



Platform Services

BACnet Alarm Management



APPLICATION NOTE

September 2013

Description: Guide to understanding and setting up BACnet Alarms, visualizing real-time and historical BACnet alarms using AlarmWorX64 Viewer and logging them to SQL database using AlarmWorX64 Logger

OS Requirement: Vista x64/ Windows 7 x64/ Windows 8 x64/ Windows Server 2008 x64/ Windows Server 2008 R2 x64/ Windows Server 2012

General Requirement: Basic GENESIS64 and BACnet knowledge

Introduction

BACnet protocol itself offers all-in-one Alarm management features including algorithmic-based generation of real-time alarms (similar to AlarmWorX64 Server), logging these alarms to internal buffer (similar to AlarmWorX64 Logger). It is important to note that all operations regarding BACnet Alarms are running inside BACnet device. GENESIS64 acts as a client which configures properties of separate objects, as well as a client which is capable of visualizing (and do further processing) real-time and historical alarms. This Application Note covers fundamental knowledge about BACnet Alarms and how to setup BACnet Alarms. In order to properly follow the example in this Application note, you would need to have the BACnet device discovered and scanned – to do so please follow *Platform Services - BACnet Network and Device discovery* Application Note.

BACnet Alarms infrastructure

In BACnet Alarms, it is fundamental to know that there's an object type called Notification Class. Instance of this object type is used as a 'gateway', which reports new Real-Time alarms. The client; which in our case is GENESIS64, specifically AlarmWorX64 Viewer or Logger via BACnet Runtime; subscribes to the selected Notification Class. When a new event occurs, the Notification Class sends a notification to BACnet Runtime. BACnet Runtime knows who is the owner of the notification and forwards it accordingly to the proper module. BACnet device can contain almost any amount of Notification Class objects (it depends on the vendor of the device, CPU speed, and device memory). Shown below is the infrastructure for the BACnet Alarms:

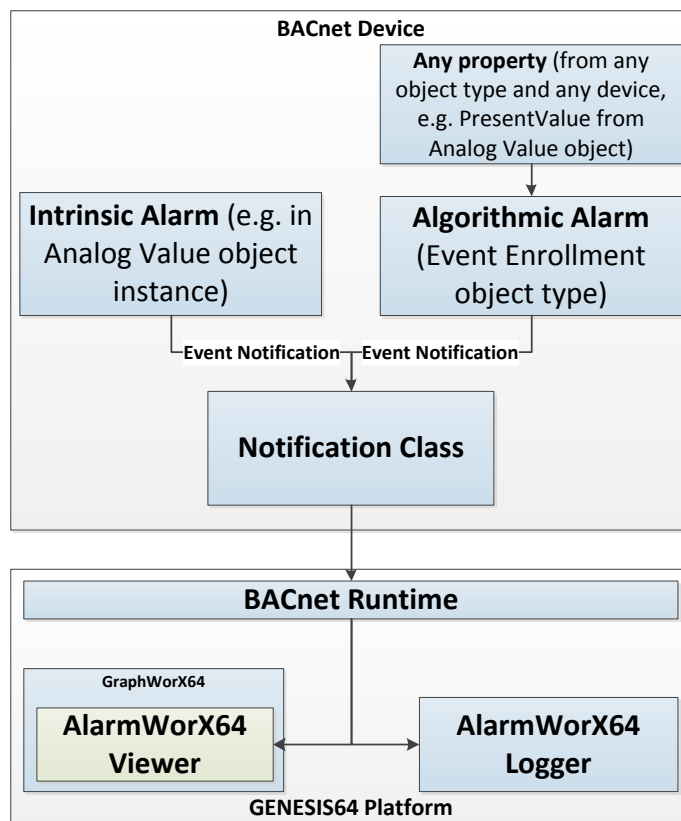


Figure 1 – BACnet Alarms/Event Infrastructure

Intrinsic (Real-time) Alarms

Some object types (Analog Input/Output/Value, Binary Input/Output/Value) do support intrinsic alarming. The given object instance can be configured accordingly to generate event notification based on object-specific rules (Analog object types are based on Low Limit, Normal and High Limit notifications, Binary object types can throw event notifications based on Binary object state). It is not required to connect an appropriate property (PresentValue in this case) with any Event Enrollment (as is the case with Algorithmic alarming).

Algorithmic (Real-time) Alarms – Event Enrollment object type

Event Enrollments are object types that allow you to reference any property from any object instance which is located in any BACnet device as a datasource. This then allows you to choose an algorithm for evaluating alarm conditions. If these specific conditions are fulfilled, the event notification is sent to the selected Notification Class Object, which is then distributed to the subscribers.



Platform Services

BACnet Alarm Management



APPLICATION NOTE

September 2013

Historical Alarms – Event Log object type

If the BACnet device supports Event Logs, it is capable of storing specific amount of BACnet events from a given Notification Class. AlarmWorX64 Viewer can then be used for visualization and navigation through the buffer.

Event States

In BACnet universe there are three standardized event states:

- Normal – value is within specified boundaries
- Off Normal – value is out of specified boundaries
- Fault – some problem occurred with correct event evaluation (e.g. sensor is malfunctioned, thus it cannot reliably provide a value)

In the same manner transitions from one state to given state have 3 different events:

- To Normal
- To Off Normal
- To Fault

Creating Notification Class

In order to create a Notification Class please follow these steps:

1. Launch Workbench Silverlight.
2. Switch to the BACnet provider.
3. In the tree, find desired device.
4. Right-click on it and from the context menu, select 'Add Object'.
5. Under the *Database* tab, choose 'Notification Class' in the drop-down list for *Object Type*.
6. By default a form will be opened – fill it accordingly (you can specify whether the given event transition requires acknowledgement (Ack Required checkbox) and what priority it has). Click on 'Apply' to create a Notification Class.

One *Device* can contain multiple Notification Classes. In comparison with AlarmWorX64 Server, the Notification Classes can be considered as simple Areas – they can be used to distinguish between alarms/events from specific set of alarm/event sources. User/operator can then subscribe to specific notification class's to get a given subset of alarms/events.

Example: You have two rooms, each with set of BACnet-enabled sensors and detectors (temperature, humidity, light intensity, etc.), all these sensor are connected to one central BACnet device.

The alarms/events from sensors/detectors located in Room A will be notified through the Notification Class (NC) with an instance 0 and sensors located in Room B through NC 1. This will allow you to separate and clearly distinguish events raised from separate rooms.

Setting up intrinsic alarm

In order to have intrinsic alarming to work as expected it is required to have Notification Class's already created in the device (you have to know its instance number). Please note that not all objects are intrinsic alarming-enabled. To set it up on a given object instance, in our case Analog Value, follow these steps (Use ICONICS BACnet simulator running on separate machine for your practice):

1. Launch Workbench Silverlight.
2. Switch to the BACnet provider.
3. In the tree, find desired device.
4. Under given device find an object that supports intrinsic alarming (In our case please select **Analog Value** instance)
5. The Runtime tab for given object will be opened and you be able to read and write values of all properties, now you have to write specific values to the following properties (under BACnet / Intrinsic Alarming section):
 - a. **Notification Class** – instance number of NC you want to use for alarm notifications
 - b. **Time Delay** – how long (in seconds) the value needs to stay over/under the specified limit before an Event is thrown (for instant notification set to 0)
 - c. **Event Enable** – which event transitions will be notified (check at least *To Normal* and *To Off Normal*)
 - d. **Limit Enabled** – this property is Analog object type specific – specify which limit should be monitored in order to throw events (in our case check both – High Limit Enabled and Low Limit Enabled)
 - e. **High Limit** – specify value of High Limit (set 300)
 - f. **Low Limit** – specify value of Low Limit (set 100)



Platform Services

BACnet Alarm Management



APPLICATION NOTE

September 2013

Please note that other Object Types can have different Intrinsic Alarm settings, depending on the object (e.g. Binary object types won't have Low and High Limits). But Notification Class, Time Delay and Event Enable properties should be used for all.

Setting up algorithmic alarm

In order to have algorithmic alarming to work as expected it is required to have the Notification Class and the target property (of given object type instance) already created in the device. As mentioned earlier, algorithmic alarming uses Event Enrollment object type to configure alarm/event source property. In this case, the algorithm is used for the alarm/event evaluation and the Notification Class is used as an alarm/event 'reporter'. To set it up please follow these steps:

1. Launch Workbench Silverlight.
2. Switch to the BACnet provider.
3. In the tree, find desired device.
4. Right-click it and, from context menu, select the 'Add Object' item.
5. Choose 'Event Enrollment' in the drop-down list and confirm by clicking OK (don't have 'Create Empty Object' checkbox checked).
6. In the opened form, you have to fill in following:
 - a. **Notification Class** - instance number of NC you want to use for alarm notifications.
 - b. **Event Type** – select **Algorithm type** used for evaluation. Please refer to BACnet specification for details (brief information about specific Event Types shown in Figure 2) – in our case please set 'Out Of Range' Event Type
 - c. **Event Parameters** - By clicking '...' button next to the 'Event Parameters' text box, open the dialog specific for 'Out Of Range' event type in order to set its parameters; it uses the same parameters as described in 'Setting up intrinsic alarm' section, please refer for details.
 - d. **Referenced Object** – By clicking '...' button next to the 'Referenced Object' text box, open the dialog to browse for **referenced property** whose value is used as the data source or set the Referenced Object values manually using the combo boxes and numeric fields.
7. Confirm the Event Enrollment object creation by clicking on the Apply button.

Event type	Event State	Event Parameters
Change Of Bitstring	Normal Off Normal	Time Delay Bitmask List of Bitstring Values
Change Of State	Normal Off Normal	Time Delay List Of Values
Change Of Value	Normal Off Normal	Time Delay Bitmask Referenced Property Increment
Command Failure	Normal Off Normal	Time Delay Feedback Property Reference
Floating Limit	Normal High Limit Low Limit	Time Delay Setpoint Reference Low Diff Limit High Diff Limit Deadband
Out Of Range	Normal High Limit Low Limit	Time Delay Low Limit High Limit Deadband
Unsigned Range	Normal High Limit Low Limit	Time Delay Low Limit High Limit
Change Of Life Safety	Normal Off Normal Life Safety Alarm	Time Delay List Of Alarm Values List Of Life Safety Alarm Values Mode Property Reference

Figure 2 – Event Type list

Subscribing AlarmWorX64 Viewer to Notification Class object (Real-Time alarms)

To display BACnet Alarms, do the following:

1. Launch GraphWorX64.
2. Insert the AlarmWorX64 Viewer control by selecting it from the 'Controls' tab and drawing a rectangle in the canvas to place an AlarmWorX64 Viewer.
3. Double-click the displayed viewer control in order to open its properties and settings.



Platform Services

BACnet Alarm Management



APPLICATION NOTE

September 2013

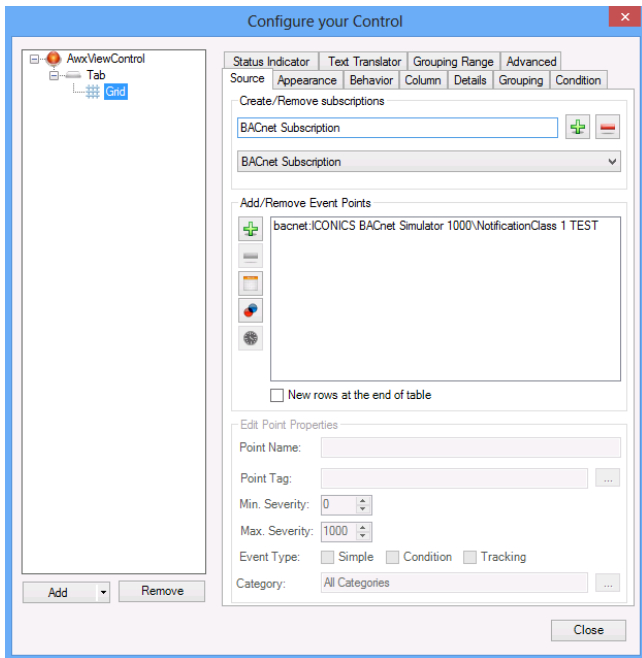


Figure 3 - Configuring BACnet Alarm Views

4. In the opened dialog, select the 'Grid' tree item.
 5. Now click the button with the '+' sign in the 'Create/Remove Subscriptions' section and choose 'Real Time Subscription' from the opened context menu.
 6. Select the 'Local Alarm Server' listbox item in the 'Add/Remove Event Points' section and click the '-' sign in the same section to remove the existing subscription.
 7. Now press the '+' button in the 'Add/Remove Event Points' section. This will open the Data Browser.
 8. In the Data Browser's 'Shortcuts' pane, on the left side, choose 'Home' and you'll be switched to the root of the Data Browser.
 9. Select BACnet from the list of folders.
 10. Choose (by double-clicking) your desired BACnet device from the list to see its provided objects (it has to be scanned prior to seeing the device in the list).
 11. In the list of objects, find your desired Notification Class that would provide notifications about alarms. These objects are marked with the attribute 'Subscribe', which is visible in the 'Attribute' column. You can sort the list by clicking the column header.
 12. Confirm the selection by clicking 'OK' in the Data Browser. This will add a Notification Class subscription to the AlarmWorX64 Viewer.
 13. After subscribing to all desired Notification Classes (repeat steps 7 to 12 to subscribe to more NCs), you can close the dialog window for AlarmWorX64 Viewer's configuration,
 14. Now switch GraphWorX64 to Runtime mode in order to receive and see alarms provided by your subscribed Notification Classes.
- ### Subscribing AlarmWorX64 Viewer to Event Log object (Historical alarms)
- To display BACnet Historical Alarms (Event Logs,) do the following:
1. Launch GraphWorX64.
 2. Insert the AlarmWorX64 Viewer control by selecting it from the 'Controls' tab and drawing a rectangle in the canvas to place a AlarmWorX64 Viewer.
 3. Double-click the displayed viewer control in order to open its properties and settings.
 4. In the opened dialog, select the 'Grid' tree item.
 5. Now click the button with the '+' sign in the 'Create/Remove Subscriptions' section and choose 'Historical Subscription' from opened the context menu.
 6. Select the 'Local Alarm Server' list box item in the 'Add/Remove Event Points' section and click the '-' sign in the same section to remove the existing subscription.
 7. Now press the '+' button in the 'Add/Remove Event Points' section. This will open the Data Browser.
 8. In the Data Browser's 'Shortcuts' pane, on the left side, choose 'Home' and you'll be switched to the root of the Data Browser.
 9. Select BACnet from the list of folders.
 10. Choose (by double-clicking) your desired BACnet device from the list to see its provided objects.
 11. In the list of objects, find your desired Event Log object providing historical alarms. These objects are marked with the attribute 'HistRead', which is visible in the 'Attribute' column. You can sort the list by clicking the column header.
 12. Confirm your selection by clicking 'OK' in the Data Browser. This will add an Event Log historical subscription to the AlarmWorX64 Viewer.
 13. After listing all desired Event Log objects (repeat steps 7 to 12), you can close the dialog window for AlarmWorX64 Viewer's configuration.
 14. Now switch GraphWorX64 to Runtime mode in order to receive and see historical alarms provided by subscribed Event Log object(s).
- Note: Current implementation supports getting data only by polling method: AlarmWorX64 Viewer asks for new data. This can be forced manually by selecting the AWX Viewer in runtime mode to bring up the ribbon. Click on the *Refresh* button to force AWX Viewer to get new data. Or, you can set a period and



Platform Services

BACnet Alarm Management



APPLICATION NOTE

September 2013

enable automatic polling in AlarmWorX64 Viewer's configuration dialog: via the 'Behavior' tab, click the checkbox and fill in the input field 'Automatically update Historical data after'.

Logging BACnet Alarms to SQL Database using AlarmWorX64 Logger

You can log BACnet Alarms using AlarmWorX64 Logger simply by selecting a Notification Class object instance (using BACnet Point Manager. This path will have the format of **bacnet:<pathToProperty>**) as a point for Alarm Subscription:

1. Launch Workbench Silverlight.
2. Switch to the AlarmWorX64 Logger provider.
3. In the tree, find the desired logging Configuration.
4. Under 'Alarm Subscription' click on the Add button – the Data Browser will be opened
5. Browse to Home -> BACnet and choose your desired BACnet device from the list to see its provided objects (it has to be scanned before to see the device in the list).
6. In the list of objects, find your desired Notification Class that would provide notifications about alarms. These objects are marked with the attribute 'Subscribe', which is visible in the 'Attribute' column. You can sort the list by clicking the column header.
7. Confirm selection by clicking 'OK' in the Data Browser. This will add a Notification Class subscription to the AlarmWorX64 Logger Configuration.
8. Start the AlarmWorX64 Logger service to log newly raised events (presuming that your target database is properly set).

Setting up AlarmWorX64 Server to generate alarms based on values from BACnet properties

It is possible to have AlarmWorX64 Server trigger alarms based on values coming from BACnet devices. The process of setting up the rules for triggering is based on Limit, Digital, Deviation, etc. The alarm type remains the same. The only thing you have to do is connect a BACnet property using the BACnet GenClient plugin -In the Data Browser, browse to the following location: Home -> OPC Classic Servers -> My Server -> BACnet, there you will see same address space as you would be using the BACnet Point Manager (tag name format is **@bacnet64:<pathToProperty>.Value**). You have to use GenClient BACnet Plugin due to the compatibility reasons. Everything else corresponds with the standard setting of an Alarm Tag in the AlarmWorX64 Server provider.

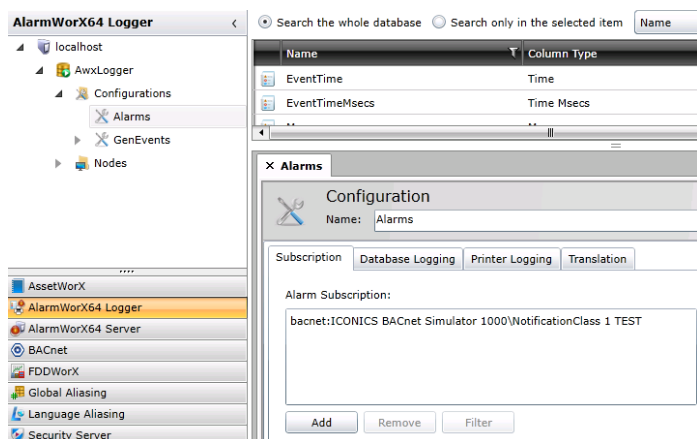


Figure 4 – AlarmWorX64 Logger setup to log BACnet Alarms