



BACnet

Compare Intrinsic and Algorithmic Reporting - The project engineers view. -

Version of Layout: EN 1.00
Author: Uwe Haeseler

- 1. INTRODUCTION..... 3**
 - 1.1. BACKGROUND 3**
 - 1.2. MARKET SITUATION 3**
 - 1.3. COMMON DESCRIPTION 4**
 - 1.4. USED PICTURES 4**
 - 1.5. LAST BUT NOT LEAST 4**
- 2. ENGINEERING DESCRIPTION 5**
 - 2.1. COMMON FEATURES 5**
 - 2.2. INTRINSIC REPORTING 6**
 - 2.3. ALGORITHMIC REPORTING 8**
- 3. COMPARISON 13**
 - 3.1. ENGINEERING 13**
 - 3.2. BACNET FRONT-END USER/OPERATOR 14**
 - 3.3. CONCLUSION 15**
- 4. OTHER PUBLICATIONS RELATED TO BOTH METHODS OF REPORTING..... 16**
- 5. LITERATURE 16**

1. Introduction

1.1. Background

At BMS installations the alarm management is a general and important function. The BACnet Standard defines two methods to handle that functionality, *Algorithmic Reporting* and *Intrinsic Reporting*.

Often consultants request in bid invitations **either** *Algorithmic Reporting* **or** *Intrinsic Reporting* to handle the alarm management in BACnet projects and like to exclude in a time the other method of reporting. The reasons for that behaviour could be:

- (1) Not enough knowledge about the practical aspects of both methods of reporting.
- (2) A possibility to prefer special vendors and exclude other products.

Against point (1) that paper hopefully helps.

The main focus of that paper is the practical use of both reporting methods. The general knowledge about alarm and event reporting will be presupposed.

Also there are discussions to extend the BACnet standard, as with the *Intrinsic Reporting* only one limit pair for analog data points is possible but there are sometimes two limit pairs required. In reality the BACnet standard does not need to be modified to achieve further limits, because that issue could be handled with *Algorithmic Reporting* (see sample below).

Details about *Algorithmic Reporting* and *Intrinsic Reporting* are described at [1] and [2]. There are also Documents in German Language: [3], [4] and [5].

1.2. Market situation

Relating to alarm management we can classify available devices as follows:

- simple devices without alarm management
- devices with *Algorithmic Reporting*
- devices with *Intrinsic Reporting*
- devices with *Algorithmic Reporting* and *Intrinsic Reporting*

If at a real project different types of devices are used it makes no sense to request, that all devices have to support one or two methods of alarm management. If there are a big number of devices without own alarm management (e.g. room controllers with an invariant application, FCU's or other) the alarm management could be handled by another device with own alarm management.

1.3. Common description

To handle alarms in a BMS 3 aspects are important:

- (1) Which data point shall be focussed for generating an alarm?
- (2) In which situation shall that alarm be generated?
- (3) To which recipient shall that alarm be sent?

At *Intrinsic Reporting* aspect (1) is identical with a data point and aspect (2) is determined by the type of that data point (e.g. Analog/Binary/Multistate Input/Output/Value and other).

At *Algorithmic Reporting* aspect (1) is also identical with a data point but for aspect (2) an additional BACnet object type is used: an Event_Enrollment object.

For aspect (3) in both methods of reporting a BACnet object type Notification_Class object is defined.

Generally it is possible to solve all alarm functionalities **either** with *Algorithmic Reporting* **or** with *Intrinsic Reporting*.

That's why it doesn't matter for a B-OWS Operator, if a device supports only one method of reporting. With *Intrinsic Reporting* it is easy to configure a big number of simple situations, in which an alarm shall be generated. But complex situations could be better configured with *Algorithmic Reporting* because it is more flexible.

If a device supports both methods of reporting, the "base load" of all alarm situations shall be configured with *Intrinsic Reporting*. The rest of complex situations could be configured with *Algorithmic Reporting* in a comfortable way.

Within this paper not discussed are basic communication aspects, like

- confirmed or unconfirmed messages,
- recipient entries using the service AddListElement or WriteProperty,
- recipient entries as BACnetObjectIdentifier or BACnetAddress,
- alarm or event reporting

1.4. Used pictures

To demonstrate the configuration of alarm management by a B-OWS in that paper, pictures are used. That picture design is probably different at other vendors, but not the principle of configuration.

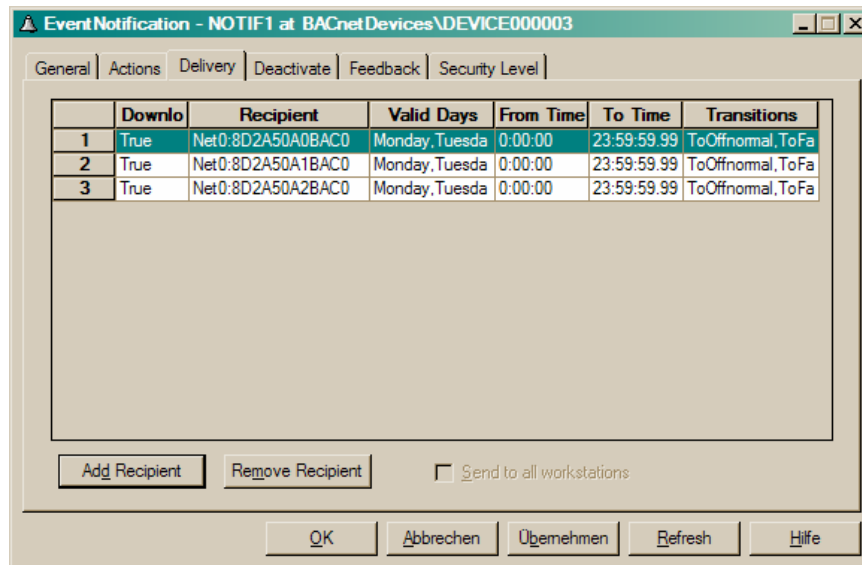
1.5. Last but not least

Many thanks to Norbert Schmalstieg for comments and correction!

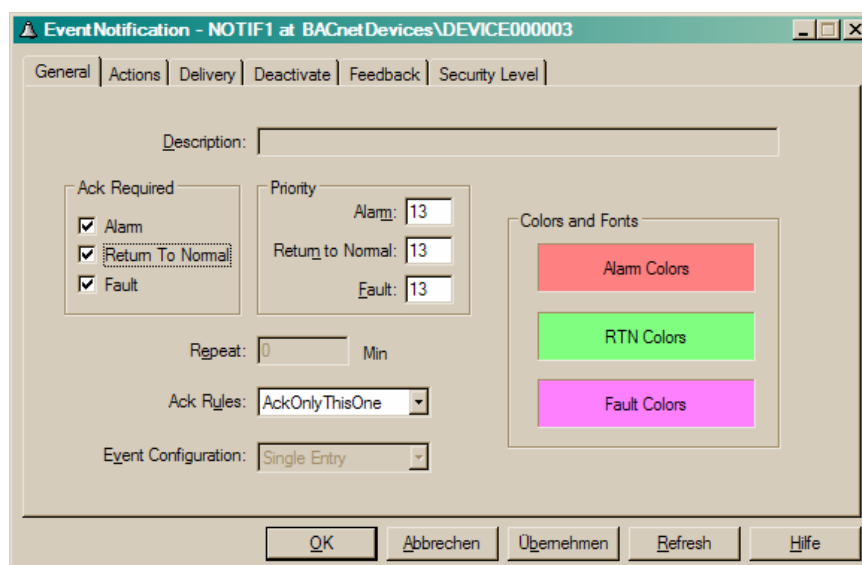
2. Engineering description

2.1. Common features

Both methods of reporting require one or more Notification_Class objects at the device. Among other things that type of BACnet object holds information about **who** (recipient, B-OWS, BACnet Client...) gets the alarm message if an alarm occurs and **when** (at which days and time-windows).



Also it is defined if an initiated alarm requires an acknowledgement or not.

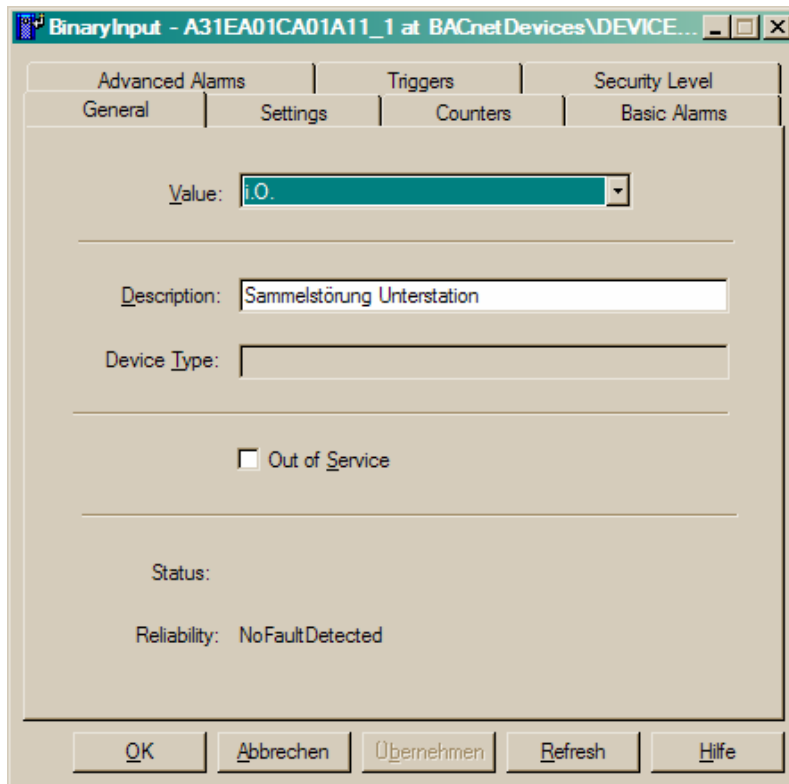


Other issues are local matter or not defined in BACnet standard (print rules, colours at the B-OWS, mail forwarding...). Such issues are not relevant if we compare both methods of reporting.

2.2. Intrinsic Reporting

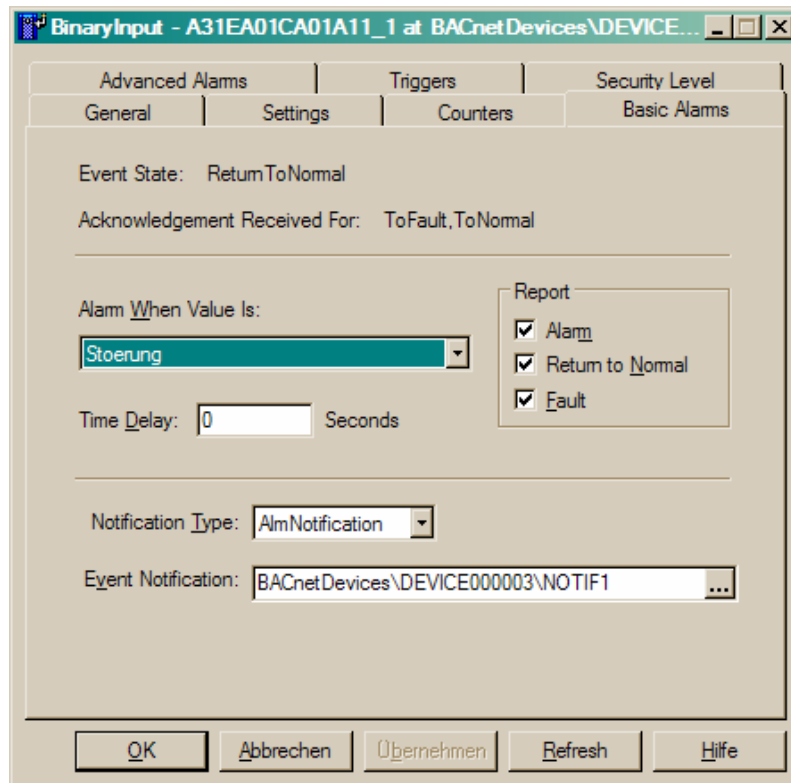
This method of reporting is embedded in a data point object, e.g. a Binary_Input, Analog_Value ... That means that the conditions for generating an alarm are described by own properties of that data point object. There is a link from that data point object to one Notification_Class object. No additional BACnet Object is necessary to handle the alarm.

Sample: a Binary_Input



The screenshot shows a configuration window titled "BinaryInput - A31EA01CA01A11_1 at BACnetDevices\DEVICE...". The window has a tabbed interface with the following tabs: "Advanced Alarms", "Triggers", "Security Level", "General", "Settings", "Counters", and "Basic Alarms". The "General" tab is selected. The "Value:" field is a dropdown menu showing "i.O.". The "Description:" field contains the text "Sammelstörung Unterstation". The "Device Type:" field is empty. There is a checkbox labeled "Out of Service" which is currently unchecked. The "Status:" field is empty. The "Reliability:" field shows "NoFaultDetected". At the bottom of the window, there are five buttons: "OK", "Abbrechen", "Übernehmen", "Refresh", and "Hilfe".

In which case an alarm shall be generated is visible at next picture, also the link to one Notification_Class object (at the picture labelled as Event Notification).



In cases, where the alarming conditions offered by *Intrinsic Reporting* are not sufficient, a combination of both methods is possible. To generate e.g. additional warning-alarms based on the change of Present_Value of an Analog_Input object, this warning aspect could be treated by an *Algorithmic Reporting* extension (see below).

2.3. Algorithmic Reporting

For that method of reporting an additional BACnet object is necessary: an Event_Enrollment object.

The conditions in which case a specific alarm shall be generated are configured at one Event_Enrollment object.

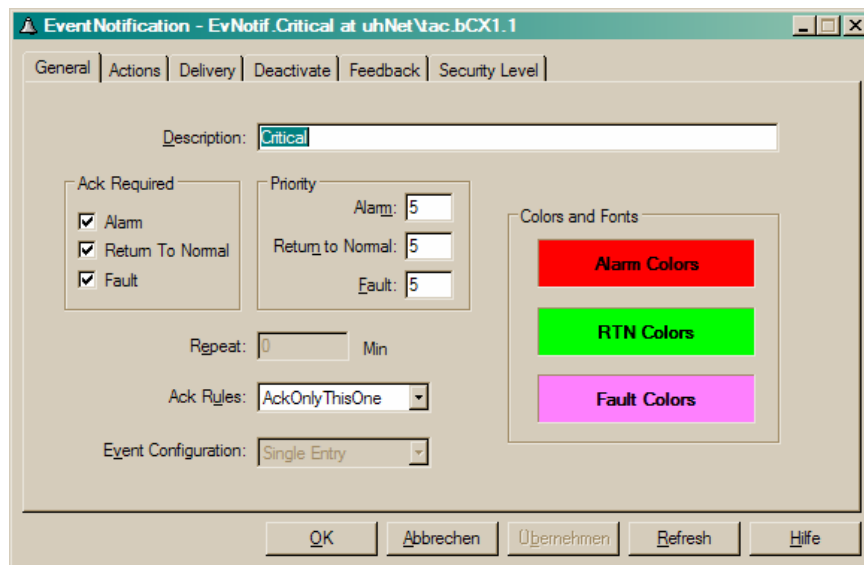
For routing the alarm to recipients there is a link from that Event_Enrollment object to one Notification_Class object.

For multiple technological situations where it could be useful to have an alarm reporting it is possible to provide a separate Event_Enrollment object at a time. That makes *Algorithmic Reporting* very flexible.

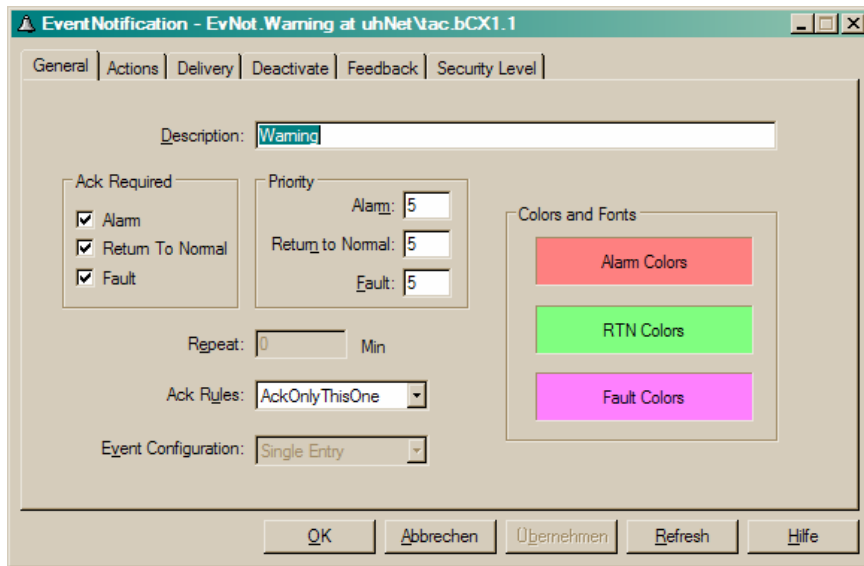
Example: an Analog_Input object is controlled by two different limits (warning, critical)

At minimum one Notification_Class object is necessary. But in most practical cases it makes sense to use two of these objects to make a distinction between distribution of warnings and critical alarms. If at one device other similar alarm situations for similar data point objects shall be configured, separately Event_Enrollment objects necessary but not more Notification_Class objects.

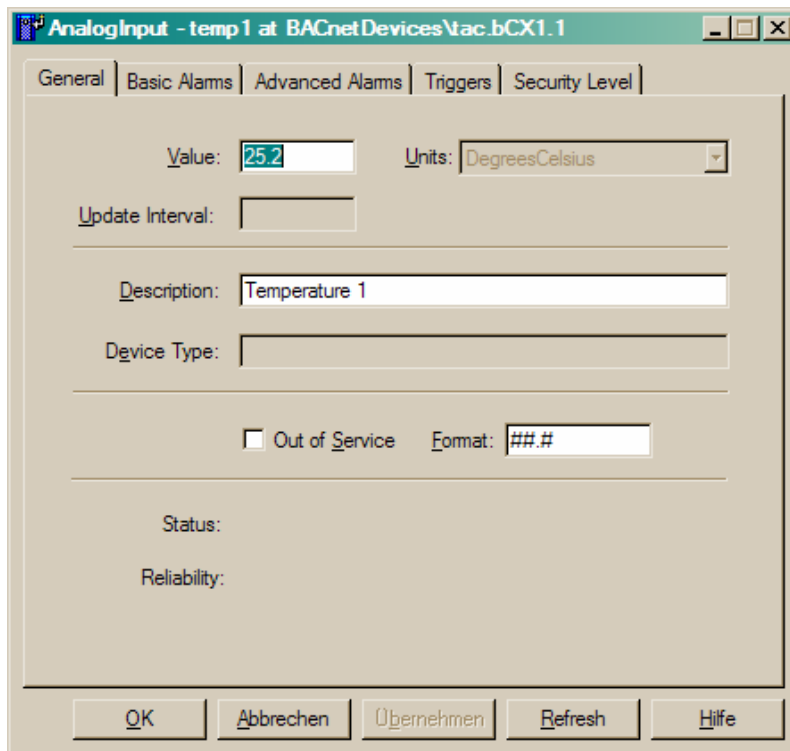
One Notification_Class object for all critical alarms:



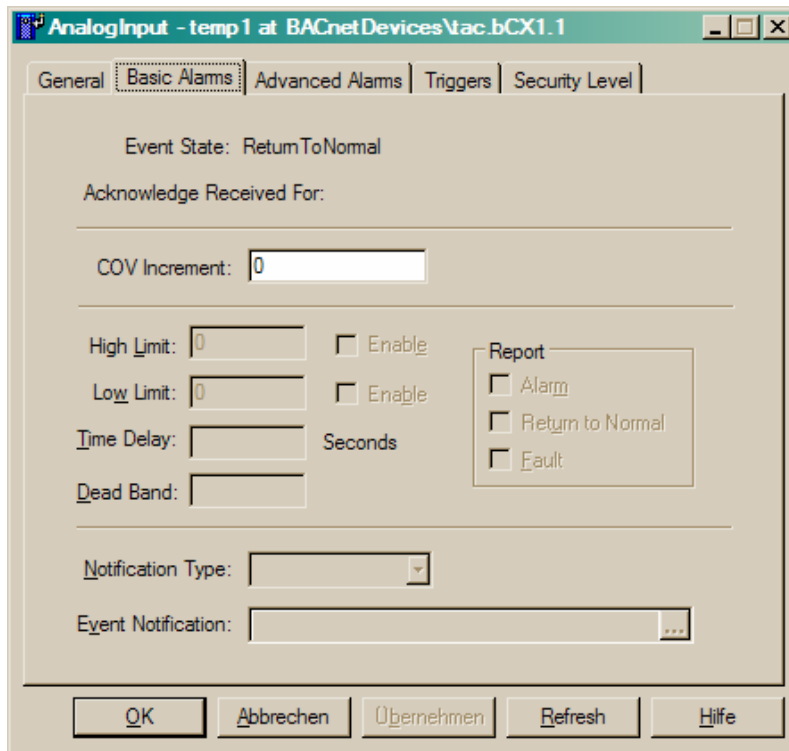
A second Notification_Class object for all warning alarms:



and one Analog_Input object as data point, which has to be observed

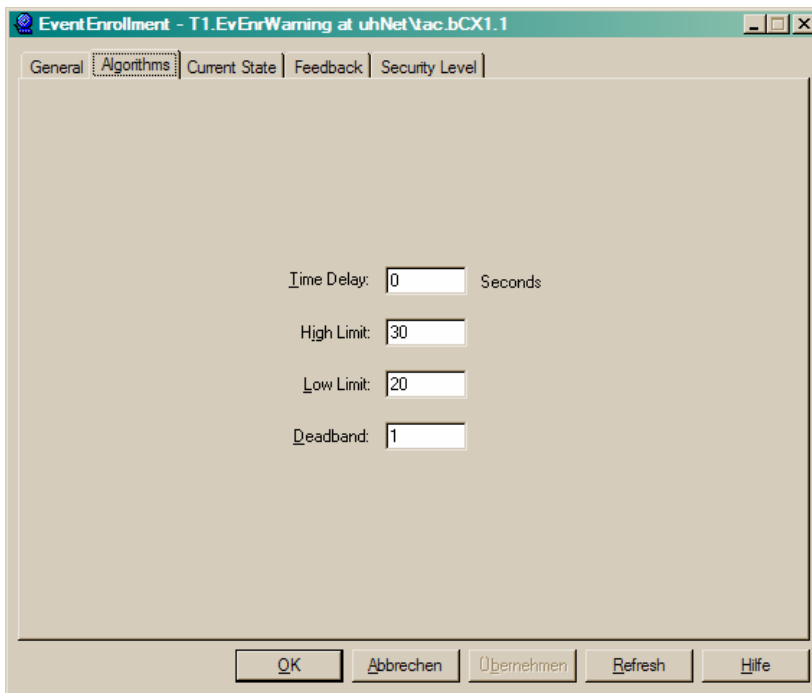
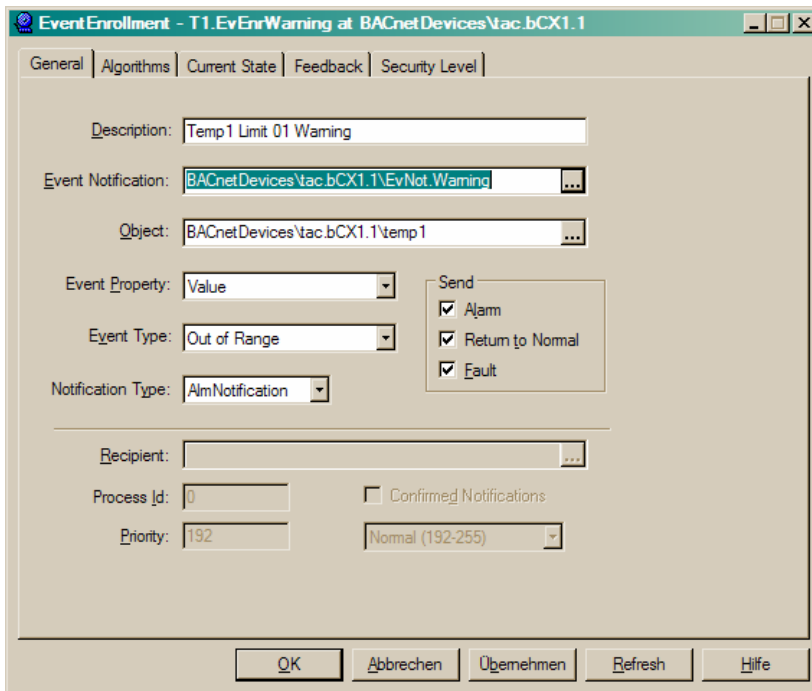


without an own (intrinsic) alarm generation feature.



The relation between a data point object property and an alarm generation is configured in an Event_Enrollment object. In the example two objects of type Event_Enrollment are used (pure *Algorithmic Reporting*):

The first Event_Enrollment object for a warning alarm of the data point “temp1”:



and the second Event_Enrollment object for a critical alarm of the data point “temp1”:

Event Enrollment - T1.EvEnrCritical at BACnetDevices\tac.bCX1.1

General | Algorithms | Current State | Feedback | Security Level

Description: Temp 1 Limit 01 Critical

Event Notification: BACnetDevices\tac.bCX1.1\EvNotif.Critical

Object: BACnetDevices\tac.bCX1.1\temp1

Event Property: Value

Event Type: Out of Range

Notification Type: AlmNotification

Send

- Alarm
- Return to Normal
- Fault

Recipient:

Process Id: 0 Confirmed Notifications

Priority: 192 Normal (192-255)

OK Abbrechen Übernehmen Refresh Hilfe

Event Enrollment - T1.EvEnrCritical at BACnetDevices\tac.bCX1.1

General | Algorithms | Current State | Feedback | Security Level

Time Delay: 0 Seconds

High Limit: 40

Low Limit: 10

Deadband: 1

OK Abbrechen Übernehmen Refresh Hilfe

Both objects observe the same data point and have the same event type (out of range). They are different in their link to the Notification_Class object and the configured limits.

Another possibility would be to configure one of the alarms with *Intrinsic Reporting* (e.g. the warning alarm) and the other one with *Algorithmic Reporting* (e.g. the critical alarm).

3. Comparison

3.1. Engineering

<i>Intrinsic Reporting</i>	<i>Algorithmic Reporting</i>
<p>(+) Easy to configure and to verify for simple alarms:</p> <ul style="list-style-type: none"> - The aspect in which situation an alarm shall be generated is embedded in the data point object. - No additional BACnet objects are necessary for alarm generation. 	<p>(-) For simple alarms:</p> <ul style="list-style-type: none"> - It needs more effort to engineer. - Additional BACnet objects are necessary.
<p>(-) It is impossible to configure directly complex alarms e.g.:</p> <ul style="list-style-type: none"> - If a temperature has to be watched about more than one limits (warning, critical...). - If more than one property (e.g. Present_Value, Out_Of_Service) shall be alarm treated. - If a floating limit shall be alarmed (only a fixed limit is defined, like the Event_Type Out_Of_Range) <p>A small subset of objects is not supported by intrinsic reporting. For example these objects:</p> <ul style="list-style-type: none"> - Device Object Type - Program Object Type - Schedule Object Type <p>To use nevertheless intrinsic reporting, proprietary extensions are necessary.</p>	<p>(+) Very flexible to configure complex alarms:</p> <ul style="list-style-type: none"> - By the BACnet Standard a big number of different Event Type's are predefined {CHANGE_OF_BITSTRING, CHANGE_OF_STATE, CHANGE_OF_VALUE, COMMAND_FAILURE, FLOATING_LIMIT, OUT_OF_RANGE, BUFFER_READY, CHANGE_OF_LIFE_SAFETY, EXTENDED} - All required alarm functions could be solved directly with a large number of Notification_Class objects.

<i>Intrinsic Reporting</i>	<i>Algorithmic Reporting</i>
<p>(-) If <i>Algorithmic Reporting</i> is not supported, for instance mirror data points¹ and proprietary programming are necessary to handle complex alarms. Such engineering is not transparent for a BACnet front end.</p> <p>If <i>Algorithmic Reporting</i> is supported, algorithmic reporting elements have to be added, what results in a mixture of both reporting methods.</p>	<p>(+) All alarm configurations are transparent because it is configured exclusively at Event_Enrollment objects.</p>
<p>(+) After an alarm event it is easy to detect which data point initiated the event, because it is the data point itself.</p>	<p>(-) After an alarm event it needs more effort to detect which data point initiated the event, because the related Event_Enrollment objects contains only a <u>link</u> to the responsible data point.</p>

If a vendor supports both methods of reporting the project engineer could handle simple alarms with *Intrinsic Reporting* and complex alarms with *Algorithmic Reporting*.

If a vendor supports only one of both methods of reporting also the alarm handling is for all technological situations possible where an alarm shall be generated.

If a vendor supports only *Intrinsic Reporting*, additional mirror data point objects could be created, which have the same value at the attribute Present_Value. Every data point object (original and mirror) can be configured with a different intrinsic method.

Also *Intrinsic Reporting* supports only a smaller subset of properties, Object Types and Event Type's than *Algorithmic Reporting*, so that in this kind proprietary extensions could be necessary.

3.2. BACnet front-end user/operator

With both methods of reporting all technological situations where an alarm shall be generated could be handled. If complex situations could not be handled with pure BACnet engineering, proprietary completions shall be possible, see chapter above.

¹ Additional data point object (e.g. Analog_Value object), whose Present_Value is set by a proprietary programming to the Present_Value of a data point object (e.g. Analog_Input object), which represents a physically input.

For a B-OWS or other client it is the same, if a messages is sent by *Intrinsic Reporting* or *Algorithmic Reporting*. A client shall be able to receive messages from both kinds of reporting.

To handle complex alarm with *Intrinsic Reporting* could be a disadvantage for the user at the BACnet front end, because it is not possible to see the relation between original data points and mirror points. In case of *Algorithmic Reporting* that relation is transparent configured at the Event_Enrollment Object.

At the end the effort to engineer a BACnet project is not really interesting for an end user. That effort depends also by vendor specific tools.

3.3. Conclusion

There is no reason to exclude *Algorithmic Reporting* or *Intrinsic Reporting* in bid invitations. Anyway, *Algorithmic Reporting* is more flexible than *Intrinsic Reporting*.

At a bid invitation it is much more important to describe the technological aspects of alarm management instead of the method of reporting (*Intrinsic Reporting* or *Algorithmic Reporting*).

4. Other publications related to both methods of reporting

At [1] (chapter 13) alarm and event services are defined. [1] is without a preference of *Intrinsic Reporting* or *Algorithmic Reporting*.

At [2] (chapter 9) additional rules for implementation of alarm and event services are defined. [2] is without a preference of *Intrinsic Reporting* or *Algorithmic Reporting*.

At [3] (chapter 5.2) we can find a explanation of alarm and event reporting. [3] is without a preference of *Intrinsic Reporting* or *Algorithmic Reporting*.

At [4] (chapter 3.3.1 page 18) without explanation the *Intrinsic Reporting* is preferred. In consideration of the samples and facts at this paper that opinion I could not follow.

At [5] (chapter 5) we can find a comprehensive explanation of alarm and event reporting. [5] is without a preference of *Intrinsic Reporting* or *Algorithmic Reporting*.

5. Literature

[1] ANSI/ASHRAE Standard 135-2004 (BACnet Standard)

[2] BTL Implementers Guide-v16

[3] BACnet_Implementation_Guide_V2.pdf (BACnet Handbuch der BIG-EU)

[4] VDI-GA-BIG-EU-BACnet-Leitfaden V2_5-05-07-22

[5] BACnet Gebäudeautomation 1.4, Hans R. Kranz