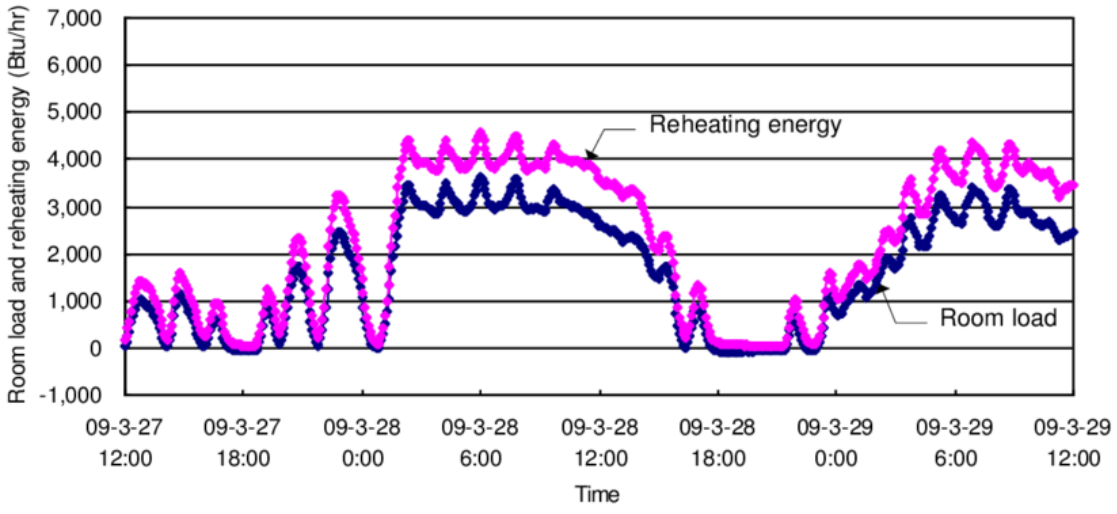
bacnet experts

ASHRAE BACnet®

Trend Log and Trend Log Multiple Objects

David Fisher

13-Jul-2021



TUTORIAL

Contents

Trend Log and Trend Log Multiple Objects

13-Jul-2021

David Fisher

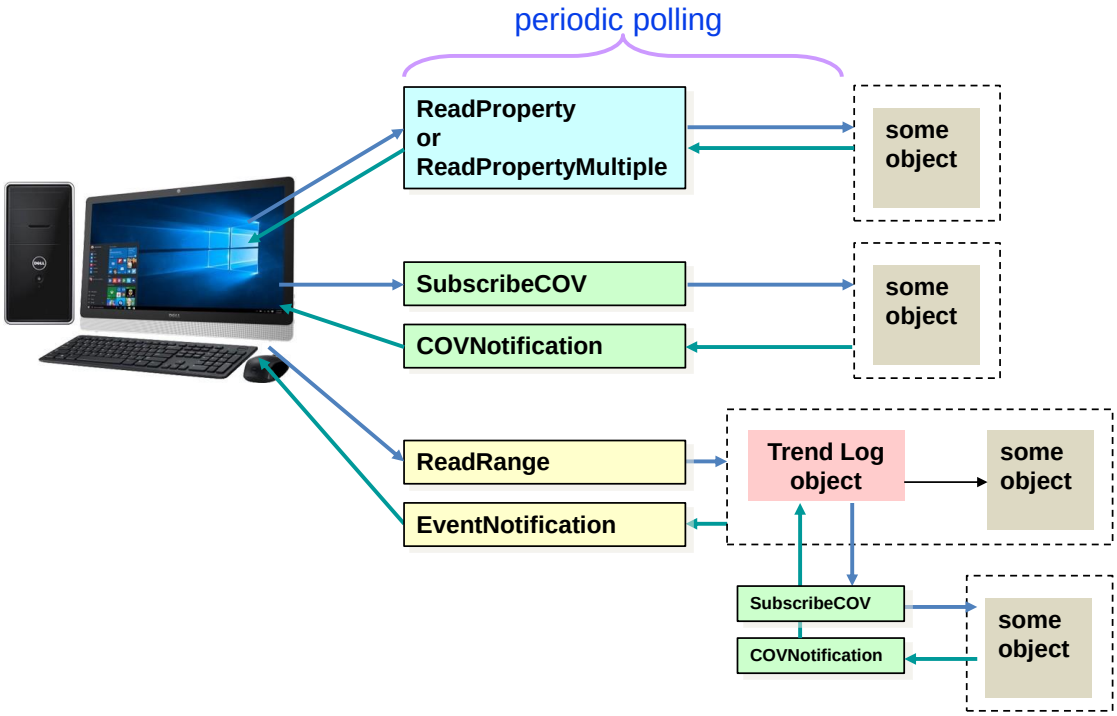
Introduction

The purpose of *Trending* is to record information about something over a period of time. Typically, one or more datapoints are sampled together at the same time interval repeatedly and the results are stored for later use. To handle situations like missing data due to network traffic, power outages and so forth, the sampled data usually includes a time-stamp with each sample.

The collection of samples for a single datapoint is called a Trend Log (TL).

There are two separate but related problems to consider with using BACnet for Trending applications. Somehow, the trend data must be gathered in the first place. This may or may not involve using BACnet services to get the data. Once sampled, the trend information must be saved in a form that is accessible using BACnet. Usually, the saved TL(s) must be transferred from the place where they have been recorded, to another device so that they can be used.

There are multiple ways to do trending using BACnet. This diagram shows the big picture:



In most cases, the ultimate destination for Trend data is a workstation. Human operators may want to examine trend data for many reasons or archive it for later use.

It is possible with some workstation software for BACnet to use the workstation itself as a data gatherer. In these cases, the requesting of datapoints and the storage of the trend log is driven by the workstation software directly.

There are two schemes typically used to do the data sampling: *periodic polling* and *change-of-value (COV)* reporting.

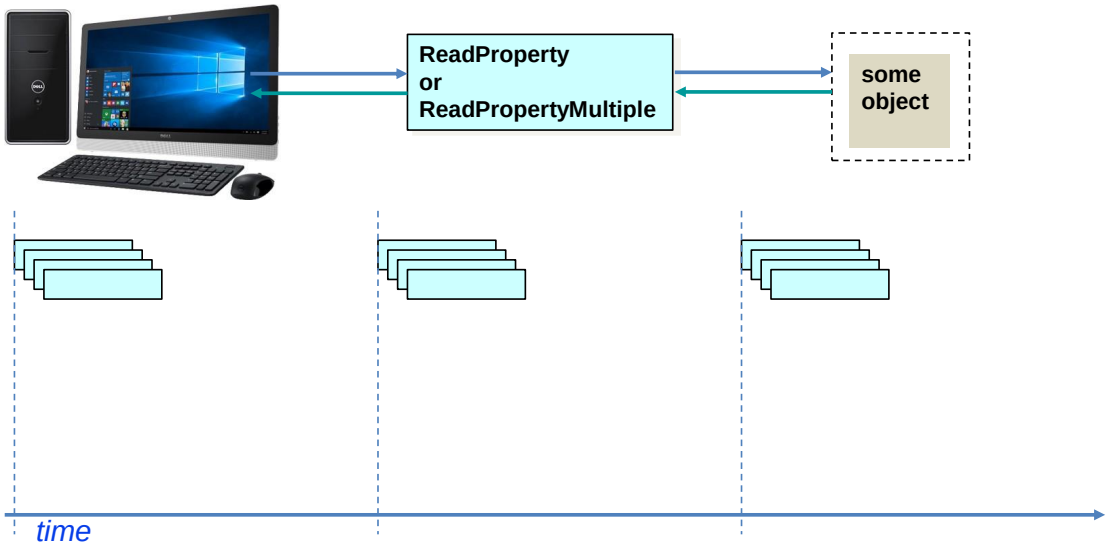
With periodic polling, the workstation uses ReadProperty or ReadPropertyMultiple services to read properties of objects in one or more BACnet devices. Every sample interval, the workstation initiates some number of these service requests to gather samples. If the sampling interval is too short, then this can be a problem because it may not be possible to gather all the data quickly enough. When a lot of this kind of trending is taking place, a lot of additional network traffic is also created.

Change-of-value (COV) reporting allows the workstation to request that the various BACnet devices containing the desired datapoint(s) only send data when it changes by some amount or more. This significantly reduces traffic, but it requires the BACnet devices to support COVNotification and subscription. The drawbacks are that many devices do not support COV subscriptions or may be limited in the number of simultaneous subscriptions they can provide. As a result, any COV-based BACnet client will have to fall-back to using ReadProperty if subscriptions cannot be created.

An alternative to centralized trending like this is to have one or more BACnet devices perform the trend sampling, and then periodically collect the “bag of samples”. Although there are proprietary ways to do that with BACnet, the standard way is to use TL objects to store the samples, and the ReadRange service to collect the saved sample data from the TL objects.

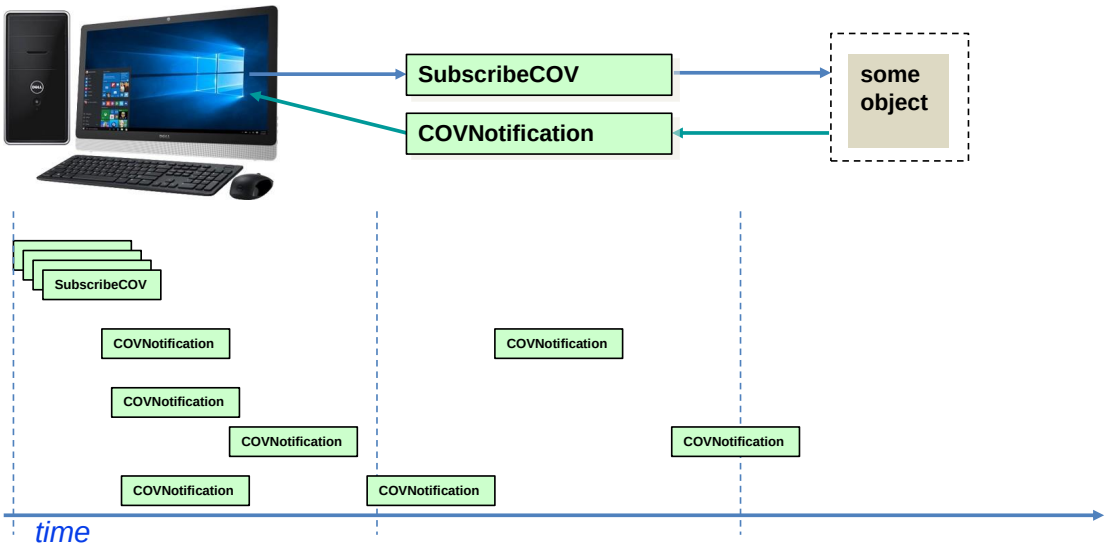
Polled Sampling

Here’s an example showing four values being trended at some interval.



Polled Sampling is very simple and reliable. The "data gatherer" uses ReadProperty or ReadPropertyMultiple requests at periodic intervals to gather data points. It is reliable in the sense that the client always "knows" if it can get the data, and if it can't then what the reason is. The problem is that this technique does not scale well. As the number of points to be gathered increases, the overall burden on the polled devices, the gatherer and the network itself increases. It is easily possible to overwhelm all three with this approach.

COV Sampling



When COV is used, initially subscriptions are issued for the points to be trended. Sometime later at random intervals COVNotifications are received to indicate changes of value. The trending application program caches these asynchronously received values and uses the most recent value for the time-stamped trend sample. Since most datapoints do not change very frequently, this approach saves network bandwidth by reducing the number of messages to convey only actual changes.

But COV is still a burden for the client because there are many reasons why it can fail. First, the target device may not even support COV subscriptions. Second, all of the available subscriptions might be in-use when the client tries to (re)subscribe. Third, the target device may reset or power cycle during use, requiring resubscription, etc. In any case the client still needs to be able to support polled sampling as a fall-back strategy.

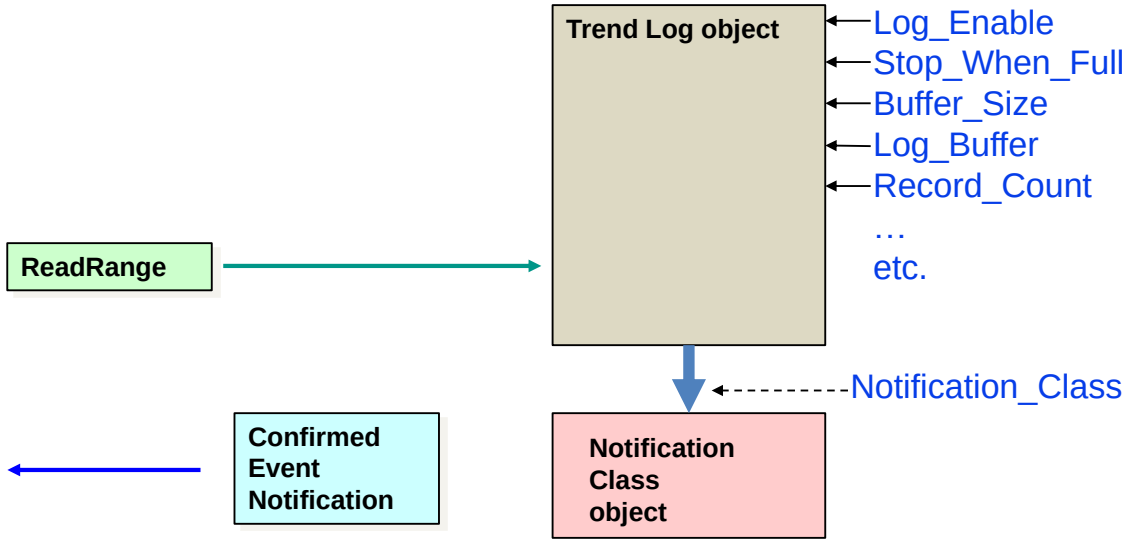
Trend Log Objects

Overall, a more scalable strategy is to take advantage of the many controllers that are part of the building automation system (BAS) and to delegate the task of gathering trend data to one or more of these controllers. The BACnet Trend Log object exists for just this purpose.

In many cases its just not practical to do centralized sampling for any significant number of trends. Even when possible, it's clearly more efficient use of network resources to push the sampling of trend data out as close to the source of the data as possible. BACnet abstracts the representation of this process into an object called the Trend Log (TL) object. Although you can think of the Trend Log object as a container for decentrally-sampled trend data, it is really a formalized way of thinking about how BACnet devices can convey how they are set up to do trending as well as how their collected data may be accessed.

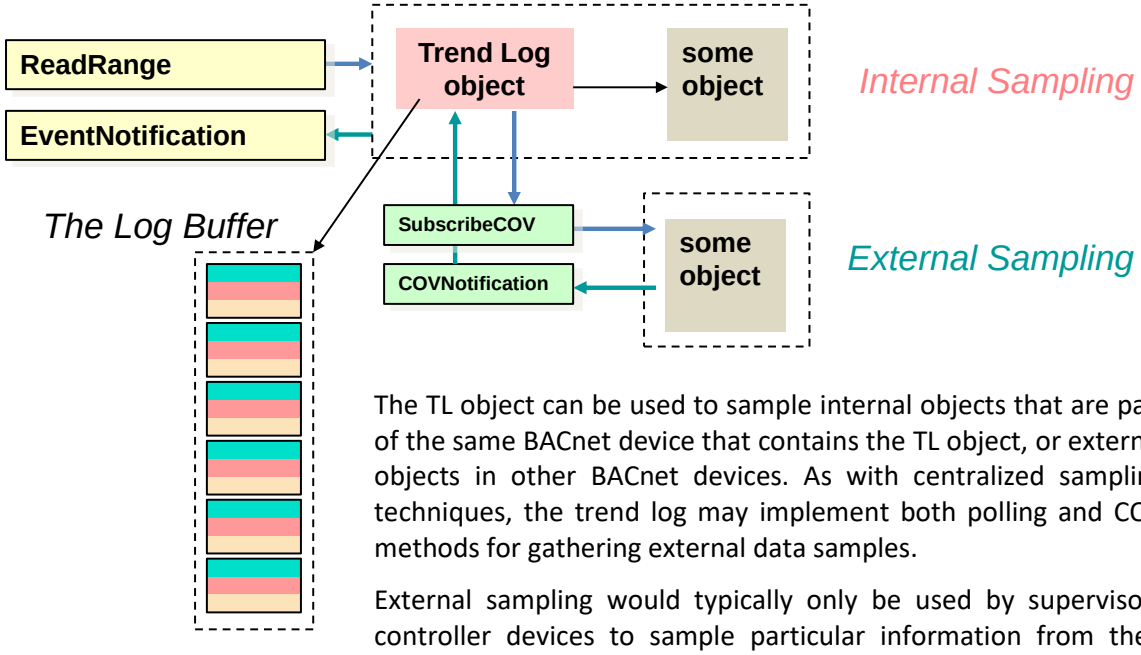
In the big picture, some BACnet device performs the trend sampling and saves the resulting data samples. Eventually some other BACnet device comes along and asks to collect some number of these samples so that the sampling device can forget about them. The setup and control of the trending process takes place using properties of the TL object. Collection of samples uses the

ReadRange service. Ideally the collection of samples is read more frequently than the sampler device can accumulate enough samples that it runs out of room to save them!



The TL object has additional sophistication in that it can initiate Confirmed Event Notifications when its internal storage has become filled, or close to filled, so that collection device(s) can be alerted of the need to empty the trended sample information. This generates a special kind of "alarm."

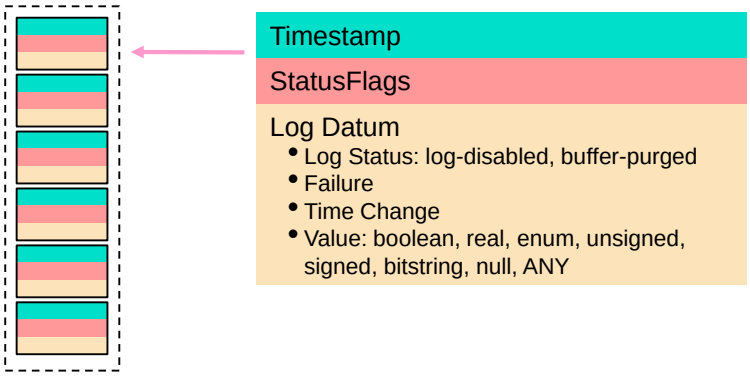
The Notification_Class property indirectly refers to a Notification Class object in the same device by virtue of the class number. The Notification Class object may be shared among several TL objects. Its purpose is to determine which BACnet devices should be recipients of the "need to collect" notification.



The TL object can be used to sample internal objects that are part of the same BACnet device that contains the TL object, or external objects in other BACnet devices. As with centralized sampling techniques, the trend log may implement both polling and COV methods for gathering external data samples.

External sampling would typically only be used by supervisory controller devices to sample particular information from their constituent devices.

Regardless of how it is sampled, the resulting collections of data can be accessed and manipulated through the Trend Log object.



The Log Buffer is a storage area that is part of the TL object which stores the trended sample data. Each timestamped entry is called a *record*.

Each of the log records contains a Timestamp which says when the record was entered in the log, StatusFlags which represent the Status_Flags property of the monitored object (if available), and the Log Datum which is usually the logged value (or sampled data).

In addition to sample data, which may be any of several simple datatypes, some log records are special because they don't hold data at all. These special log records keep a history of certain important kinds of events that may occur. For example, whenever the Log_Enable is changed, an entry is made in the log. If Record_count is written with a zero, then a buffer-purged record is made in the log. If any sampling causes an error, then a failure record is entered in lieu of the sample. If the device's time or date are changed then a time-change record is entered.

So, the Log_Buffer really is a running history of everything that has happened to the TL object!

As a rule, log records are created either as a result of periodic polling, or a COV notification. However, it is also possible for log records to be created by a "trigger" which is a special property of the Trend Log object that when written causes a new record to be created.

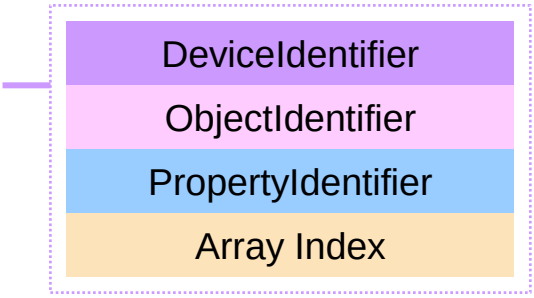
The records in the Log Buffer have some "not obvious" characteristics. When periodic polling is being used, one would expect sequential log records to be time-stamped with times that are spaced apart by some fixed sample interval. Barring failures and other events (like starting or stopping the TL), if the TL has been running for N time periods we should see N records in the log. If COV is being used for data gathering, an internal place-holder bucket retains the most recent COV value so that this can be "periodically sampled". You can also configure the log buffer to retain only COV values, in other words only value changes. In the same way that COV reduces network bandwidth when gathering, you can reduce the log buffer size by retaining only records that represent real changed values even when periodic polling is being used.

The TL object provides for a lot of flexibility in choosing the size for the Log Buffer. It is allowed to have a fixed-size Log Buffer that has room for N entries. In such cases, when the Log Buffer becomes "full" the next subsequent sample can "wrap around" and overwrite the oldest sample. Or, alternatively, when this happens the TL can be automatically stopped. (STOP_WHEN_FULL==true). The Log Buffer may also allow itself to grow up to the total available space.

Going back to one of our original objectives with TL objects, we wanted to distribute the burden of gathering samples into devices that contain the objects to be sampled, or supervisory

controllers of those devices. In such cases, every so often the original client (who wanted the samples) can go to each TL object and collect the samples using the ReadRange service. Of course, the client can do this periodically or according to its own algorithm. However, the TL object provides a mechanism to help with this. The Notification_Threshold property says how many records must be in the Log Buffer before the device issues a BUFFER_READY event notification. This is a special kind of (usually confirmed) EventNotification that indicates a normal-to-normal transition. Clients can listen for this type of notification and use it to trigger the sending of ReadRange requests to "unload" the TL's Log Buffer.

How does the TL know what object property is to be sampled? The Log_DeviceObjectProperty property of the TL object is a normal BACnetDeviceObjectPropertyReference (DOP). This special datatype allows you to specify an *ObjectIdentifier* of the object to be sampled; a *PropertyIdentifier* of the property of that object to be sampled (usually this is Present_Value). In some cases, you may be sampling a BACnet array property and in those cases you may only be interested in one particular slot in the array, in which case the DOP will include an *ArrayIndex*. Earlier we talked about the fact that some devices, when they implement TL objects, are only *internal sampling* meaning that only other objects in the same device as the device that owns the Trend Log can be sampled. However, if the TL device has client capabilities, then the DOP may also include the DeviceIdentifier of a target device (external) whose object property should be sampled.



Trend Log Multiple Objects

Some applications require more than one object property to be sampled at the same interval. These situations are exactly the same as TL objects, except that there is more than one DOP, and each log record may need to contain multiple values. The Trend Log Multiple (TLM) object serves this purpose, but is otherwise the same general idea as TL.

Important Trend Log Object Properties

The TL object has a number of unique properties that we'll discuss here.

Enable	true to enable logging
Start_Time, Stop_Time	the (inclusive) range of date/time during which logging is running
Log_DeviceObjectProperty	the DOP that this TL is sampling
Log_Interval	periodic interval (in hundredths of seconds) for sampling
COV_Resubscription_Interval	when COV subscribing is being used this is the periodic interval in seconds after which the COV subscription will be resubscribed. The actual subscription requests twice this lifetime.
Client_COV_Increment	when COV is active, if this increment is non-NULL then it is used in place of the COV_Increment of the referenced object in order to determine when changes have occurred.
Stop_When_Full	when true and the Log Buffer becomes full, the logging is automatically stopped. When false and the log does not have the capability to automatically expand its size, the log will wrap around to the oldest sample instead.
Buffer_Size	the size of Log Buffer in terms of the maximum number of records it can hold. This property is not required to be writable, but if the object allows writing to Buffer_Size (as a means of expanding/contracting the log) then writing is not allowed unless Enable is false.
Record_Count	the count of records in the Log Buffer at the moment. Optionally, the object may allow writing of zero to this property which has the effect of deleting all existing records and adding a single "buffer cleared" event to the log.
Total_Record_Count	the total number of records added to the log since "creation." The standard does not define what "creation" means, but it does say that when the Total_Record_Count exceeds 2^{32}-1, the count wraps around to 1.
Logging_Type	polling, COV or triggered
Align_Intervals	when true, periodic logging is aligned to the appropriate interval second, minute, hour, day
Interval_Offset	when Align_Intervals is true, this is the offset in hundredths of a second from the beginning of a log interval until logging begins.
Trigger	when this property is written with a true value and Logging_Type is triggered it triggers logging to occur and Trigger remains true until the logging completes.

Notification_Threshold	the number of records that must be in the Log Buffer to trigger a BUFFER_READY event notification.
Records_Since_Notification	the number of log records collected since the previous notification, or since the beginning of logging if no previous notification has occurred.
Last_Notify_Record	the record count as of the last notification.

Data Values

Since the DOP can point to arbitrary object properties, it is possible for TL objects to sample many different possible datatypes. This is a subtle requirement however. The standard does not say that TL must be able to support *any possible DOP*. The TL implementer decides which object types it is willing to support and therefore may reject some DOPs that are filled-in with references to properties whose datatypes are not supported.

The Log Buffer contains a list of BACnetLogRecords. Each record's "log-datum" allows for a choice between a number of different datatyped values Boolean, Real, Enumerated, Unsigned, Integer, Bitstring, Null. It can also hold TL-specific datatypes like "log-status", "failure", and "time-change". As a catch-all, BACnet allows TL log-datum to hold "any-value".

This idea (different datatyped log-datums) is a key design consideration fot TL objects. First of all, not all TL applications require this level of flexibility, and most TL use cases could be covered by a subset of these many datatypes e.g. unsigned, real. Even values like enumerated and unsigned may often be restricted to smaller maximum values such as bytes or 16 bit words. Enumerated, unsigned, integer and BitString datatypes are allowed to presume restricted ranges such as 32 bits of precision. Such measures greatly reduce the overall size required to store the Log Buffer.

As a weird edge case, we can conceive of applications where other datatypes are required. For example, CharacterString values, or even larger structured datatype values. Most TL objects do not support this extended functionality even though the standard says how to do so if you want to.

TLM objects have these same issues but potentially multiplied by N values per log record.

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