BACnet Interoperability Building Blocks

Following up on the discussions in Philadelphia, I would like to submit the following proposal for a way we might address the increasingly critical need of consulting engineers for a simple, understandable methodology for specifying interoperable BACnet systems. It has become clear to everyone by now that having standard objects and services is necessary but, by itself, insufficient to ensure interoperability. It is likewise insufficient to merely prescribe groups of objects and services. What is needed is to define how specific combinations of BACnet objects and services are to be used together to achieve various commonly understood and desired functions. To some extent, this runs counter to the long-asserted mantra that the SPC (SSPC) was never going to tell anyone how to "make their system work." But the inescapable fact is that it is precisely a specification of "how things work" that is required to make systems interoperable. Needless to say, the pie can be sliced in many ways. I look forward to many carefree, idyllic hours of debate on just how we are going to do the slicing so that everyone can amicably live with their slice(s)... Please pass the whipped cream.

Here are my assumptions:

1) It is unimportant what we call the following prescriptions but we need to clearly distinguish them from the current conformance class and functional groups. I previously suggested the designation "*BACnet Operational Groups (BOGs)*" but perhaps "*BACnet Profiles (BPs)*", "*BACnet Configuration Specifications (BCSs)*" or "*BACnet Interoperability Building Blocks (BIBBs)*" would be better -- or something entirely different. Let your imagination run wild - just like we did when we picked "BACnet" itself out of all the various possibilities that had been dreamed up. (ASHnet anyone?) The important thing is to define them. Then we can figure out what to call them. Let's call them BIBBs for now.

2) It is critical that each BIBB be named in such a way that it is obvious to consultants with only a knowledge of the desired result to which function it pertains. Thus, the BIBBs must have simple names like "Alarm Management" or "Scheduling".

3) The "black box" concept needs to be implemented. A consultant should be able to use language like "Each supplied device shall implement the BACnet capabilities contained in the 'Scheduling (xxx)' and 'Alarm Management (xxx)' BIBBs." (I'll get to (xxx) in a moment.) It is up to us to define how each BIBB applies to different types of BACnet devices so that manufacturers know what is expected when a specification comes along. Since the terms "client" and "server"

seem to confuse people, I propose, for now at least, to use the terms "Workstation" and "Field Panel." In general, these terms, understood by everyone, convey the intended idea. Naturally, in hierarchical systems, there will be times when a field panel may be a client to a lower-level server. In such cases, I would like to propose that we use the terms "Provider" and "User". See, for example, Point Sharing.

4) The nub of our problem, of course, is that many functions can be carried out in more than one way using the capabilities that BACnet provides. This is not an accident of nature. SPC 135P knew that there exists a range of product capabilities and that there is always the possibility of reducing the number of "bytes on the wire" by imposing a greater requirement for processing power. The question that then arises is "how do we describe BAC functions in terms of this range of processing power?" We could talk about "Less Powerful" and "More Powerful". We could use terms like "Basic" and "Advanced". Or "Basic" and "Enhanced". Or "Standard" and "High Performance", etc., etc. The problem is that none of these terms really mean anything to a consultant. "Advanced" compared to what? What good is "More Powerful" in my application?

It finally occurred to me that maybe the best way out of this mess is again to use terms that anyone can understand: "Small System" and "Large System" (= (xxx) above) . Properly described, I think these terms actually capture the spirit of what most of us on SPC 135P had in mind when we talked about performance issues. A "small" <u>device</u> tended to be equated with a device without much processing power; the proverbial "dirtball controller." Similarly, a "small" <u>system</u> was thought of as a system with few nodes, thus not requiring high bandwidth or sophisticated processing to accomplish real work. In a like manner, "large" has always been associated with lots of processing power, many nodes, or both. In any event, this is the model I think we should explore.

5) Another issue is the number of gradations for any given BIBB. With just two descriptors, "Small System" and "Large System," there would be only two sets of capabilities. Some BIBBs may only need one. Some might need more. I would suggest, in the interest of simplicity, that we keep it to two initially. As will be seen below, this seems to work well for most of the BIBBs I have tried to define. In any case, let the debate begin.

The following BIBBs have been drafted:

• Alarm Management (Small System, Large System)
• Closed Loop Control (Small System, Large System)
• Event Logging (Small System, Large System)

• Event Sharing (Small System, Large System)
• Graphics (Small System, Large System)
• Point Sharing (Small System, Large System)
• Remote Configuration
• Scheduling (Small System, Large System)
• Trend Logging (Small System, Large System)

Each BIBB is described in terms of a Name, Purpose, Required Services, Required Objects, and Explanation.

6) Finally, it must be noted that what is presented here only addresses one side of this issue. The description of each BIBB in terms of the BACnet capabilities required to implement it is intended to convey to implementers what they need to provide when they see a certain BIBB in a job specification. The other side of the issue is to communicate to specifiers, i.e., consulting engineers, when the use of a particular BIBB is appropriate.

If you study the following BIBBs you will see that there are two types. In one type, the large system model simply adds functionality that the small system model does not offer. For example, in Closed Loop Control (Small System) the Loop object properties have the access attributes as specified in the standard, i.e., they are at most read-only (although writable at an implementer's discretion). In Closed Loop Control (Large System) some of the properties are required to be writable, an extension to the standard. A specifier can make a judgment on which BIBB to specify based on the expected impact that parameter writability would have on the application and the additional cost, if any, of the enhanced capability.

The second type of BIBB describes two distinct ways of achieving the same end result. The difference is network efficiency. These kinds of BIBBs can be compared quantitatively using the same technique as is always used to compare computer system performance: benchmarks. Let me give an example.

Suppose we want to compare the Graphics (Small System) and Graphics (Large System) BIBBs. As described below, the small system model uses ReadProperty to gather the data for display; the large system model uses as one possibility the COV mechanism. Given a benchmark, we can compute the number of bytes to display the graphic using each mechanism and thus have a comparison of the relative efficiency, expressed in terms of bytes-on-the-wire for each approach.

Here is a sample benchmark (if you don't like this one create your own; in fact maybe the CCFG WG should create a range of them...):

Number of data points in the graphic:	10	(assume analog points)
Maximum screen update interval:	5	seconds
Length of time graphic is running:	60	seconds
Rate of COV-triggerable events:	10	events/minutes

With these benchmark parameters we can now easily compute how many application layer octets are required by each BIBB (if you want to add in the data link and network layer octets, be my guest):

Graphics (Small System)

Each point must be read every 5 seconds. Since the graphic runs for one minute, there will be 12 sets of 10 readings, i.e., 120 ReadProperty transactions.

# octets for ReadProperty:	11
# octets or ComplexACK:	17
# octets per transaction (11+17):	28
# octets for the graphic (120*28):	**3360**

Graphics (Large System)

Here we must first issue 10 SubscribeCOV requests. Assume the Issue Confirmed Notifications parameter (2 octets) is omitted (we only need Unconfirmed Notifications for a graphic) and the Lifetime parameter = 0 for indefinite lifetime.

# octets for 10 SubscribeCOVs:	130
# octets for 10 SimpleACKs:	30
# octets for 10 initial UnconfirmedCOVNotifications:	270
# octets for 10 more COV notifications during graphic:	270
# octets to cancel 10 COV subs at graphic shutdown:	110
# octets for the graphic (sum of the above):	**810**

Thus, for this benchmark, the large system model uses only about one quarter the number of bytes to produce essentially the same result, a 4 to 1 advantage.

So when should a specifier call for one BIBB as opposed to the other? It comes down to analyzing a job's bandwidth situation. If there are only a few controllers but the network is fast, say Ethernet, the small system model will probably work fine. If, however, there are many controllers, the possibility of other high priority traffic (not related to the operator's graphics), and the network is slower, say ARCNET, the large system model would seem a better specification. In either case, the BIBBs tell a prospective supplier exactly what BACnet capabilities are required. The specifying engineer still has to make some engineering judgments based on a knowledge of the proposed network but that is exactly what the owner is paying him/her for! And with clearly defined - and specifiable alternatives - the calculations are relatively straightforward (which, as stated above, we should probably do as a service to the specifier community).

Here are thumbnail sketches of the BIBBs.

Alarm Management (Small System, Large System)

These BIBBs require implementation of the event notification services. Intrinsic reporting is assumed at a minimum. The small BIBB assumes that the workstation maintains the alarm acknowledgment history (equivalent to the Acked_Transitions property of Event Enrollment objects or objects that support intrinsic reporting) and the current alarm state (equivalent to the Event_State property, likewise of such objects). For the small system BIBB no alarm acknowledgment function is required at the field panel level. The large BIBB assumes that the field panel maintains the alarm acknowledgment history and alarm state. AcknowledgeAlarm is also required. Large system field panels may do algorithmic reporting in lieu of intrinsic reporting. They may also do both, but they must do at least one. Note that intrinsic reporting requires support of the Notification Class object. Thus both the small and large BIBBs require this object.

Obviously, one could imagine a scenario where the alarm mechanism is entirely workstation-initiated, i.e., the workstation goes out, reads some properties, does some calculations and decides there is an alarm. Systems are, of course, free to do this but since this does not make use of the services intended by the SPC to perform the alarm management function they should not be able to claim that they do "Alarm Management" as defined in these BIBBs.

Closed Loop Control (Small System, Large System)

For the small system, only ReadProperty is required for the observing the properties of the required read-only Loop object. In the large system, certain properties of the Loop object are required to be writable and both Read- and WriteProperty are required for the manipulation of the Loop object's properties.

Event Logging (Small System, Large System)

The small system BIBB assumes a workstation-based solution but the autonomous generation of event notifications by the field panels. The workstation does all the logging.

Besides performing the notifications required above, the large system field panel also logs the events to a local file. The File object type and the AtomicReadFile service are required so that the workstation can read the log file. The large field panel must also support GetEnrollmentSummary and, if algorithmic reporting is present, the EventEnrollment object type.

Note that the event logging BIBBs will be significantly altered if and when we reach consensus on a Log object type or agree on a log file format.

Event Sharing (Small System, Large System)

Event sharing for small systems just means the use of the event notification services. Here is an instance where "workstation" is not as good a term as "user" and "field panel" is not as good as "provider" so I've used the latter terms.

Large system devices must also support GetEnrollmentSummary.

Graphics (Small System, Large System)

Small system graphics is the simplest BIBB of all. The workstation just uses ReadProperty to gather data for display.

In the large system BIBB I have required support of the COV mechanism as well as the dynamic creation of Group objects. These things were put in the standard for a good reason: this is it!

Point Sharing (Small System, Large System)

Small system point sharing requires the user of the point data to initiate ReadProperty service requests and providers to execute them, i.e., send back the data.

I think for large systems, point sharing should be implemented using the COV services.

Remote Configuration

This one makes me a little queasy. I have specified use of the VT services. Maybe we should just offer a choice to implementers: as long as they use BACnet to do the job, they conform. Thus, they could use their choice of the VT, PrivateTransfer, or file access services, or a combination thereof. The point is that the configuration is done over the network using BACnet services.

Scheduling (Small System, Large System)

Small systems need to support the Schedule and Calendar objects, the basic property access services, and the TimeSynchronization service. Large systems add the capability to define Command objects.

Trend Logging (Small System, Large System)

Similar to event logging, the small system BIBB is workstation-based. The workstation uses ReadProperty to read values, then files them. In the large system, the field panel files the values and these are then read as a file by the workstation.

Again, adoption of a Log object type would fundamentally change the large system BIBB.

And now, for your entertainment and reflection, the BIBBs...

X BACnet Interoperability Building Block: Alarm Management (Small System)

This BACnet interoperability building block provides for alarm management in small systems.

X.1 Required Services

Table X-1. Alarm Management (Small System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ConfirmedEventNotification		x	x	
UnconfirmedEventNotification		x	x	
GetAlarmSummary	x			x

X.2 Required Objects

Table X-2. Alarm Management (Small System) Required Objects

BACnet Objects	Workstation	Field Panel
NotificationClass		x

In addition, each field panel claiming support of "Alarm Management (Small System)" shall implement one or more objects that provide "intrinsic reporting" as described in 13.2.

X.3 Explanation

In "Alarm Management (Small System)" the management of alarms is predominantly a workstation function. The field panels support the generation of alarms through the "intrinsic reporting" technique, but they are not required to maintain acknowledgment status information since acknowledgments are not required. It is entirely the responsibility of the workstation to keep track of which alarms are defined, where they reside, and their current status. If operator acknowledgment is provided, it is strictly an internal workstation function that does not require communication with the field panel. Upon workstation startup, each workstation shall synchronize its alarm database by issuing a GetAlarmSummary service request to each field panel from which it wishes to receive subsequent alarm status change information.

X BACnet Interoperability Building Block: Alarm Management (Large System)

This BACnet interoperability building block provides for alarm management in large systems.

X.1 Required Services

Table X-1. Alarm Management (Large System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ConfirmedEventNotification		x	x	
UnconfirmedEventNotification		x	x	
GetAlarmSummary	x			x
GetEnrollmentSummary	x			x
AcknowledgeAlarm	x			x

X.2 Required Objects

Table X-2. Alarm Management (Large System) Required Objects

BACnet Objects	Workstation	Field Panel
NotificationClass		x
EventEnrollment (Algorithmic Reporting)		x

Each field panel claiming support of "Alarm Management (Large System)" shall implement one or more objects that provide "intrinsic reporting" as described in 13.2. If a field panel also supports "algorithmic change reporting" as described in 13.3, then implementation of the EventEnrollment object type is also required.

X.3 Explanation

In "Alarm Management (Large System)" the management of alarms is a more cooperative activity than in "Alarm Management (Small System)." The field panels support the generation of alarms through the "intrinsic reporting" technique and may, optionally, support "algorithmic change reporting." In the latter case, EventEnrollment objects are required. The field panels maintain acknowledgment status information as needed by their application and the workstation is expected to acknowledge alarm receipt upon request. Upon workstation startup, each workstation shall synchronize its alarm database by issuing a GetAlarmSummary service request to each field panel from which it wishes to receive subsequent alarm status change information.

X BACnet Interoperability Building Block: Closed Loop Control (Small System)

This BACnet interoperability building block for small systems enables communication related to closed loop control algorithms that can be mapped to the BACnet Loop object type.

X.1 Required Services

Table X-1. Closed Loop Control (Small System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ReadProperty	x			x

X.2 Required Objects

Table X-2. Closed Loop Control (Small System) Required Objects

BACnet Objects	Workstation	Field Panel
Loop		x

X.3 Explanation

"Closed Loop Control" provides the capability to monitor the operation of closed loop analog control algorithms that can be modeled using the BACnet Loop object type. This includes conventional P, PI, and PID control. Note that all properties of the Loop object are read-only.

X BACnet Interoperability Building Block: Closed Loop Control (Large System)

This BACnet interoperability building block for large systems enables communication related to closed loop control algorithms that can be mapped to the BACnet Loop object type.

X.1 Required Services

Table X-1. Closed Loop Control (Large System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ReadProperty	x			x
WriteProperty	x			x

X.2 Required Objects

Table X-2. Closed Loop Control Required Objects

BACnet Objects	Workstation	Field Panel
Loop		x

To claim conformance to this BIBB, the following properties of the Loop object, at a minimum, are required to be present and writable: Out_Of_Service, Update_Interval, Setpoint_Reference, Setpoint, Action, Proportional_Constant, Integral_Constant, Derivative_Constant, and Bias.

X.3 Explanation

"Closed Loop Control" provides the capability to monitor the operation of closed loop analog control algorithms that can be modeled using the BACnet Loop object type. This includes conventional P, PI, and PID control. The properties that are required to be present and writable in X.2 are those typically used in control loop tuning.

X BACnet Interoperability Building Block: Event Logging (Small System)

This BACnet interoperability building block provides for event logging in small systems.

X.1 Required Services

Table X-1. Event Logging (Small System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ConfirmedEventNotification		x	x	
UnconfirmedEventNotification		x	x	

X.2 Required Objects

Table X-2. Event Logging (Small System) Required Objects

BACnet Objects	Workstation	Field Panel
NotificationClass		x
EventEnrollment (Algorithmic Reporting)		x

Each field panel claiming support of "Event Logging (Small System)" shall implement one or more objects that provide "intrinsic reporting" as described in 13.2 or, if algorithmic reporting is implemented, one or more EventEnrollment objects, or both.

X.3 Explanation

In "Event Logging (Small System)" the work of logging events is entirely a workstation function. "Logging" for events means maintaining a file of information that contains a record of the event and the time the event occurred. The occurrence of the event is signaled by the receipt of either a confirmed or unconfirmed event notification service request.

X BACnet Interoperability Building Block: Event Logging (Large System)

This BACnet interoperability building block provides for event logging in large systems.

X.1 Required Services

Table X-1. Event Logging (Large System) Required Services

	Workstation		Field Panel	
BACnet Service	Initiate	Execute	Initiate	Execute
ConfirmedEventNotification		x	x	
UnconfirmedEventNotification		x	x	
GetEnrollmentSummary	x			x
AtomicReadFile	x			x

X.2 Required Objects

Table X-2. Event Logging (Large System) Required Objects

BACnet Objects	Workstation	Field Panel
NotificationClass		x
EventEnrollment (Algorithmic Reporting)		x
File		x

Each field panel claiming support of "Event Logging (Large System)" shall implement one or more objects that provide "intrinsic reporting" as described in 13.2 or, if algorithmic reporting is implemented, one or more EventEnrollment objects, or both. In addition, each field panel shall maintain one or more event log files whose attributes are network visible by means of related File objects.

X.3 Explanation

In "Event Logging (Large System)" the field panels support the generation of event notifications through either the "intrinsic reporting" technique or the "algorithmic change reporting" technique, or both. In the case of algorithmic change reporting, EventEnrollment objects are required. The capabilities of "Event Logging (Small System)", where the work of logging events is entirely a workstation function, is supplemented here by the addition of one or more log files that are maintained by the field panel. The contents of the file(s) pointed to by the field panel's File object(s) are accessible by means of the AtomicReadFile service.

X BACnet Interoperability Building Block: Event Sharing (Small System)

This BACnet interoperability building block provides for event sharing in small systems.

X.1 Required Services

Table X-1. Event Sharing (Small System) Required Services

BACnet Service	User		Provider	
	Initiate	Execute	Initiate	Execute
ConfirmedEventNotification		x	x	
UnconfirmedEventNotification		x	x	

X.2 Required Objects

Table X-2. Event Sharing (Small System) Required Objects

BACnet Objects	User	Provider
NotificationClass		x

Each provider claiming support of "Event Sharing (Small System)" shall implement one or more objects that provide "intrinsic reporting" as described in 13.2.

X.3 Explanation

"Event Sharing (Small System)" provides for the asynchronous distribution of event notifications through the confirmed and unconfirmed event notification services.

X BACnet Interoperability Building Block: Event Sharing (Large System)

This BACnet interoperability building block provides for event sharing in large systems.

X.1 Required Services

Table X-1. Event Sharing (Large System) Required Services

BACnet Service	User		Provider	
	Initiate	Execute	Initiate	Execute
ConfirmedEventNotification		x	x	
UnconfirmedEventNotification		x	x	
GetEnrollmentSummary	x			x

X.2 Required Objects

Table X-2. Event Sharing (Large System) Required Objects

BACnet Objects	User	Provider
NotificationClass		x
EventEnrollment (Algorithmic Reporting)		x

Each provider claiming support of "Event Sharing (Large System)" shall implement one or more objects that provide "intrinsic reporting" as described in 13.2. If a provider also supports "algorithmic change reporting" as described in 13.3, then implementation of the EventEnrollment object type is also required.

X.3 Explanation

In "Event Sharing (Large System)" the providers support the generation of event notifications through the "intrinsic reporting" technique and may, optionally, support "algorithmic change reporting." In the latter case, EventEnrollment objects are required.

X BACnet Interoperability Building Block: Graphics (Small System)

This BACnet interoperability building block enables the retrieval of any property of any object for display on a workstation's graphic.

X.1 Required Services

Table X-1. Graphics (Small System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ReadProperty	x			x

X.2 Explanation

In "Graphics (Small System)" the graphically displayed data are updated entirely by the action of the workstation. Thus, the sequence and rate of update are solely dependent on the graphic application program in the workstation.

X BACnet Interoperability Building Block: Graphics (Large System)

This BACnet interoperability building block enables the retrieval of any property of any object for display on a workstation's graphic. This BIBB improves the performance of graphics by eliminating communications overhead once a graphic has been initiated.

X.1 Required Services

Table X-1. Graphics (Large System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ReadProperty	x			x
ReadPropertyMultiple	x			x
CreateObject	x			x
DeleteObject	x			x
SubscribeCOV	x			x
ConfirmedCOVNotification		x	x	
UnconfirmedCOVNotification		x	x	

X.2 Required Objects

Table X-2. Graphics (Large System) Required Objects

BACnet Objects	Workstation	Field Panel
Group		x

X.3 Explanation

In "Graphics (Large System)" three optimization techniques are available.

The first simply uses ReadPropertyMultiple to read several properties at one time.

The second involves creating a Group object whose present value represents the property values that are to be displayed. Any property of any object may be referred to in this way. The graphic is updated by periodically reading the present value of the Group object. Group objects have no preset lifetime. Thus this technique is well-suited to situations where the same graphic is frequently desired.

The third technique is available for a limited set of properties of a limited set of object types. See Table 13-1. SubscribeCOV messages are sent each time a graphic is initiated and are maintained for the lifetime of the graphic. Once initiated, such graphics make optimal use of the network since no communication occurs unless and until a COV has taken place. It is the workstation's responsibility to cancel all outstanding COV subscriptions when the graphic display is terminated. This technique is particularly suited to situations where it is desired to leave a graphic on display for extended periods of time without placing a load on the network or where the graphic is not expected to be used with great frequency. An example is a graphic depicting the elements of a control loop which is periodically tuned over the network.

X BACnet Interoperability Building Block: Point Sharing (Small System)

This BACnet interoperability building block enables point data, i.e., a property of a particular object, to be shared by virtue of its value being available via the ReadProperty service. Such objects could represent shared points such as a common outdoor air temperature sensor, a binary input representing a globally significant contact closure, and so on.

X.1 Required Services

Table X-1. Point Sharing (Small System) Required Services

	User		Provider	
BACnet Service	Initiate	Execute	Initiate	Execute
ReadProperty	x			x

X.2 Required Objects

No objects are required except the one whose property or properties are to be made available for shared use.

X.3 Explanation

In "Point Sharing (Small System)" common values of interest are made available by means of the ReadProperty service initiated by the device desiring to share the point information.

X BACnet Interoperability Building Block: Point Sharing (Large System)

This BACnet interoperability building block enables the subscription to, and distribution of, changes of values of any object that supports COV notification. Such objects could represent shared points such as a common outdoor air temperature sensor, a binary input representing a globally significant contact closure, and so on.

X.1 Required Services

Table X-1. Point Sharing Required Services

	User		Provider	
BACnet Service	Initiate	Execute	Initiate	Execute
SubscribeCOV	x			x
ConfirmedCOVNotification		x	x	
UnconfirmedCOVNotification		x	x	

X.2 Required Objects

Each field panel claiming support of "Point Sharing" shall implement at least one instance of one of the following standard object types that supports COV reporting as indicated in Table 13-1, namely, Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Multi-state Input, or Multi-state Output, or Loop.

X.3 Explanation

In "Point Sharing" common values of interest are made available by means of the COV subscription and notification services. [Perhaps we should define "Point Sharing (Provider)" and "Point Sharing (Subscriber)" BIBBs to differentiate between those panels that offer sharing as opposed to those that only make use of shared values from others.]

X BACnet Interoperability Building Block: Remote Configuration

This BACnet interoperability building block provides for the remote configuration of devices using the virtual terminal services.

X.1 Required Services

Table X-1. Alarm Management (Small System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
VT-Open	x			x
VT-Close	x			x
VT-Data	x	x	x	x

X.2 Explanation

While device configuration activities are generally outside the scope of BACnet, support of "Remote Configuration" allows a workstation to communicate with a field panel in virtual terminal mode and thus to potentially access a configuration application on the remote device.

X BACnet Interoperability Building Block: Scheduling (Small System)

This BACnet interoperability building block provides for scheduling of time related activities in small systems.

X.1 Required Services

Table X-1. Scheduling (Small System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ReadProperty	x			x
WriteProperty	x			x
TimeSynchronization	x			x

X.2 Required Objects

Table X-2. Scheduling (Small System) Required Objects

BACnet Objects	Workstation	Field Panel
Calendar		x
Schedule		x

X.3 Explanation

"Scheduling (Small System)" provides the capabilities needed to allow a workstation to inspect and modify the properties of Calendar and Schedule objects used by a field panel to control its network-visible time-related activities. In addition, the time-of-day clock in the field panel is synchronized by use of TimeSynchronization service requests issued by the workstation.

X BACnet Interoperability Building Block: Scheduling (Large System)

This BACnet interoperability building block provides for scheduling of multiple time related activities in large systems.

X.1 Required Services

Table X-1. Scheduling (Large System) Required Services

	Workstation		Field Panel	
BACnet Service	Initiate	Execute	Initiate	Execute
ReadProperty	x			x
WriteProperty	x			x
TimeSynchronization	x			x

X.2 Required Objects

Table X-2. Scheduling (Large System) Required Objects

BACnet Objects	Workstation	Field Panel
Calendar		x
Command		x
Schedule		x

X.3 Explanation

"Scheduling (Large System)" provides the capabilities needed to allow a workstation to inspect and modify the properties of Calendar and Schedule objects used by a field panel to control its network-visible time-related activities. The time-of-day clock in the field panel is synchronized by use of TimeSynchronization service requests issued by the workstation. In addition, each field panel claiming to support this BIBB shall also support the Command object type.

X BACnet Interoperability Building Block: Trend Logging (Small System)

This BACnet interoperability building block provides for trend logging in small systems.

X.1 Required Services

Table X-1. Trend Logging (Small System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ReadProperty	x			x

X.2 Required Objects

Each field panel claiming support of "Trend Logging (Small System)" shall implement at least one instance of one of the following standard object types: Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Multi-state Input, or Multi-state Output.

X.3 Explanation

In "Trend Logging (Small System)" the work of logging trend data is entirely a workstation function. "Logging" of trend data means maintaining a file of periodic readings of some property of some object. The property is read at the discretion of the workstation using the ReadProperty service.

X BACnet Interoperability Building Block: Trend Logging (Large System)

This BACnet interoperability building block provides for trend logging in large systems.

X.1 Required Services

Table X-1. Trend Logging (Large System) Required Services

BACnet Service	Workstation		Field Panel	
	Initiate	Execute	Initiate	Execute
ReadProperty	x			x
AtomicReadFile	x			x

X.2 Required Objects

Table X-2. Trend Logging (Large System) Required Objects

BACnet Objects	Workstation	Field Panel
File		x

In addition to one or more File objects, each field panel claiming support of "Trend Logging (Large System)" shall implement at least one instance of one of the following standard object types: Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Multi-state Input, or Multi-state Output.

X.3 Explanation

In "Trend Logging (Large System)" the ability of the workstation to log trend data is supplemented by the addition of one or more trend log files that are maintained by the field panel. The contents of the file(s) pointed to by the field panel's File object(s) are accessible by means of the AtomicReadFile service. "Logging" of trend data means maintaining a file of periodic readings of some property of some object.