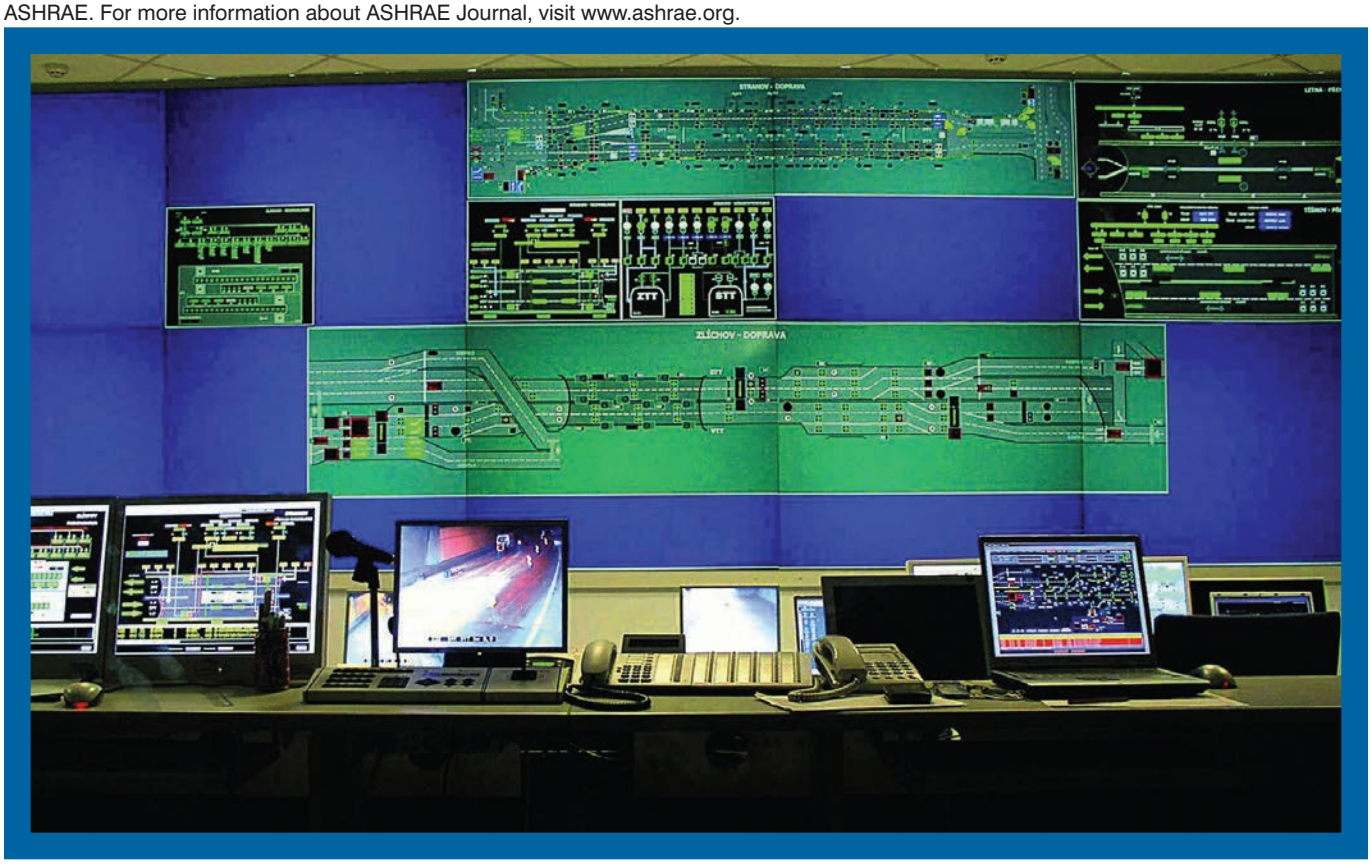

Image: ©Siemens AG

Event reporting has been revised in the BACnet standard, including improved fault reporting and a new alarm forwarding mechanism.

BACnet Alarming Revised

By Bernhard Isler, Member ASHRAE

The BACnet® data and services model for event reporting was developed to incorporate new object types and event algorithms. The experience gained from implementations and the installed base highlighted some issues with event reporting. To address these, the BACnet committee initiated a special effort for overhauling the event reporting specification. This includes clarification and enhancement of the model, consistency among intrinsic and algorithmic reporting, improved fault reporting, better scalability through a new alarm forwarding mechanism, and a new stateless alert notification feature.

Do not worry; it is not all new! The alarming revision was carefully crafted to preserve backward compatibility to the maximum extent. However, for the sake of consistency and correctness, some tweaks were applied here and there. But good notification clients (e.g., workstations) will seamlessly work with devices that implement the revised event reporting. A new and supplemental Notification Forwarding feature enables improved scalability. Stateless Alert Reporting has been added to allow report-

About the Author

Bernhard Isler is a senior system architect at Siemens Building Technologies Division, Control Products and Systems, Zug, Switzerland.

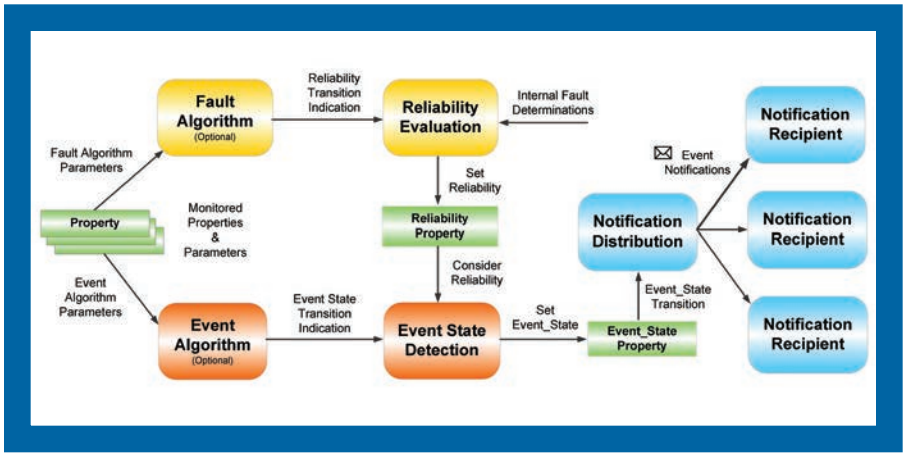

Figure 1: Event detection and reporting model.

ing alerts. Alerts are informational and do not interfere with the regular event reporting. Treatment by an operator is not expected for alerts.

The revision is published by ASHRAE as Addendum 135-2010*af* to ANSI/ASHRAE Standard 135-2010. In the following discussion, many small details have been left out so as to focus on key topics and how they look after the revision. For a detailed list, see the sidebar, "Other Notable Changes in the Revision."

Event Detection and Reporting Model

The event detection and reporting model (*Figure 1*) was clarified in its concepts and terminology. Along with this, language and concept consistency has been brought into the standard, for both intrinsic and algorithmic event reporting. The specification is concentrated in the rewritten Clause 13.2 of the standard.

With the revision, the only difference between intrinsic and algorithmic event reporting mechanisms is that the monitored values for the Event and Fault Algorithms are from properties internal to the object (intrinsic), or from properties of another object (algorithmic). The Event and Fault Algorithms used for both mechanisms are exactly the same, and are now specified only once in the standard.

For Offnormal Event State Detection, an Event Algorithm evaluates monitored properties, and requests a transition of the event state. From this and the Reliability property, the Event State Detection function determines the resulting event state and sets the Event_State property accordingly. All standard Event Algorithms are now specified at a single place in the standard (rewritten Clause 13.3), using a more formal specification language. New options have been added to some of them so as to allow additional transitions, typically to the same event state under defined conditions.

In fault detection, a Fault Algorithm may be involved. Standardized Fault Algorithms replace parts of former intrinsic reporting specifications for faults, for example in Multi-state Input objects. The Reliability Evaluation function considers the Fault Algorithm's result, as well as other fault determinations internal to the object. Finally it sets the Reliability property. Fault detection is connected to event reporting solely through the Reliability property. The new Fault Algorithms are now specified using the same language as for Event Algorithms (rewritten Clause 13.4). The former content of clauses 13.2 to 13.4 has been placed reasonably in these new clauses, but is not lost.

The distribution of Event Notifications is performed by the Notification Distribution function. The well-known Notification Class object belongs to this function. Sending event notification messages to Notification Recipients is triggered by transitions of the Event_State property.

Event State Detection State Machine

The Event State Detection runs a state machine, whose current state is indicated to the outside through the value of the Event_State property. The state machine has been clarified to properly cover transitions to and from faults (*Figure 2*). The state machine is basically driven by the results of the Event Algorithm, but also

Other Notable Changes In the Revision

- Property presence requirements are refined and clarified for optional properties.
- Event reporting property language is made consistent throughout.
- The acknowledgment mechanism for events is clarified.
- Intrinsic event reporting can be configured to be enabled or disabled per object.
- Event State Detection can be inhibited dynamically by a flag that is local to the object or in another object.
- Reliability evaluation can be inhibited dynamically by a flag local to the object.
- A distinct time delay has been added to the To-Normal transition.
- Event message texts to be conveyed in event notifications may be configured through BACnet.
- Recipients may not ignore event notifications due to a message text character set that is not supported by the recipient.
- Requirements on notification servers are relaxed in that minimal forms of recipient lists are allowed.

by the value and transitions of the Reliability property, and other conditions in the object.

It is now clearly specified that the Fault event state may be entered from any event state, including the Fault event state itself. The event state is Fault whenever the Reliability property indicates that a fault condition has been evaluated, i.e., the value is different from No-Fault-Detected. In the Fault event state, the Event Algorithm is not executed. Most notable, leaving from the Fault event state is always a transition to the Normal event state, regardless of the current event detection conditions. The Event Algorithm is resumed after returning to the Normal event state.

Transitions to and from the Fault event state were formerly notified using the same notification message variants as used for transitions to and from Offnormal. To convey the value of the Reliability property (a key piece of information in the Fault event state), the new event notification variant Change-of-Reliability is now used for the notification of any transition to and from Fault.

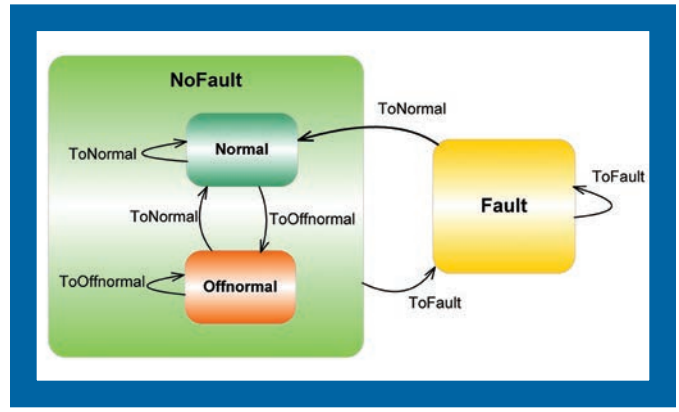

Figure 2: Event State Detection state machine.

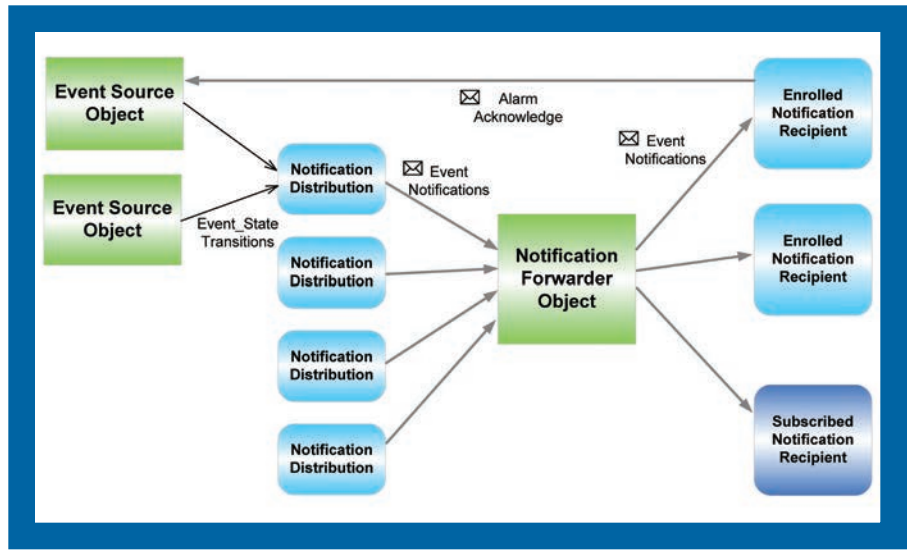

Figure 3: Notification forwarding.

Applying the Algorithms

The exact same fault and event algorithms are applied in intrinsic and algorithmic reporting. In intrinsic reporting, the algorithms monitor properties local to the object. In algorithmic reporting, the algorithms run in Event Enrollment objects and monitor properties in other objects local to the device, or even properties of objects in remote devices. Event Enrollment objects now also support reporting of Fault event states, consistent with the Fault event state reporting in intrinsic reporting.

Objects may run a Fault Algorithm only, an Event Algorithm only, or both. Objects not running a Fault Algorithm will not go into the Fault event state due to property values monitored by the Fault Algorithm. But internal determinations may still lead to a Fault event state and respective Change-of-Reliability notifications.

Objects not running an Event Algorithm will not go into any Offnormal event state. A Fault Algorithm or internal determinations may cause the object to enter the Fault event state.

Notification Forwarding

A new mechanism has been added that enables enhanced scalability of event reporting. This complements the Notification Distribution on the recipient side; therefore, it is completely unrelated to Event State Detection. The core element of this mechanism is the new Notification Forwarder object type. This object type represents a Notification Recipient that forwards event notifications, received from Notification Distribution, to subsequent Notification Recipients (*Figure 3*). For the ultimate recipient, the essential content of the event notifications is not changed and remains as issued by the original event source object that detected the event state transition. The Notification Forwarder object is not involved in the acknowledgment of event state transitions. Acknowledgment requests are sent to, and handled by, the original event source object.

How does this enhance scalability? At the lower end of the scale, small devices need to support just a single fixed Notification Recipient, which is "all devices on the local BACnet network," i.e., the destination address is set to local broadcast. In this network, one device that is more capable is installed that has a Notification Forwarder object. This Notification Forwarder object acts as a recipient for all notifications that are broadcast on the local network. It forwards them to all Notification Recipients enrolled or subscribed in the Notification Forwarder.

On the higher end, Notification Forwarders may be used to simplify device configuration. With the support of temporary Notification Recipients, devices or the system can be config-

ured so that a temporary device may subscribe at a single point, i.e., a central Notification Forwarder object, for receiving all notifications generated in the device or system.

The Notification Forwarder supports two ways by which Notification Recipients are known. Enrollment of Notification Recipients works in exactly the same way as it does in regular Notification Class objects. Subscription is a new mechanism that has been added. It supports runtime subscriptions, as in Change Of Value (COV) reporting. The subscriptions are considered temporary and have a lifetime, after which the subscription is automatically dropped. This mechanism was added to support Notification Recipients that connect at runtime (typically temporarily) and therefore, should not cause persistent configuration changes.

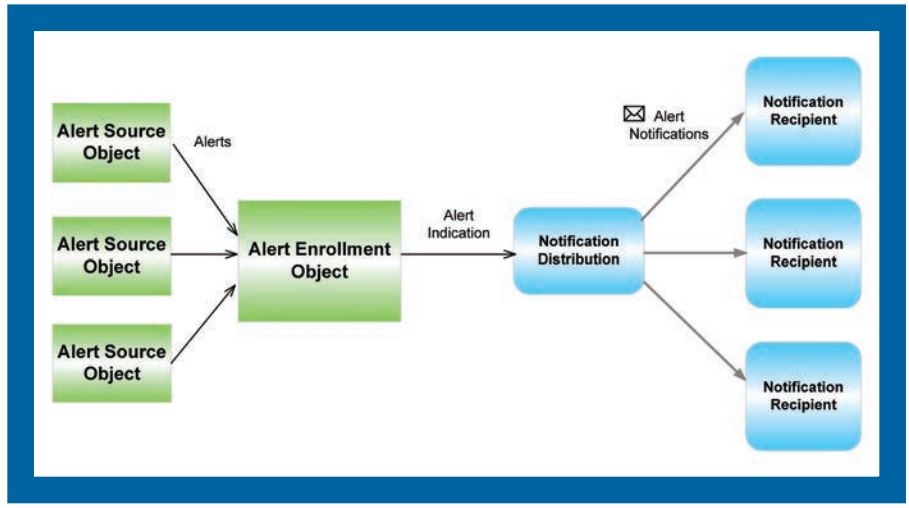

Figure 4: Alert reporting.

Process Identifier	As defined in the Notification Class for the recipient.
Initiating Device Identifier	The Device object's identifier of the device in which the Alert Enrollment object resides.
Event Object Identifier	**The Alert Enrollment object's identifier.**
Time Stamp	The time of occurrence of the alert.
Notification Class	The Notification Class referred to by the Alert Enrollment object.
Priority	The priority as defined in the Notification Class for the To-Normal transition.
Event Type	Event type 'Extended' always.
Message Text	Message text associated with the alert.
Notify Type	The value of the Alert Enrollment object's Notify_Type property.
AckRequired	This flag is cleared (FALSE) always.
From State	Event state Normal always.
To State	Event state Normal always.
Event Values	The 'Extended' option of BACnetNotificationParameters is used.
{	
Vendor ID	The identification of the vendor that defined the alert notification.
Extended Event Type	Numeric extended event type identifier as per the Vendor ID.
Parameters	A sequence of primitive data types or property values.
{	
Alert Source Object	**The object identifier of the Alert Source object. Always first in the 'Parameters' sequence.**
{other parameters}	Optionally, a sequence of primitive data type values or property values.
}	
}	

Figure 5: Alert notification parameter details.

Stateless Alert Reporting

In the past, many manufacturers implemented proprietary mechanisms to report informational alerts. The standard had no good way of generating such notifications. The standard event notifications are bound to an Event_State property, and transitions of the value of this property are reported. With the alarming revision, the new concept of stateless alert reporting has been introduced for providing informational alerts. This is a concept separate from Event State Detection. Informational alerts are stateless and not acknowledgeable. As an example, an object may report that the equipment it represents needs maintenance.

The two major elements in alert reporting are: a new Alert Enrollment object, and alert notification messages. The Alert Enrollment object is the core element in the concept and acts as the Event Source object for the alert notifications (*Figure 4*). Alert notifications are regular event notifications of event type 'Extended', but applied in a particular way.

The Alert Enrollment object contains all information required for managing information alerts from objects in the device. If a single Alert Enrollment object is present in a device, then all informational alerts from all objects in the device flow through that object. Multiple Alert Enrollment objects may be present in a device. In that case, the association of Alert Source objects with Alert Enrollment objects is determined by the device, but is not net-

work visible. If there are alerts that are not effectively initiated by a particular object, then the Device object acts as the Alert Source object.

For distribution of alerts, an Alert Enrollment object is linked to a Notification Class object that is part of the regular Notification Distribution function. There may be distinct Notification Class objects for alerts, or the same Notification Class objects are used for the alerts as are used for event state transition notifications.

Alerts are stateless and are not related to the event state of the Alert Source object, if any. Also, alerts do not support acknowledgment. An Alert Source object may issue various alerts, and does not need to have properties related to event reporting. Any object present in a device with an Alert Enrollment object, including the Device object itself, may issue alerts, regardless of its object type and support of optional functionality. The determination of conditions to issue an alert is internal to the Alert Source object and not network visible.

For consistency, the Alert Enrollment object has the common set of properties related to event reporting. But most of them will contain a static value. For example, the Event_State will always indicate the Normal event state. The Acked-Transitions flags will always have all flags set. In others, only the value for the To-Normal transition is relevant.

Alert Notifications

Alert notifications are conveyed using the standard confirmed or unconfirmed event notification service requests. The difference between event notifications from Event State Detection and that from alert notifications, is that the Event Source object is always and exclusively of type Alert Enrollment. There are no regular event notifications from an Alert Enrollment object.

Since the Event Source object conveyed in the notification is the Alert Enrollment object, how does the notification recipient know about the Alert Source object that has issued the alert? And further, how is alert information conveyed in the notification? What information is conveyed? The answer is that the 'Extended' variant of event notifications is used. Although the notification parameters of this variant are defined by the vendor, the first extended parameter value must be the object identifier of the Alert Source object. *Figure 5* shows the resulting event notification, and what the single message parameters convey.

The alert notification parameters are filled in from different sources, if not constant. The Alert Enrollment object, as the Event Source object, provides the 'Initiating Device Identifier', the 'Event Object Identifier', the 'Time Stamp', the 'Notification Class', the 'Message Text' and the 'Notify Type'. The Notification Class object, referred to by the Alert Enrollment object, provides the 'Process Identifier' and the 'Priority', from values of its respective properties.

All that is conveyed in the 'Event Values' section of the alert notification is provided by the effective Alert Source object. The 'Vendor ID', in a sense, identifies the name space for the particular type of alert conveyed. This typically will be the Vendor ID of the device manufacturer. A manufacturer could also use some other vendor's alert definition, so the other manufacturer's Vendor ID would be used here. Also, ASHRAE could define standard alerts under its Vendor ID 0. The 'Extended Event Type' is a numeric identifier of the particular alert type. These identifiers must be administered by the respective vendor indicated in the 'Vendor ID' parameter.

In the 'Parameters' section, the Alert Source's object identifier must be present as first value. This is where recipients have to look for the Alert Source object, which originated the alert. Following the Alert Source object identifier, an arbitrary sequence of primitive data type values or property values that include the property identification may follow. The effective content and sequence is defined by the vendor, and interpreted by the client based on the 'Vendor ID' and the 'Extended Event Type'.

Future Developments On Alarming

With Addendum 135-2010*af*, the first phase of alarming revisions and enhancements has been completed. The Objects & Services Working Group is working on further improvements. The following topics are being discussed:

- Extension of Event Algorithms and addition of new algorithms to support additional data types of values monitored by the algorithms. As an example, a new Change-of-Value event algorithm variant is to be added that will support event reporting on change of discrete values.
- New Fault Algorithms that define limits for floating point and other numeric values beyond which a particular fault is indicated.
- Relaxation of requirements to support complex alarm summarization services; only the execution of GetEventInformation is required to be supported by event reporting devices.
- Update of related topics to become consistent with the alarming revision, e.g., workstation BIBB requirements, or alarm and event BIBBs for event subscription.

You may now be overwhelmed by this revision. So was the committee when the topics to be addressed were identified, and the amount of work needed became clearer. For timely completion, a number of topics were postponed and excluded from this revision. There is still work ahead, but it will build on this revision. The topics currently in discussion are outlined in the sidebar, "Future Developments on Alarming."

Most of the changes are clarifications, consistency improvements, or are optional additions. This revision represents a significant improvement to the BACnet standard.