Service property is a required property in most objects that often can be used for auto/manual control, as long as it is writable in the controller.

Input Override

In the example of a failed sensor, the operator will be required to manually override the Input object's value. The Analog Input, Binary Input, and Multistate Input objects all have the Out_Of_Service property, which can be used for this purpose. Writing Out_Of_Service to TRUE will disconnect the Present_Value of the input from the physical measurement and allow the Present_Value to be written with a new value. The Out_Of_Service property is reflected in the object's status flags so it is easy for the OWS to know and display that the object has been manually overridden.

Output Override

The second example is manually starting an exhaust fan. Typically, this would be done using a Binary Output object. Output objects have an Out_Of_Service property, but it is not suitable for Auto/Manual control. If Out_Of_Service is TRUE in Output objects, the Present_Value is disconnected from the actual physical output. This freezes the physical output at its current value and the output's Present_Value can be freely changed without concern for cycling real equipment. This is useful for testing of programmed sequences, but of no value for an operator's manual override need.

Advertisement formerly in this space.

Advertisement formerly in this space.

All Output objects have a built-in mechanism that includes a standard provision for operator Auto/Manual control, known as the Priority_Array property. The 16 slot writable Priority_Array controls the Present_Value to the value written at the highest priority. Priority slots can be relinquished, which enables values at a lower priority to take control.

In the example of starting the exhaust fan, the OWS would write the output using the Manual Operator priority, thereby overriding all the lower level automatic control priorities and cause the fan to start. If the fan still does not start, the likely cause is that there is another higher priority command keeping it stopped. For the operator to fully understand the output's value, the OWS must enable the operator to view the commands at each priority and relinquish the command, if necessary.

For outputs, there is nothing in the object's status flags to indicate that the point has been manually overwritten. The OWS must read the Priority_Array to determine what is controlling the object and provide indication to the operator.

A significant wrinkle exists to the previous, as this example can vary dramatically because several manufacturers use Value objects to represent physical outputs. The Value object is often used for outputs when the product does not have the resources to support the implementation of the Priority_Array required by outputs. In accordance with the BACnet standard, the Priority_Array property can be implemented optionally in Value objects. In the simplest implementation (no Priority_Array), the Out_Of_Service property can be written to TRUE and the Present_Value written to the desired value. If the Value object has a Priority_Array, there are two methods to manually override the fan; either the Priority_Array mechanism described for the Output object or by using the Out_Of_Service property. To be interoperable, the OWS must be able to determine the appropriate method for each piece of equipment.

Value Override

The third example is an operator overriding normally calculated setpoints contained in Value objects. Typically, a value object is overridden by writing the Value object's Out_Of_Service property to TRUE. This will stop internal calculations from changing the Present_Value of the Value object. However, if the Value object has a Priority_Array the setpoint value can be overridden by writing directly to the Present_Value using the 'Manual Operator' priority.

The BACnet OWS needs to be flexible to accommodate various vendor's implementations and present the Auto/Manual information in a simple, consistent manner to the users. A well-designed OWS ensures that building operators will not be concerned with the earlier details.

Another operator expectation of manual override is timed-override. In timed-override, objects revert back to their automatic value after an operator-specified time interval. In BACnet systems this is typically accomplished automatically by the workstation that the operator used to initiate the override. Optionally, vendors may implement proprietary extensions to objects that maintain the timed override status. This enables the workstation to be removed from the network before the completion of the override time.

Alarms

Alarms annunciate any abnormal system parameter. The annunciation needs to be as prominent as the alarm condition dictates. Often, alarms should reoccur until they are acknowledged or the condition is resolved. With a BACnet system, operators can expect to see and have the ability to acknowledge alarms from any vendor's controllers. However, the presentation and setup of alarms will vary between vendors.

BACnet defines two alarming mechanisms known as *intrinsic* and *algorithmic*. BACnet controllers are not required to implement both, but a BACnet OWS will be able to annunciate and record alarms generated with either mechanism. In *intrinsic alarming*, the conditions that define the alarm are contained within each object. The operator will define the alarm conditions as part of each object's setup. In *algorithmic alarming*, a separate object that defines the alarm condition is created for each alarm.

Both algorithmic and intrinsic alarms use the Event Notification service to annunciate alarms, which convey information including the device, object, timestamp, class of alarm, priority, message, and specific values that caused the alarm to occur. BACnet also provides Alarm Summary services that enable an OWS to poll devices to learn the current status of all alarms in the system.

Currently, the BACnet committee is working on several proposals to further enhance the interoperable alarming capabilities of BACnet systems.

Trending

Trend logs are indispensable for an operator when analyzing the performance of the system or in generating reports. Either of the two trend objects that BACnet provides for recording data can provide the user with presentations containing several object values. The BACnet Trend Log object provides for the logging of any single object either at a fixed interval or upon a change of value. The Trend Log Multiple object added to the standard in 2008 allows for multiple values to be logged simultaneously at a fixed interval or upon a trigger.

Operators can expect that a BACnet OWS will be able to configure and view single-point trends in any vendor's controller. However, as the Trend Log Multiple object is relatively new, it has not yet been implemented by many vendors. Alternatively, many single-point Trend Log objects are often presented together to create a presentation with multiple points.

Both the single-point and multipoint BACnet Trend Logs have the ability to send notifications when the controller's logging buffer is approaching full. This is useful for long-term archiving software to know when to retrieve data from a controller.

Programming

The heart of the BAS is the programmed operations sequence that the system performs automatically to control the facility. BACnet defines the Program object, which gives the overall status of each programmed sequence and provides properties that enable an OWS to request that the program be stopped and started. An operator can expect that this will be interoperable between vendors, but programming details are not within the scope of BACnet.

Backup and Restore

During the course of system operation, controllers experience database changes. The operator is required to maintain a current backup of each controller's database. BACnet makes it possible for any OWS to backup and restore any other vendor's controllers, including all proprietary objects.

Conclusion

The BACnet protocol enables a single OWS to operate a BAS comprised of controllers from many BACnet vendors. Operators should expect differences in controller capabilities, and therefore, certain operator functions may differ from one controller to another.⬢

Advertisement formerly in this space.