Addendum *c* to ANSI/ASHRAE 135-1995

ASHRAE® STANDARD

BACnet™ - A Data Communication Protocol for Building Automation and Control Networks

©2001 American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.

Approved by the ASHRAE Standards Committee June 23, 2001; by the ASHRAE Board of Directors June 28, 2001; and by the American National Standards Institute September 7, 2001.

This standard is under continuous maintenance by a Standing Standard Project Committee (SSPC) for which the Standards Committee has established a documented program for regular publication of addenda or revisions, including procdedures for timely, documented, consensus action on requests for change to any part of the standard. The change submittal form, instructions and deadlines may be obtained in electronic form from ASHRAE's Internet Home Page, *http://www.ashrae.org*, or in paper form from the Manager of Standards. The latest edition of an ASHRAE Standard and printed copies of a public review draft may be purchased from ASHRAE Customer Service, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. E-mail: *orders@ashrae.org*. Fax: 404-321-5478. Telephone: 404-636-8400 (worldwide), or toll free 1-800-527-4723 (for orders in U.S. and Canada).

AMERICAN SOCIETY OF HEATING, REFRIGERATING AND AIR-CONDITIONING ENGINEERS, INC.
1791 Tullie Circle, NE · Atlanta GA 30329-2305

ASHRAE STANDING STANDARD PROJECT COMMITTEE 135
Cognizant TC: TC 1.4, Control Theory and Applications
SPLS Liaison: Ronald E. Jarnagin

*Steven T. Bushby, *Chairman*
William O. Swan III, *Vice-Chairman*
*Keith A. Corbett, *Secretary*
*Barry B. Bridges
*A. J. Capowski
*Jeffery Cosiol
*Thomas S. Ertsgaard
*Daniel P. Giorgis
*Stephen Karg
*J. D. Ljungquist
*Jerald P. Martocci
*Carl Neilson
*Mark A. Railsback
*David Robin
*Daniel A. Traill
*Grant N. Wichenco
Dana R. Epperson
Winston I. Hetherington
David J. Branson
Kevin G. Sweeney

**Denotes members of voting status this standard was approved for publication.*

ASHRAE STANDARDS COMMITTEE 2000-2001

Martha J. Hewett, *Chairman*
Nance C. Lovvorn, *Vice-Chairman*
Van D. Baxter
Dean S. Borges
Waller S. Clements
Piotr A. Domanski
Richard A. Evans
John F. Hogan
Ronald E. Jarnagin
David E. Knebel
Frederick H. Kohloss
William J. Landman
Rodney H. Lewis
Ross D. Montgomery
Davor Novosel
Joseph A. Pietsch
James A. Ranfone
Michael Tavares
Steven T. Taylor
James K. Vallort
Thomas E. Watson
Bruce A. Wilcox
J. Richard Wright
Gerald C. Groff, BOD ExO
William J. Buck, CO

Claire B. Ramspeck, Manager of Standards

SPECIAL NOTE

This American National Standard (ANS) is a national voluntary consensus standard developed under the auspices of the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). Consensus is defined by the American National Standards Institute (ANSI), of which ASHRAE is a member and which has approved this standard as an ANS, as "substantial agreement reached by directly and materially affected interest categories. This signifies the concurrence of more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that an effort be made toward their resolution." Compliance with this standard is voluntary until and unless a legal jurisdiction makes compliance mandatory through legislation.

ASHRAE obtains consensus through participation of its national and international members, associated societies, and public review.

ASHRAE Standards are prepared by a Project Committee appointed specifically for the purpose of writing the Standard. The Project Committee Chair and Vice-Chair must be members of ASHRAE; while other members may or may not be members of ASHRAE, all must be technically qualified in the subject area of the standard. Every effort is made to balance the concerned interests on all Project Committees.

The Manager of Standards of ASHRAE should be contacted for:

a. interpretation of the contents of this Standard,
b. participation in the next review of the Standard,
c. offering constructive criticism for improving the Standard,
d. permission to reprint portions of the Standard.

ASHRAE INDUSTRIAL ADVERTISING POLICY ON STANDARDS

ASHRAE Standards and Guidelines are established to assist industry and the public by offering a uniform method of testing for rating purposes, by suggesting safe practices in designing and installing equipment, by providing proper definitions of this equipment and by providing other information which may serve to guide the industry. The creation of ASHRAE Standards is determined by the need for them, and conformance to them is completely voluntary.

In referring to this standard and marking of equipment and in advertising, no claim shall be made, either stated or implied, that the product has been approved by ASHRAE.

DISCLAIMER

ASHRAE uses its best efforts to promulgate standards for the benefit of the public in light of available information and accepted industry practices. However, ASHRAE does not guarantee, certify, or assure the safety or performance of any products, components, or systems tested, designed, installed, or operated in accordance with ASHRAE's Standards or Guidelines or that any tests conducted under its standards will be nonhazardous or free from risk.

This informative foreword is not part of the standard.

Foreword

The purpose of this addendum is to add a number of independent substantive changes to the BACnet standard. These modifications are the result of change proposals made pursuant to ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135. The changes and their rationale are summarized below.

135*c*-1. Add a new Life Safety Point object type that provides a network-visible way to represent the characteristics of initiating and indicating devices in fire, life safety, and security applications, p. 1.

Some of the ASN.1 changes needed to add this object type are included in 135*c*-3.

135*c*-2. Add a new Life Safety Zone object type that provides a network-visible way to represent the characteristics associated with an arbitrary group of BACnet Life Safety Point and Life Safety Zone objects, p. 13.

Some of the ASN.1 changes needed to add this object type are included in 135*c*-3.

135*c*-3. Add functionality to the existing BACnet alarm and event features needed to support the Life Safety Point and Life Safety Zone object types, p. 22.

To fully integrate the proposed Life Safety Point and Life Safety Zone object types into the existing BACnet alarm and event features, several additions are required.

135*c*-4. Add a new LifeSafetyOperation service that provides silence and reset capabilities needed for life safety systems, p. 29.

This new service provides a way to implement silence and reset operation for latched troubles and alarms that are needed for life safety systems.

135*c*-5. Add a new subclause to 16 to describe the use of existing BACnet services to provide backup and restore capability, p. 31.

The original standard did not address the issue of backing up and restoring device characteristics in a common way. This new subclause provides these capabilities using existing services. The ReinitializeDevice service is used to control the state of the device being backed-up or restored. The file services are used to retrieve the configuration file(s).

135*c*-6. Define a new service, SubscribeCOVProperty, to allow COV notifications for arbitrary properties of an object with subscriber-specified COV increments, p. 38.

The prior COV services do not allow a device to subscribe to arbitrary properties of an object, nor do they allow for different subscribers to specify different COV increments. To broaden the utility of the COV subscription mechanism, a new service called SubscribeCOVProperty is added. The service is structured identically to SubscribeCOV in the first 4 parameters so as to allow for code reuse by implementers. In addition, the COV subscriptions in a device have been made network visible through a new property in the device object. Note that the service still allows a zero value for the Lifetime parameter so as to mimic the operational behavior of SubscribeCOV.

135*c*-7. Add Vendor ID to proprietary MS/TP frames, p. 45.

Proprietary MS/TP frames (frame types 128 through 255) were at best risky to use due to the lack of a vendor identification field. While various techniques such as CRCs can be used to reduce the risk of accidentally interpreting another vendor's proprietary frame, they do not entirely eliminate the risk. This addition to the standard solves the problem by requiring the unique vendor identification value to be inserted at the start of the data portion of proprietary frames.

135*c*-8 Add a new service, GetEventInformation, that provides enough information to acknowledge alarms, p. 46.

The prior GetAlarmSummary service does not provide enough information to perform an alarm acknowledgment. The returned list includes objects that have an active alarm, but not objects that have an Event_State whose value is NORMAL and also have unacknowledged transitions.

In addition, there was no simple service to get information about objects that have a Notify_Type = EVENT, which is also a requirement, as BACnet makes no distinction between EVENT events and ALARM events.

The solution is to define a new service, GetEventInformation. This service is intended to supersede the GetAlarmSummary service.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-1995 is indicated through the use of *italics* while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout. These instances are 135*c*-1, 2, 4, 5, 6 and 9.

In some portions of this addendum, the value of 'X' is used in the clause number. This is a placeholder that will be replaced at the time of publication of a consolidated version of the standard with a value placing it in the proper sequence in the parent Clause.

135*c*-1. Add a new Life Safety Point object type that represents the characteristics of initiating and indicating devices in the fire, life safety, and security applications.

12.X Life Safety Point Object Type

The Life Safety Point object type defines a standardized object whose properties represent the externally visible characteristics associated with initiating and indicating devices in fire, life safety and security applications. The condition of a Life Safety Point object is represented by a *mode* and a *state*.

Mode changes determine the object's inner logic and, consequently, influence the evaluation of the state. Typically, the operating *mode* would be under operator control.

The *state* of the object represents the condition of the controller according to the logic internal to the device. The implementation of the logic applied to such controllers to determine the various possible states is a local matter.

Typical applications of the Life Safety Point object include automatic fire detectors, pull stations, sirens, supervised printers, etc. Similar objects can be identified in security control panels.

The Life Safety Point object type and its properties are summarized in Table 12-X and described in detail in this subclause.

NOTE: Do not confuse the Present_Value state with the Event_State property, which reflects the offnormal state of the Life Safety Point object.

Table 12-X. Properties of the Life Safety Point Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	BACnetLifeSafetyState	R[1]
Tracking_Value	BACnetLifeSafetyState	O
Description	CharacterString	O
Device_Type	CharacterString	O
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Reliability	BACnetReliability	R[1]
Out_Of_Service	BOOLEAN	R
Mode	BACnetLifeSafetyMode	W
Time_Delay	Unsigned	O[2]
Notification_Class	Unsigned	O[2]
Life_Safety_Alarm_Values	List of BACnetLifeSafetyState	O[2]
Alarm_Values	List of BACnetLifeSafetyState	O[2]
Fault_Values	List of BACnetLifeSafetyState	O[2]
Event_Enable	BACnetEventTransitionBits	O[2]
Acked_Transitions	BACnetEventTransitionBits	O[2]
Notify_Type	BACnetNotifyType	O[2]
Event_Time_Stamps	BACnetARRAY [3] of BACnetTimeStamp	O[2]
Silenced	BACnetSilencedState	R
Operation_Expected	BACnetLifeSafetyOperation	R
Maintenance_Required	BACnetMaintenance	O
Setting	Unsigned8	O
Direct_Reading	REAL	O[3]
Units	BACnetEngineeringUnits	O[3]
Member_Of	List of BACnetDeviceObjectReference	O

[1] These properties are required to be writable when Out_Of_Service is TRUE.
[2] These properties are required if the object supports intrinsic alarming.
[3] If either of these properties is present, then both must be present.

12.X.1 Object_Identifier

This property, of type BACnetObjectIdentifier, is a numeric code that is used to identify the object. It shall be unique within the BACnet Device that maintains it.

12.X.2 Object_Name

This property, of type CharacterString, shall represent a name for the object that is unique within the BACnet Device that maintains it. The minimum length of the string shall be one character. The set of characters used in the Object_Name shall be restricted to printable characters.

12.X.3 Object_Type

This property, of type BACnetObjectType, indicates membership in a particular object type class. The value of this property shall be LIFE_SAFETY_POINT.

12.X.4 Present_Value

This property, of type BACnetLifeSafetyState, reflects the state of the Life Safety Point object. The means of deriving the Present_Value shall be a local matter. Present_Value may latch non-NORMAL state values until reset. The Present_Value property shall be writable when Out_Of_Service is TRUE (see 12.X.11).

12.X.5 Tracking_Value

This optional property, of type BACnetLifeSafetyState, reflects the non-latched state of the Life Safety Point object. The means of deriving the state shall be a local matter. Unlike Present_Value, which may latch non-NORMAL state values until reset, Tracking_Value shall continuously track changes in the state.

12.X.6 Description

This optional property, of type CharacterString, is a string of printable characters whose content is not restricted.

12.X.7 Device_Type

This optional property, of type CharacterString, is a text description of the physical device that the Life Safety Point object represents.

12.X.8 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of the Life Safety Point object. Three of the flags are associated with the values of other properties of this object. A more detailed status could be determined by reading the properties that are linked to these flags. The relationship between individual flags is not defined by the protocol. The four flags are

{IN_ALARM, FAULT, OVERRIDDEN, OUT_OF_SERVICE}

where:

IN_ALARM	Logical FALSE (0) if the Event_State property (12.X.9) has a value of NORMAL, otherwise logical TRUE (1).
FAULT	Logical TRUE (1) if the Reliability property (12.X.10) does not have a value of NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	Logical TRUE (1) if the point has been overridden by some mechanism local to the BACnet Device. In this context "overridden" is taken to mean that the physical input(s) are no longer tracking changes to the Present_Value property and the Reliability property is no longer a reflection of the physical input(s).
OUT_OF_SERVICE	Logical TRUE (1) if the Out_Of_Service property (12.X.11) has a value of TRUE, otherwise logical FALSE (0).

12.X.9 Event_State

The Event_State property, of type BACnetEventState, is included in order to provide a way to determine if this object has an active event state associated with it. The Event_State property shall indicate the event

state of the object. If the object does not support intrinsic reporting, then the value of this property shall be NORMAL.

12.X.10 Reliability

The reliability property, of type BACnetReliability, provides an indication of whether the Present_Value or the operation of the physical input(s) in question are "reliable" as far as the BACnet Device or operator can determine and, if not, why. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, OPEN_LOOP, SHORTED_LOOP, MULTI_STATE_FAULT, UNRELIABLE_OTHER}.

12.X.10.1 Conditions for Generating a TO-FAULT Event

A TO-FAULT event is generated under these conditions:

(a) the TO-FAULT flag must be enabled in the Event_Enable property, and
(b) the Present_Value must equal at least one of the values in the Fault_Values list.

12.X.11 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the input(s) or process the object represents is not in service. This means that changes to the Present_Value property are decoupled from the input(s) or process when the value of Out_Of_Service is TRUE. In addition, the Reliability property and the corresponding state of the FAULT flag of the Status_Flags property shall be decoupled when Out_Of_Service is TRUE. While the Out_Of_Service property is TRUE, the Present_Value and Reliability properties may be changed to any value as a means of simulating specific fixed conditions or for testing purposes. Other functions that depend on the state of the Present_Value or Reliability properties shall respond to changes made to these properties while Out_Of_Service is TRUE, as if those changes had occurred to the input(s) or process.

12.X.12 Mode

This writable property, of type BACnetLifeSafetyMode, shall convey the desired operating mode for the Life Safety Point object. The Life Safety Point object shall generate CHANGE_OF_LIFE_SAFETY event notifications for any mode transition if the respective flags TO-OFFNORMAL, TO-FAULT or TO_NORMAL are set in the Event_Enable property (see 12.X.15.1, 12.X.15.2, 12.X.16.1, 12.X.16.2, 12.X.17.1, and 12.X.17.2).

12.X.13 Time_Delay

This optional property, of type Unsigned, shall specify the minimum period of time in seconds that the Present_Value must remain:

(a) equal to any of the values in the Life_Safety_Alarm_Values property before a TO-OFFNORMAL event is generated, or
(b) equal to any of the values in the Alarm_Values property before a TO-OFFNORMAL event is generated, or
(c) not equal to any of the values in the Life_Safety_Alarm_Values property, Alarm_Values property, or Fault_Values property before a TO-NORMAL event is generated.

This property is required if intrinsic reporting is supported by this object.

12.X.14 Notification_Class

This optional property, of type Unsigned, shall specify the notification class to be used when handling and generating event notifications for this object. The Notification_Class property implicitly refers to a Notification Class object that has a Notification_Class property with the same value. This property is required if intrinsic reporting is supported by this object.

12.X.15 Life_Safety_Alarm_Values

This optional property, of type List of BACnetLifeSafetyState, shall specify any states the Present_Value must equal before a TO-OFFNORMAL event is generated and event state LIFE_SAFETY_ALARM is entered. This property is required if intrinsic reporting is supported by this object.

12.X.15.1 Conditions for Generating a TO-OFFNORMAL Event

A TO-OFFNORMAL event is generated under these conditions:

(a) the TO-OFFNORMAL flag must be enabled in the Event_Enable property, and
(b) the Present_Value must equal any of the values in the Life_Safety_Alarm_Values list, and
(c) the Present_Value must remain within the Life_Safety_Alarm_Values list for a minimum period of time, specified by the Time_Delay property.

New events shall be generated upon a change of Mode if the TO-OFFNORMAL flag is set in the Event_Enable property.

12.X.15.2 Conditions for Generating a TO-NORMAL Event

Once equal, the Present_Value must become not equal to any of the states in the Life_Safety_Alarm_Values property, and not equal to any of the states in the Alarm_Values property, and not equal to any of the states in the Fault_Values property, before a TO-NORMAL event is generated under these conditions:

(a) the TO-NORMAL flag must be enabled in the Event_Enable property, and
(b) the Present_Value must remain not equal to any of the states in the Life_Safety_Alarm_Values property, and
(c) the Present_Value must remain not equal to any of the states in the Alarm_Values property, and
(d) the Present_Value must remain not equal to any of the states in the Fault_Values property, and
(e) the Present_Value must remain equal to the same value for a minimum period of time, specified by the Time_Delay property.

New events shall be generated upon a change of Mode if the TO-NORMAL flag is set in the Event_Enable property.

12.X.16 Alarm_Values

This optional property, of type List of BACnetLifeSafetyState, shall specify any states the Present_Value must equal before a TO-OFFNORMAL event is generated and event state OFFNORMAL is entered. This property is required if intrinsic reporting is supported by this object.

12.X.16.1 Conditions for Generating a TO-OFFNORMAL Event

A TO-OFFNORMAL event is generated under these conditions:

(a) the TO-OFFNORMAL flag must be enabled in the Event_Enable property, and
(b) the Present_Value must equal any of the values in the Alarm_Values list, and
(c) the Present_Value must remain within the Alarm_Values list for a minimum period of time specified by the Time_Delay property.

New events shall be generated upon a change of Mode if the TO-OFFNORMAL flag is set in the Event_Enable property.

12.X.16.2 Conditions for Generating a TO-NORMAL Event

Conditions for generating a TO-NORMAL event are defined in 12.X.15.2.

12.X.17 Fault_Values

This optional property, of type List of BACnetLifeSafetyState, shall specify any states the Present_Value must equal before a TO-FAULT event is generated. If Present_Value becomes equal to any of the states in the Fault_Values list, and no physical fault has been detected for any inputs that the Present_Value represents, then the Reliability property shall have the value MULTI_STATE_FAULT. The Fault_Values property is required if intrinsic reporting is supported by this object.

New events shall be generated upon a change of Mode if the TO-FAULT flag is set in the Event_Enable property.

12.X.17.1 Conditions for Generating a TO-FAULT Event

A TO-FAULT event is generated under these conditions:

(a) the TO-FAULT flag must be enabled in the Event_Enable property, and
(b) the Present_Value must equal at least one of the values in the Fault_Values list.

12.X.17.2 Conditions for Generating a TO-NORMAL Event

Conditions for generating a TO-NORMAL event are defined in 12.X.15.2.

12.X.18 Event_Enable

This optional property, of type BACnetEventTransitionBits, shall convey three flags that separately enable and disable reporting of TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events that are based on Present_Value and/or Mode changes. This property is required if intrinsic reporting is supported by this object.

12.X.19 Acked_Transitions

This optional property, of type BACnetEventTransitionBits, shall convey three flags that separately indicate the receipt of acknowledgements for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events. These flags shall be cleared upon the occurrence of the corresponding event and set under any of these conditions:

(a) upon receipt of the corresponding acknowledgement;

(b) upon the occurrence of the event if the corresponding flag is not set in the Event_Enable property (meaning event notifications will not be generated for this condition and thus no acknowledgement is expected);
(c) upon the occurrence of the event if the corresponding flag is set in the Event_Enable property and the corresponding flag in the Ack_Required property of the Notification Class object implicitly referenced by the Notification_Class property of this object is not set (meaning no acknowledgement is expected).

This property is required if intrinsic reporting is supported by this object.

12.X.20 Notify_Type

This optional property, of type BACnetNotifyType, shall convey whether the notifications generated by the object should be Events or Alarms. This property is required if intrinsic reporting is supported by this object.

12.X.21 Event_Time_Stamps

This optional property, of type BACnetARRAY[3] of BACnetTimeStamp, shall convey the times of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events, respectively. Time stamps of type Time or Date shall have 'FF' in each octet, and Sequence number time stamps shall have the value 0 if no event notification of that type has been generated since the object was created. This property is required if intrinsic reporting is supported by this object.

12.X.22 Silenced

This property, of type BACnetSilencedState, shall indicate whether the most recently occurring transition for this object that has produced an audible or visual indication has been silenced by the receipt of a LifeSafetyOperation service request or a local process.

12.X.23 Operation_Expected

The Operation_Expected property, of type BACnetLifeSafetyOperation, shall specify the next operation expected by this object to handle a specific life safety situation.

12.X.24 Maintenance_Required

This optional property, of type BACnetMaintenance, shall indicate the type of maintenance required for the life safety point. This may be periodic maintenance, or a "parameter-determined" maintenance, such as dirtiness value for an associated detector, and shall be determined locally.

12.X.25 Setting

This optional property, of type Unsigned8, shall be used to convey the desired setting of the input(s) or process used to determine the logical state of the Present_Value. The range of the Setting property shall be from 0 (least sensitive) to 100 (most sensitive). The interpretation of the setting and the actual number of useful settings for a given Life Safety Point object shall be a local matter.

12.X.26 Direct_Reading

This optional property, of type Real, shall indicate an analog quantity that reflects the measured or calculated reading from an initiating device. The manner in which this reading is used to determine the logical state of the object shall be a local matter. If this property is present, then the Units property shall also be present.

12.X.27 Units

This optional property, of type BACnetEngineeringUnits, shall indicate the units of the quantity represented by the Direct_Reading property. If this property is present, then the Direct_Reading property shall also be present.

12.X.28 Member_Of

This optional property, of type List of BACnetDeviceObjectReference, shall indicate those Life Safety Zone objects of which this Life Safety Point object is considered to be a zone member. Each object in the Member_Of list shall be a Life Safety Zone object.

This property may be restricted to only support references to objects inside of the device containing the Life Safety Point object. If the property is writable and is restricted to referencing objects within the containing device, an attempt to write a reference to an object outside the containing device into this property shall cause a Result(-) to be returned with an error class of PROPERTY and an error code of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

[Add to **21**, BACnetConfirmedServiceChoice, p. 350]

(under Alarm and Event Services)
lifeSafetyOperation (27),

(under Security Services)
-- Services added after 1995
-- readRange (26), see Object Access Services
-- lifeSafetyOperation (27) see Alarm and Event Services

[Add to **21**, BACnet-Confirmed-Service-Request, p. 351]

(under Alarm and Event Services)
lifeSafetyOperation [27] LifeSafetyOperation-Request,

(under Security Services)
-- Services added after 1995
-- readRange [26], see Object Access Services
-- lifeSafetyOperation [27] see Alarm and Event Services

[Add to **21**, under Confirmed Alarm and Event Services, pp. 352-354]

LifeSafetyOperation-Request *::= SEQUENCE {*

requestingProcessIdentifier	*[0]*	*Unsigned32,*
requestingSource	*[1]*	*CharacterString,*
request	*[2]*	*BACnetLifeSafetyOperation,*
objectIdentifier	*[3]*	*BACnetObjectIdentifier OPTIONAL*

}

[Add to **21**, BACnetError, p. 361]

(under Alarm and Event Services)
LifeSafetyOperation [27] Error,

(under Security Services)
-- Services added after 1995
-- readRange [26] see Object Access Services
-- lifeSafetyOperation [27] see Alarm and Event Services

[Add to **21**, BACnetEngineeringUnits production, pp. 367-370]

```
BACnetEngineeringUnits ::= ENUMERATED {
 --Other
        percent-obscuration-per-foot        (143),
        percent-obscuration-per-meter       (144),
        …
        }
```

-- The last enumeration used in this version in ~~142~~ *144*.

[Change to **21**, BACnetPropertyIdentifier, p. 378]

```
BACnetPropertyIdentifier ::= ENUMERATED {
        …
        device-type                         (31),
        direct-reading                      (156),
        effective-period                    (32),
        …
        last-restore-time                   (157),
        life-safety-alarm-values            (166),
        ...
        low-limit                           (59),
        maintenance-required                (158),
        manipulated-variable-reference      (60),
        …
        max-pres-value                      (65),
        member-of                           (159),
        minimum-off-time                    (66),
        …
        min-pres-value                      (69),
        mode                                (160),
        model-name                          (70),
        …
        object-type                         (79),
        operation-expected                  (161),
        optional                            (80),
        …
        setpoint-reference                  (109),
        setting                             (162),
        silenced                            (163),
        start-time                          (142),
        …
        total-record-count                  (145),
        tracking-value                      (164),
        units                               (117),
        …
        window-samples                      (147),
        zone-members                        (165),
        …
        -- see variance-value               (151),
        -- see active-cov-subscriptions     (152),
        -- see backup-failure-timeout       (153),
        -- see configuration-files          (154),
        -- see database-revision            (155),
```

-- see direct-reading *(156),*
-- see last-restore-time, *(157),*
-- see maintenance-required *(158),*
-- see member-of *(159),*
-- see mode *(160),*
-- see operation-expected *(161),*
-- see setting *(162),*
-- see silenced *(163),*
-- see tracking-value *(164),*
-- see zone-members *(165),*
-- see life-safety-alarm-values *(166),*

}
-- The highest enumeration used in this version is ~~151~~ *166*.

[Add to **21**, BACnetServicesSupported, p. 379]

(under Alarm and Event Services)
lifeSafetyOperation *(37),*

(under Unconfirmed Services)
-- Services added after 1995
-- utcTimeSynchronization *(36),* *see Object Access Services*
-- lifeSafetyOperation *(37)* *see Alarm and Event Services*
}

[Change **Annex A**, p. 408]

Annex A, Standard Object Types Supported:

Life Safety Point ☐ ☐ ☐ _____ _____

[Add the following to **Annex C**, p. 415]

LIFE-SAFETY-POINT ::= SEQUENCE {

object-identifier	[75]	BACnetObjectIdentifier,
object-name	[77]	CharacterString,
object-type	[79]	BACnetObjectType,
present-value	[85]	BACnetLifeSafetyState,
tracking-value	[164]	BACnetLifeSafetyState OPTIONAL,
description	[28]	CharacterString OPTIONAL,
device-type	[31]	CharacterString OPTIONAL,
status-flags	[111]	BACnetStatusFlags,
event-state	[36]	BACnetEventState,
reliability	[103]	BACnetReliability,
out-of-service	[81]	BOOLEAN,
mode	[160]	BACnetLifeSafetyMode,
time-delay	[113]	Unsigned OPTIONAL,
notification-class	[17]	Unsigned OPTIONAL,
life-safety-alarm-values	[166]	SEQUENCE OF BACnetLifeSafetyState OPTIONAL,
alarm-values	[7]	SEQUENCE OF BACnetLifeSafetyState OPTIONAL,
fault-values	[39]	SEQUENCE OF BACnetLifeSafetyState OPTIONAL,
event-enable	[35]	BACnetEventTransitionBits OPTIONAL,
acked-transitions	[0]	BACnetEventTransitionBits OPTIONAL,
notify-type	[72]	BACnetNotifyType OPTIONAL,,
event-time-stamps	[130]	SEQUENCE OF BACnetTimeStamp OPTIONAL,

```
                                    --accessed as a BACnetARRAY
    silenced                [163]   BACnetSilencedState,
    operation-expected      [161]   BACnetLifeSafetyOperation,
    maintenance-required    [158]   BACnetMaintenance OPTIONAL,
    setting                 [162]   Unsigned8 OPTIONAL,
    direct-reading          [156]   REAL OPTIONAL,
    units                   [117]   BACnetEngineeringUnits OPTIONAL,
    member-of               [159]   SEQUENCE OF BACnetDeviceObjectReference OPTIONAL
    }
```

[Add the following to **Annex D**, p. 428]

Annex D:

D.21 Example of a Life Safety Point object

In this example, a smoke detector is represented as a Life Safety Point object.

```
Property:   Object_Identifier =          (Life Safety Point, Instance 2)
Property:   Object_Name =                "SMK3W"
Property:   Object_Type =                LIFE_SAFETY_POINT
Property:   Present_Value =              PREALARM
Property:   Tracking_Value =             PREALARM
Property:   Description =                "Floor 3, West Zone Smoke Detector"
Property:   Device_Type =                "Old Smokey model 123"
Property:   Status_Flags =               {TRUE, FALSE, FALSE, FALSE}
Property:   Event_State =                LIFE_SAFETY_ALARM
Property:   Reliability =                NO_FAULT_DETECTED
Property:   Out_Of_Service =             FALSE
Property:   Mode =                       ON
Property:   Time_Delay =                 10
Property:   Notification_Class =         39
Property:   Life_Safety_Alarm_Values =   (ALARM)
Property:   Alarm_Values =               (PREALARM)
Property:   Fault_Values =               (FAULT)
Property:   Event_Enable =               {TRUE, TRUE, TRUE}
Property:   Acked_Transitions =          {TRUE, TRUE, TRUE}
Property:   Notify_Type =                ALARM
Property:   Event_Time_Stamps =          ((23-MAR-95, 18:50:21.2),
                                          (*-*-*, *:*:*.*),
                                          (23-MAR-95, 19:01:34.0))
Property:   Silenced =                   SILENCE_AUDIBLE
Property:   Operation_Expected =         RESET_ALARM
Property:   Maintenance_Required =       NONE
Property:   Setting =                    50
Property:   Direct_Reading =             84.3
Property:   Units =                      PERCENT-OBSCURATION-PER-METER
Property:   Member_Of =                  ((Life Safety Zone, Instance 5))
```

[Addition to **E.1**, p. 437]

E.1.X Example of the LifeSafetyOperation Service

This example illustrates the use of the LifeSafetyOperation service to reset a trouble condition.

Service =	*LifeSafetyOperation*
'Requesting Process Identifier' =	*18*
'Requesting Source' =	*"MDL"*
'Request' =	*RESET*
'Object Identifier' =	*(Life Safety Point, 1)*

[Addition to **F.1**, p. 459]

F.1.X Encoding for Example E.1.X - LifeSafetyOperation

X'00'	*PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)*
X'02'	*Maximum APDU Size Accepted=206 octets*
X'0F'	*Invoke ID=15*
X'1B'	*Service Choice=27 (LifeSafetyOperation-Request)*
X'09'	*SD Context Tag 0 (Requesting Process Identifier, L=1)*
X'12'	*18*
X'1C'	*SD Context Tag 1 (Monitored Object Identifier, L=4)*
X'00'	*ANSI X3.4 Encoding*
X'4D444C'	*"MDL"*
X'29'	*SD Context Tag 2 (Request, L=1)*
X'04'	*4 (RESET)*
X'3C'	*SD Context Tag 3 (Object Identifier, L=4)*
X'05400001'	*Life Safety Point, Instance Number = 1*

Assuming the service procedure executes correctly, a simple acknowledgement is returned:

X'20'	*PDU Type=2 (BACnet-SimpleACK-PDU)*
X'0F'	*Invoke ID=15*
X'1B'	*Service ACK Choice=27 (LifeSafetyOperation)*

135*c*-2. Add a new Life Safety Zone object type that represents the characteristics associated with an arbitrary group of BACnet Life Safety Point and Life Safety Zone objects.

12.X Life Safety Zone Object Type

The Life Safety Zone object type defines a standardized object whose properties represent the externally visible characteristics associated with an arbitrary group of BACnet Life Safety Point and Life Safety Zone objects in fire, life safety and security applications. The condition of a Life Safety Zone object is represented by a *mode* and a *state*.

Mode changes determine the object's inner logic and, consequently, influence the evaluation of the state. Typically, the operating *mode* would be under operator control.

The *state* of the object represents the condition of the controller according to the logic internal to the device. The implementation of the logic applied to such controllers to determine the various possible states is a local matter.

Typical applications of the Life Safety Zone object include fire zones, panel zones, detector lines, extinguishing controllers, remote transmission controllers, etc. Similar objects can be identified in security control panels.

The Life Safety Zone object type and its properties are summarized in Table 12-X and described in detail in this subclause.

NOTE: Do not confuse the Present_Value state with the Event_State property, which reflects the offnormal state of the Life Safety Zone object.

Table 12-X. Properties of the Life Safety Zone Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	BACnetLifeSafetyState	R[1]
Tracking_Value	BACnetLifeSafetyState	O
Description	CharacterString	O
Device_Type	CharacterString	O
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Reliability	BACnetReliability	R[1]
Out_Of_Service	BOOLEAN	R
Mode	BACnetLifeSafetyMode	W
Time_Delay	Unsigned	O[2]
Notification_Class	Unsigned	O[2]
Life_Safety_Alarm_Values	List of BACnetLifeSafetyState	O[2]
Alarm_Values	List of BACnetLifeSafetyState	O[2]
Fault_Values	List of BACnetLifeSafetyState	O[2]
Event_Enable	BACnetEventTransitionBits	O[2]
Acked_Transitions	BACnetEventTransitionBits	O[2]
Notify_Type	BACnetNotifyType	O[2]
Event_Time_Stamps	BACnetARRAY [3] of BACnetTimeStamp	O[2]
Silenced	BACnetSilencedState	R
Operation_Expected	BACnetLifeSafetyOperation	R
Maintenance_Required	BOOLEAN	O
Zone_Members	List of BACnetDeviceObjectReference	R
Member_Of	List of BACnetDeviceObjectReference	O

[1] These properties are required to be writable when Out_Of_Service is TRUE.
[2] These properties are required if the object supports intrinsic alarming.

12.X.1 Object_Identifier

This property, of type BACnetObjectIdentifier, is a numeric code that is used to identify the object. It shall be unique within the BACnet Device that maintains it.

12.X.2 Object_Name

This property, of type CharacterString, shall represent a name for the object that is unique within the BACnet Device that maintains it. The minimum length of the string shall be one character. The set of characters used in the Object_Name shall be restricted to printable characters.

12.X.3 Object_Type

This property, of type BACnetObjectType, indicates membership in a particular object type class. The value of this property shall be LIFE_SAFETY_ZONE.

12.X.4 Present_Value

This property, of type BACnetLifeSafetyState, reflects the state of the Life Safety Zone object. The means of deriving the Present_Value shall be a local matter. Present_Value may latch non-NORMAL state values until reset. The Present_Value property shall be writable when Out_Of_Service is TRUE (see 12.X.11).

12.X.5 Tracking_Value

This optional property, of type BACnetLifeSafetyState, reflects the non-latched state of the Life Safety Zone object. The means of deriving the state shall be a local matter. Unlike Present_Value, which may latch non-NORMAL state values until reset, Tracking_Value shall continuously track changes in the state.

12.X.6 Description

This optional property, of type CharacterString, is a string of printable characters whose content is not restricted.

12.X.7 Device_Type

This optional property, of type CharacterString, is a text description of the physical zone or area that the Life Safety Zone object represents. It will typically be used to describe the locale of the Life Safety Point objects that are Zone_Members of the Life Safety Zone object.

12.X.8 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of the Life Safety Zone object. Three of the flags are associated with the values of other properties of this object. A more detailed status could be determined by reading the properties that are linked to these flags. The relationship between individual flags is not defined by the protocol. The four flags are

{IN_ALARM, FAULT, OVERRIDDEN, OUT_OF_SERVICE}

where:

IN_ALARM	Logical FALSE (0) if the Event_State property (12.X.9) has a value of NORMAL, otherwise logical TRUE (1).
FAULT	Logical TRUE (1) if the Reliability property (12.X.10) does not have a value of NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	Logical TRUE (1) if the point has been overridden by some mechanism local to the BACnet Device. In this context "overridden" is taken to mean that the physical input(s) are no longer tracking changes to the Present_Value property and the Reliability property is no longer a reflection of the physical input(s).
OUT_OF_SERVICE	Logical TRUE (1) if the Out_Of_Service property (12.X.11) has a value of TRUE, otherwise logical FALSE (0).

12.X.9 Event_State

The Event_State property, of type BACnetEventState, is included in order to provide a way to determine if this object has an active event state associated with it. The Event_State property shall indicate the event state of the object.

12.X.10 Reliability

The reliability property, of type BACnetReliability, provides an indication of whether the Present_Value or the operation of the physical input(s) in question are "reliable" as far as the BACnet Device or operator can determine and, if not, why. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, OPEN_LOOP, SHORTED_LOOP, MULTI_STATE_FAULT, UNRELIABLE_OTHER}.

12.X.10.1 Conditions for Generating a TO-FAULT Event

A TO-FAULT event is generated under these conditions:

(a) the TO-FAULT flag must be enabled in the Event_Enable property, and
(b) the Present_Value must equal at least one of the values in the Fault_Values list.

12.X.11 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the input(s) or process the object represents is not in service. This means that changes to the Present_Value property are decoupled from the input(s) or process when the value of Out_Of_Service is TRUE. In addition, the Reliability property and the corresponding state of the FAULT flag of the Status_Flags property shall be decoupled when Out_Of_Service is TRUE. While the Out_Of_Service property is TRUE, the Present_Value and Reliability properties may be changed to any value as a means of simulating specific fixed conditions or for testing purposes. Other functions that depend on the state of the Present_Value or Reliability properties shall respond to changes made to these properties while Out_Of_Service is TRUE, as if those changes had occurred to the input(s) or process.

12.X.12 Mode

This writable property, of type BACnetLifeSafetyMode, shall convey the desired operating mode for the Life Safety Zone object. The Life Safety Zone object shall generate CHANGE_OF_LIFE_SAFETY event notifications for any mode transition if the respective flags TO-OFFNORMAL, TO-FAULT or TO_NORMAL are set in the Event_Enable property (see 12.X.15.1, 12.X.15.2, 12.X.16.1, 12.X.16.2, 12.X.17.1, and 12.X.17.2).

12.X.13 Time_Delay

This optional property, of type Unsigned, shall specify the minimum period of time in seconds that the Present_Value must remain:

(a) equal to any of the values in the Life_Safety_Alarm_Values property before a TO-OFFNORMAL event is generated, or
(b) equal to any of the values in the Alarm_Values property before a TO-OFFNORMAL event is generated, or
(c) not equal to any of the values in the Life_Safety_Alarm_Values property, Alarm_Values property, or Fault_Values property before a TO-NORMAL event is generated.

This property is required if intrinsic reporting is supported by this object.

12.X.14 Notification_Class

This optional property, of type Unsigned, shall specify the notification class to be used when handling and generating event notifications for this object. The Notification_Class property implicitly refers to a Notification Class object that has a Notification_Class property with the same value.

12.X.15 Life_Safety_Alarm_Values

This optional property, of type List of BACnetLifeSafetyState, shall specify any states the Present_Value must equal before a TO-OFFNORMAL event is generated and event state LIFE_SAFETY_ALARM is entered. This property is required if intrinsic reporting is supported by this object.

12.X.15.1 Conditions for Generating a TO-OFFNORMAL Event

A TO-OFFNORMAL event is generated under these conditions:

(a) the TO-OFFNORMAL flag must be enabled in the Event_Enable property, and
(b) the Present_Value must equal any of the values in the Life_Safety_Alarm_Values list, and
(c) the Present_Value must remain within the Life_Safety_Alarm_Values list for a minimum period of time, specified by the Time_Delay property.

New events shall be generated upon a change of Mode if the TO-OFFNORMAL flag is set in the Event_Enable property.

12.X.15.2 Conditions for Generating a TO-NORMAL Event

Once equal, the Present_Value must become not equal to any of the states in the Life_Safety_Alarm_Values property, and not equal to any of the states in the Alarm_Values property, and not equal to any of the states in the Fault_Values property, before a TO-NORMAL event is generated under these conditions:

(a) the TO-NORMAL flag must be enabled in the Event_Enable property, and
(b) the Present_Value must remain not equal to any of the states in the Life_Safety_Alarm_Values property, and
(c) the Present_Value must remain not equal to any of the states in the Alarm_Values property, and
(d) the Present_Value must remain not equal to any of the states in the Fault_Values property, and
(e) the Present_Value must remain equal to the same value for a minimum period of time, specified by the Time_Delay property.

New events shall be generated upon a change of Mode if the TO-NORMAL flag is set in the Event_Enable property.

12.X.16 Alarm_Values

This optional property, of type List of BACnetLifeSafetyState, shall specify any states the Present_Value must equal before a TO-OFFNORMAL event is generated and event state OFFNORMAL is entered. This property is required if intrinsic reporting is supported by this object.

12.X.16.1 Conditions for Generating a TO-OFFNORMAL Event

A TO-OFFNORMAL event is generated under these conditions:

(a) the TO-OFFNORMAL flag must be enabled in the Event_Enable property, and
(b) the Present_Value must equal at least one of the values in the Alarm_Values list, and
(c) the Present_Value must remain equal to the same value for a minimum period of time, specified by the Time_Delay property.

New events shall be generated upon a change of Mode if the TO-OFFNORMAL flag is set in the Event_Enable property.

12.X.16.2 Conditions for Generating a TO-NORMAL Event

Conditions for generating a TO-NORMAL event are defined in 12.X.15.2.

12.X.17 Fault_Values

This optional property, of type List of BACnetLifeSafetyState, shall specify any states the Present_Value must equal before a TO-FAULT event is generated. If Present_Value becomes equal to any of the states in the Fault_Values list, and no physical fault has been detected for any inputs that the Present_Value represents, then the Reliability property shall have the value MULTI_STATE_FAULT. The Fault_Values property is required if intrinsic reporting is supported by this object.

New events shall be generated upon a change of Mode if the TO-FAULT flag is set in the Event_Enable property.

12.X.17.1 Conditions for Generating a TO-FAULT Event

A TO-FAULT event is generated under these conditions:

(a) the TO-FAULT flag must be enabled in the Event_Enable property, and
(b) the Present_Value must equal at least one of the values in the Fault_Values list.

12.X.17.2 Conditions for Generating a TO-NORMAL Event

Conditions for generating a TO-NORMAL event are defined in 12.X.15.2.

12.X.18 Event_Enable

This optional property, of type BACnetEventTransitionBits, shall convey three flags that separately enable and disable reporting of TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events that are based on Present_Value or Mode changes.

12.X.19 Acked_Transitions

This optional property, of type BACnetEventTransitionBits, shall convey three flags that separately indicate the receipt of acknowledgements for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events. These flags shall be cleared upon the occurrence of the corresponding event and set under any of these conditions:

(a) upon receipt of the corresponding acknowledgement;
(b) upon the occurrence of the event if the corresponding flag is not set in the Event_Enable property (meaning event notifications will not be generated for this condition and thus no acknowledgement is expected);
(c) upon the occurrence of the event if the corresponding flag is set in the Event_Enable property and the corresponding flag in the Ack_Required property of the Notification Class object implicitly referenced by the Notification_Class property of this object is not set (meaning no acknowledgement is expected).

12.X.20 Notify_Type

This optional property, of type BACnetNotifyType, shall convey whether the notifications generated by the object should be Events or Alarms.

12.X.21 Event_Time_Stamps

This optional parameter, of type BACnetARRAY[3] of BACnetTimeStamp, shall convey the times of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events, respectively. Time stamps of type Time or Date shall have 'FF' in each octet, and Sequence number time stamps shall have the value 0 if no event notification of that type has been generated since the object was created. This property is required if intrinsic reporting is supported by this object.

12.X.22 Silenced

This property, of type BACnetSilencedState, shall indicate whether the most recently occurring transition for this object that has produced an audible or visual indication has been silenced by the receipt of a LifeSafetyOperation service request or a local process.

12.X.23 Operation_Expected

The Operation_Expected property, of type BACnetLifeSafetyOperation, shall specify the next operation expected by this object to handle a specific life safety situation.

12.X.24 Maintenance_Required

This optional property, of type BOOLEAN, shall indicate that maintenance is required for one or more of the life safety points that are members of this zone.

12.X.25 Zone_Members

This property, of type List of BACnetDeviceObjectReference, shall indicate which Life Safety Point and Life Safety Zone objects are members of the zone represented by this object.

This property may be restricted to only support references to objects inside of the device containing the Life Safety Zone object. If the property is writable and is restricted to referencing objects within the containing device, an attempt to write a reference to an object outside the containing device into this property shall cause a Result(-) to be returned with an error class of PROPERTY and an error code of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

12.X.26 Member_Of

This optional property, of type List of BACnetDeviceObjectReference, shall indicate those Life Safety Zone objects of which this Life Safety Zone object is considered to be a zone member. Each object in the Member_Of list shall be a Life Safety Zone object.

This property may be restricted to only support references to objects inside of the device containing the Life Safety Zone object. If the property is writable and is restricted to referencing objects within the containing device, an attempt to write a reference to an object outside the containing device into this property shall cause a Result(-) to be returned with an error class of PROPERTY and an error code of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

[Change **Annex A**, p. 408]

Annex A, Standard Object Types Supported:

Life Safety Zone ☐ ☐ ☐ _____ _____

[Add the following to **Annex C**, p. 415]

LIFE-SAFETY-ZONE ::= SEQUENCE {

object-identifier	[75]	BACnetObjectIdentifier,
object-name	[77]	CharacterString,
object-type	[79]	BACnetObjectType,
present-value	[85]	BACnetLifeSafetyState,
tracking-value	[164]	BACnetLifeSafetyState OPTIONAL,
description	[28]	CharacterString OPTIONAL,
device-type	[31]	CharacterString OPTIONAL,
status-flags	[111]	BACnetStatusFlags,
event-state	[36]	BACnetEventState,
reliability	[103]	BACnetReliability,
out-of-service	[81]	BOOLEAN,
mode	[160]	BACnetLifeSafetyMode,
time-delay	[113]	Unsigned OPTIONAL,
notification-class	[17]	Unsigned OPTIONAL,
life-safety-alarm-values	[166]	SEQUENCE OF BACnetLifeSafetyState OPTIONAL,
alarm-values	[7]	SEQUENCE OF BACnetLifeSafetyState OPTIONAL,
fault-values	[39]	SEQUENCE OF BACnetLifeSafetyState OPTIONAL,
event-enable	[35]	BACnetEventTransitionBits OPTIONAL,
acked-transitions	[0]	BACnetEventTransitionBits OPTIONAL,
notify-type	[72]	BACnetNotifyType OPTIONAL,
event-time-stamps	[130]	SEQUENCE OF BACnetTimeStamp OPTIONAL, – accessed as a BACnetARRAY
silenced	[163]	BACnetSilencedState,
operation-expected	[161]	BACnetLifeSafetyOperation,
maintenance-required	[158]	BOOLEAN OPTIONAL,
zone-members	[165]	SEQUENCE OF BACnetDeviceObjectReference,
member-of	[159]	SEQUENCE OF BACnetDeviceObjectReference OPTIONAL

}

[Add the following to **Annex D**, p. 428]

Annex D:

D.21 Example of a Life Safety Zone Object

In this example, a fire zone is represented as a Life Safety Zone object.

Property:	Object_Identifier =	(Life Safety Zone, Instance 2)
Property:	Object_Name =	"SMK3"
Property:	Object_Type =	LIFE_SAFETY_ZONE
Property:	Present_Value =	PREALARM
Property:	Tracking_Value =	PREALARM
Property:	Description =	"Floor 3 Smoke"
Property:	Status_Flags =	{TRUE FALSE, FALSE, FALSE}
Property:	Event_State =	LIFE_SAFETY_ALARM

Property:	Reliability =	NO_FAULT_DETECTED
Property:	Out_Of_Service =	FALSE
Property:	Mode =	ON
Property:	Time_Delay =	10
Property:	Notification_Class =	39
Property:	Life_Safety_Alarm_Values =	(ALARM)
Property:	Alarm_Values =	(PREALARM)
Property:	Fault_Values =	(FAULT)
Property:	Event_Enable =	{TRUE, TRUE, TRUE}
Property:	Acked_Transitions =	{TRUE, TRUE, TRUE}
Property:	Notify_Type =	ALARM
Property:	Event_Time_Stamps =	((23-MAR-95, 18:50:21.2), (*-*-*, *:*:*.*), (23-MAR-95,19:01:34.0))
Property:	Silenced =	UNSILENCED
Property:	Operation_Expected =	SILENCE_AUDIBLE
Property:	Maintenance_Required =	NONE
Property:	Zone_Members =	((Life Safety Point, Instance 22), (Life Safety Point, Instance 23))
Property:	Member_Of =	((Life Safety Zone, Instance 5))

135*c*-3. Add functionality to the existing BACnet alarm and event features needed to support the Life Safety Point and Life Safety Zone object types.

[Change **12.10.5**, p. 182]

12.10.5 Event_Type
…This parameter is an enumerated type that may have any of the following values:

{CHANGE_OF_BITSTRING, CHANGE_OF_STATE, CHANGE_OF_VALUE, COMMAND_FAILURE, FLOATING_LIMIT, OUT_OF_RANGE, BUFFER_READY, *CHANGE_OF_LIFE_SAFETY*}

[Change **Table 12-13:** to add a new Life Safety entry]

Event type	Event State	Event Parameters
CHANGE_OF_LIFE_SAFETY	*NORMAL,* *OFFNORMAL,* *LIFE_SAFETY_ALARM* …	*Time_Delay* *List_Of_Alarm_Values* *List_Of_Life_Safety_Alarm_Values* *Mode_Property_Reference*

[Change **12.10.7**, p. 183 to add these parameters]

12.10.7 Event_Parameters

List_Of_Life_Safety_Alarm_Values — *This parameter is a list of BACnetLifeSafetyState that applies to the CHANGE_OF_LIFE_SAFETY algorithm. If the value of the referenced property changes to one of the values in the List_Of_Life_Safety_Alarm_Values, then the Event_State property of the Event Enrollment object makes a transition TO-OFFNORMAL and appropriate notifications are sent. The resulting event state is LIFE_SAFETY_ALARM.*

List_Of_Alarm_Values — *This parameter is a list of BACnetLifeSafetyState that applies to the CHANGE_OF_LIFE_SAFETY algorithm. If the value of the referenced property changes to one of the values in the List_Of_Alarm_Values, then the Event_State property of the Event Enrollment object makes a transition TO-OFFNORMAL and appropriate notifications are sent. The resulting event state is OFFNORMAL.*

Mode_Property_Reference — *This parameter, of type BACnetDeviceObjectPropertyReference, applies to the CHANGE_OF_LIFE_SAFETY algorithm. It identifies the device, object, and property that provides the operating mode of the referenced object providing life safety functionality (normally the Mode property). This parameter may reference only object properties that are of type BACnetLifeSafetyMode.*

[Change the fourth paragraph of **13**, p. 216]

…Any of ~~eight~~ *the* standardized algorithms *described in Clause 13.3* may be used…

[Change **Table 13-1**, p. 218]

Add Life Safety Point and Life Safety Zone to object types Binary Input, Binary Output, Binary Value, Multi-state Input, Multi-state Output.

[Change **13.2**, p. 219, add a new paragraph after the last paragraph]

In the case of Life Safety Zone and Life Safety Point, the Life_Safety_Alarm_Values property lists each of the possible Present_Value states that shall be interpreted as LIFE_SAFETY_ALARM. The Alarm_Values property lists each of the possible Present_Value states that shall be interpreted as OFFNORMAL. The Fault_Values property lists each of the possible Present_Value states that shall be interpreted as FAULT. All other Present_Value states shall be interpreted as NORMAL. Transitions to any of the states may generate notifications if the corresponding flags are set in Event_Enable.

[Change **Table 13-2**, p. 220]

Object Type	Criteria	Event Type
Life Safety Point *Life Safety Zone*	*If Present_Value changes to become equal to one of the values in the Life_Safety_Alarm_Values list AND remains equal to a value within the Life_Safety_Alarm_Values list for longer than Time_Delay AND the new transition is enabled in Event-Enable* *OR* *If Present_Value changes to become equal to one of the values in the Alarm_Values list AND remains equal to a value within the Alarm_Values list for longer than Time_Delay AND the new transition is enabled in Event-Enable* *OR* *Mode changes*	*CHANGE_OF_LIFE_SAFETY*

[Change **Table 13-3**, p. 221]

Object	Event Type	Notification Parameters	Referenced Object's Properties
Life Safety Point *Life Safety Zone*	*CHANGE_OF_LIFE_SAFETY*	*New_State* *New_Mode* *Status_Flags* *Operation_Expected*	*Present_Value* *Mode* *Status_Flags* *Operation_Expected*

[Change **Table 13-4**, p. 221]

Event Type	Notification Parameters	Description
CHANGE_OF_LIFE_SAFETY	*New_State* *New_Mode* *Status_Flags* *Operation_Expected*	*The new value of the referenced property* *The new mode of the referenced object* *The Status_Flags of the referenced object* *The next operation requested by the referenced object*

[Change **13.3**, p. 222]

1st paragraph

…Any of ~~eight~~ *the* standardized algorithms *described in this clause*…

3rd paragraph

…~~Eight~~ *The following* event type algorithms are specified because of their widespread occurrence in building automation systems: ~~They are:…~~

…*(h) CHANGE_OF_LIFE_SAFETY*

[Add **13.3.X**]

13.3.X CHANGE_OF_LIFE_SAFETY Algorithm

A CHANGE_OF_LIFE_SAFETY occurs when the value of the referenced property becomes equal to any of the values in the List_Of_Life_Safety_Alarm_Values, and remains within the set of values in this list for Time_Delay seconds. The resulting event state is LIFE_SAFETY_ALARM.

A CHANGE_OF_LIFE_SAFETY also occurs when the value of the referenced property becomes equal to any of the values in the List_Of_Alarm_Values, and remains within the set of values in this list for Time_Delay seconds. The resulting event state is OFFNORMAL.

A CHANGE_OF_LIFE_SAFETY also occurs for any change of the property referred to by the Mode_Property_Reference.

For the purpose of event notification, CHANGE_OF_LIFE_SAFETY events generate a TO-OFFNORMAL transition.

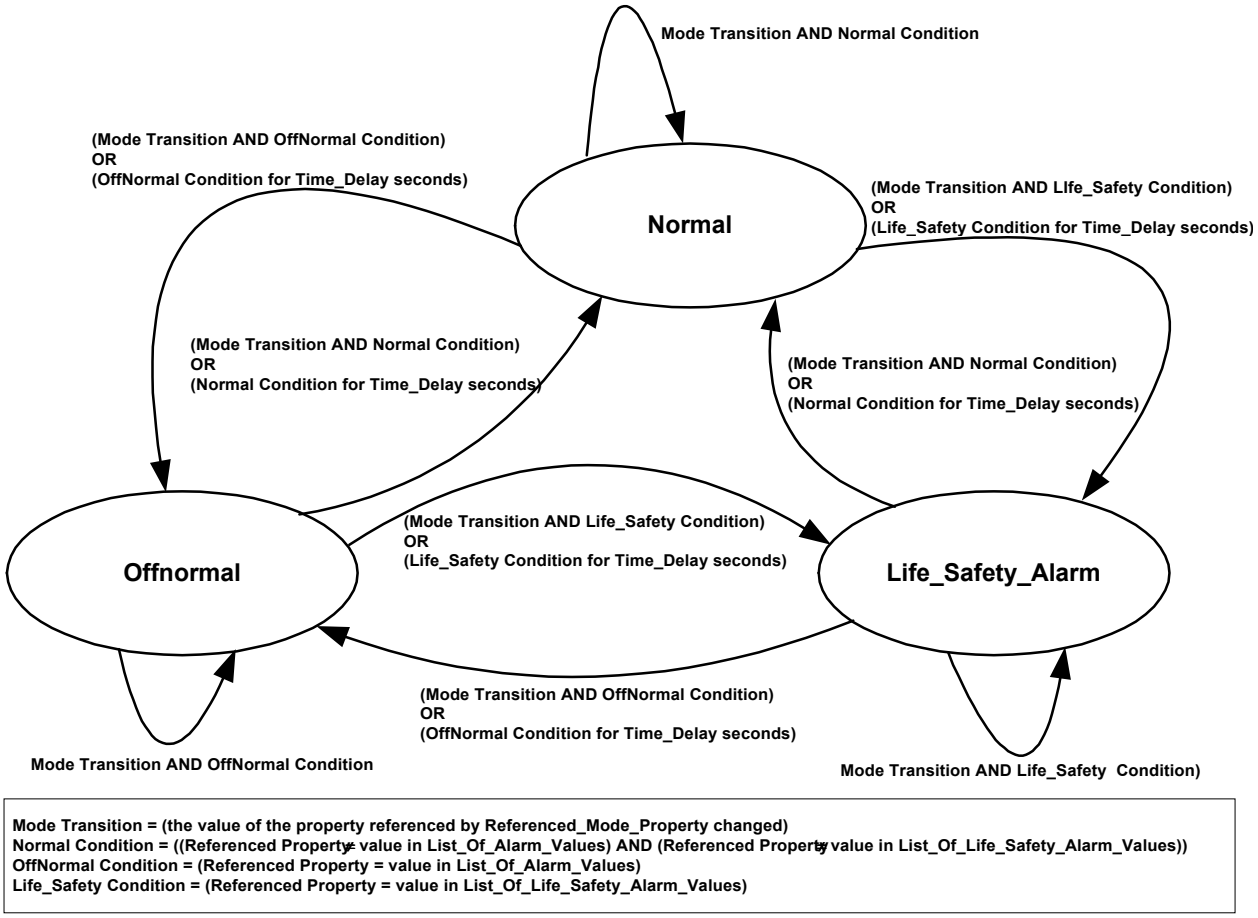

Figure 13-X. CHANGE_OF_LIFE_SAFETY algorithm.

[Change **Table 23-1**, p. 391]

Enumeration Name	Reserved Range	Maximum Value
...		
BACnetLifeSafetyState	*0-255*	*65535*
BACnetLifeSafetyMode	*0-255*	*65535*
BACnetSilencedState	*0-63*	*65535*
BACnetLifeSafetyOperation	*0-63*	*65535*
BACnetMaintenance	*0-255*	*65535*

[Changes to **21**]

BACnetObjectType ::= ENUMERATED {
. . .
life-safety-point *(21),*
life-safety-zone *(22),*

BACnetObjectTypesSupported ::= BIT STRING {
. . .
life-safety-point (21),
life-safety-zone (22),
}

BACnetLifeSafetyState ::= ENUMERATED {
quiet (0),
pre-alarm (1),
alarm (2),
fault (3),
fault-pre-alarm (4),
fault-alarm (5),
not-ready, (6),
active (7),
tamper (8),
test-alarm (9),
test-active (10),
test-fault (11),
test-fault-alarm (12),
holdup (13),
duress (14),
tamper-alarm (15),
abnormal (16),
emergency-power (17),
delayed (18),
blocked (19),
local-alarm (20),
general-alarm (21),
supervisory (22),
test-supervisory (23),
...
}
-- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
-- 256-65535 may be used by others subject to procedures and constraints described in Clause 23.

BACnetMaintenance ::= ENUMERATED {
none (0),
periodic-test (1),
need-service-operational (2),
need-service-inoperative (3),
...
}
-- Enumerated values 0-~~63~~ 255 are reserved for definition by ASHRAE. Enumerated values
-- 256-65535 may be used by others subject to procedures and constraints described in
-- Clause 23.

BACnetSilencedState ::= ENUMERATED {
unsilenced (0),
audible-silenced (1),
visible-silenced (2),
all-silenced (3),
...
}
-- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
-- 64-65535 may be used by others subject to procedures and constraints described in

-- Clause 23.

BACnetLifeSafetyMode ::= ENUMERATED {

off	*(0),*
on	*(1),*
test	*(2),*
manned	*(3),*
unmanned	*(4),*
armed	*(5),*
disarmed	*(6),*
prearmed	*(7),*
slow	*(8),*
fast	*(9),*
disconnected	*(10),*
enabled	*(11),*
disabled	*(12),*
automatic-release-disabled	*(13),*
default	*(14),*
...	
}	

-- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
-- 256-65535 may be used by others subject to procedures and constraints described in
-- Clause 23.

BACnetDeviceObjectReference ::= SEQUENCE {

deviceIdentifier [0] BACnetObjectIdentifier OPTIONAL,
objectIdentifier [1] BACnetObjectIdentifier
}

BACnetLifeSafetyOperation ::= ENUMERATED {

none	*(0),*
silence	*(1),*
silence-audible	*(2),*
silence-visual	*(3),*
reset	*(4),*
reset-alarm	*(5),*
reset-fault	*(6),*
...	
}	

-- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
-- 64-65535 may be used by others subject to procedures and constraints described in
-- Clause 23.

BACnetEventState ::= ENUMERATED {

…
life-safety-alarm (5),
}

-- The last enumeration used in this version is 5.

BACnetEventType ::= ENUMERATED {

…
complex-event-type (6),
buffer-ready (7),
change-of-life-safety (8),
…
}

-- The last enumeration used in this version is 8.

BACnetEventParameter ::= CHOICE {

…
change-of-life-safety [8] SEQUENCE {
time-delay [0] Unsigned,
list-of-life-safety-alarm-values [1] SEQUENCE OF BACnetLifeSafetyState,
list-of-alarm-values [2] SEQUENCE OF BACnetLifeSafetyState,
mode-property-reference [3] BACnetDeviceObjectPropertyReference

}
}

BACnetNotificationParameters ::= CHOICE {

…
change-of-life-safety [8] SEQUENCE {
new-state [0] BACnetLifeSafetyState,
new-mode [1] BACnetLifeSafetyMode,
status-flags [2] BACnetStatusFlags,
operation-expected [3] BACnetLifeSafetyOperation
}
}

BACnetPropertyStates ::= CHOICE {

…
life-safety-mode [12] BACnetLifeSafetyMode
life-safety-state [13] BACnetLifeSafetyState,
}

135*c*-4. Add a new LifeSafetyOperation service that provides silence and reset capabilities needed for life safety systems.

13.X LifeSafetyOperation Service

The LifeSafetyOperation service is intended for use in fire, life safety and security systems to provide a mechanism for conveying specific instructions from a human operator to:

(a) silence audible or visual notification appliances, or
(b) reset latched notification appliances.

Ensuring that the LifeSafetyOperation request actually comes from a person with appropriate authority is a local matter.

13.X.1 Structure

The structure of the LifeSafetyOperation primitive is shown in Table 13-X. The terminology and symbology used in this table are explained in 5.6.

Table 13-X. Structure of LifeSafetyOperation Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Requesting Process Identifier	M	M(=)		
Requesting Source	M	M(=)		
Request	M	M(=)		
Object Identifier	U	U(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
Error Type			M	M(=)

13.X.1.1 Argument

This parameter shall convey the parameters for the LifeSafetyOperation confirmed service request.

13.X.1.1.1 Requesting Process Identifier

This parameter, of type Unsigned32, specifies an identifying number of significance to the sending device that uniquely identifies the process which initiated the service request. The assignment and meaning of process identifiers shall be a local matter.

13.X.1.1.2 Requesting Source

This parameter, of type CharacterString, specifies the identity of the human operator that initiated the LifeSafetyOperation service request.

13.X.1.1.3 Request

This parameter, of type BACnetLifeSafetyOperation, shall convey the requested operation:

{SILENCE, SILENCE_AUDIBLE, SILENCE_VISUAL, RESET, RESET_ALARM, RESET_FAULT}

13.X.1.1.4 Object Identifier

This parameter, of type BACnetObjectIdentifier, shall convey the specific BACnet object to which the life safety request is directed. If this parameter is not present, then all applicable objects within the receiving BACnet device shall be silenced or reset accordingly based on the 'Request' provided.

13.X.1.2 Result(+)

The 'Result(+)' parameter shall indicate that the service request succeeded.

13.X.1.3 Result(-)

The 'Result(-)' parameter shall indicate that the service request has failed. The reason for the failure shall be specified by the 'Error Type' parameter.

13.X.1.3.1 Error Type

This parameter consists of two component parameters: (1) the 'Error Class' and (2) the 'Error Code'. See Clause 18.

13.X.2 Service Procedure

The responding BACnet-user shall first verify the validity of the 'Object Identifier' parameter and return a 'Result(-)' response with the appropriate error class and code if either the 'Request' is invalid or if the 'Object Identifier' parameter is present and specifies an object that is unknown or does not represent an appropriate request for this object type.

If the 'Object Identifier' parameter is not present, then the responding BACnet-user shall attempt to operate all applicable objects in the device based on the 'Request' parameter. If the 'Object Identifier' parameter is present, the responding BACnet-user shall attempt to silence or reset the object specified in the 'Object Identifier' parameter based on the 'Request' parameter. In either case, the responding BACnet-user shall issue a Result(+) primitive.

135*c*-5. Add a new subclause to 16 to describe the use of existing BACnet services to provide backup and restore capability.

[Add a new normative subclause to **16**]

16.X - Backup and Restore

This clause describes the procedures to be used to backup and restore the configuration of BACnet devices.

16.X.1 The Backup and Restore Procedures

In BACnet building control systems, many devices will have configuration data that is set up by a vendor's proprietary configuration tool. This setup may consist of network visible BACnet objects and/or non-network visible settings. This section outlines the standard method that BACnet devices will employ if an interoperable device backup and restore feature is to be provided.

The backup and restore procedures use File objects to hold and transfer the configuration data. The content and format of the configuration files is a local matter. The choice of whether to use stream-based files or record-based files is a local matter. The services required to support the backup and restore procedures are ReinitializeDevice, ReadProperty, WriteProperty, AtomicWriteFile, AtomicReadFile, and optionally CreateObject.

16.X.2 Backup

For the purposes of this discussion, the device performing the backup procedure will be referred to as device A, and the device being backed up will be device B.

16.X.2.1 Initiation of the Backup Procedure

Device A sends a ReinitializeDevice(STARTBACKUP, <password>) message to device B. Device A will await a response from device B before continuing with the backup procedure.

16.X.2.2 Preparation for Backup

Upon receipt of the ReinitializeDevice(STARTBACKUP, <password>) message, if device B is able to perform a backup procedure, device B will prepare for the backup procedure and respond with a 'Result(+)' to the ReinitializeDevice service request.

If device B is unable to perform a backup procedure or is already performing a backup procedure, then it will respond to the ReinitializeDevice service request with a 'Result(-)' response. Assuming device B supports the backup procedure and the request was properly formulated, the valid Error Class:Error Codes that can be returned are :

DEVICE:CONFIGURATION_IN_PROGRESS - if device B is already processing a backup or a restore request.

SERVICES:SERVICE_REQUEST_DENIED – if the password that was provided was incorrect or if a password is required and one was not provided.

After device B responds to the ReinitializeDevice request with a 'Result(+)', the configuration File objects must exist in the device. It is a local matter as to whether device B will respond to other requests while it is in backup mode. The exception to this is that device B must accept and fulfill read requests by device A that consist of accesses to device B's Device object and/or its configuration File objects. Any services that are rejected due to an in-progress backup procedure will be rejected with an error class of DEVICE and error code of DEVICE_BUSY.

It is a local matter as to whether device B will continue to perform control actions while it is in backup mode. If device B changes its operational behavior during a backup procedure, then the System_Status property of the Device object shall be set to BACKUP_IN_PROGRESS.

16.X.2.3 Loading the Backup Parameters

Upon receipt of a 'Result(+)' response from device B to the ReinitializeDevice(STARTBACKUP, <password>) message, device A will read the Configuration_Files property of the Device object. This property will be used to determine the files to read and in what order the files will be read. The value of the Configuration_Files property is not guaranteed to contain a complete or correct set of configuration File object references before the backup request is accepted by device B.

16.X.2.4 Backing Up the Configuration Files

Once device A has determined the files that make up the device configuration image, device A will determine the type of each file and will use the AtomicReadFile service to read each configuration file from device B. Each file will be read as a stream of bytes, or as a sequence of records depending on the File_Access_Method property of the File object. Stream access files will be read in byte order, and record access files will be read in record order. The files will be read in the same order as they appear in the Configuration_Files property.

It is a local matter as to what device A does with the configuration files, although the intent of the service is to allow an operator to archive the setup of device B such that device B may be restored at a later date should its configuration become corrupt.

It is left up to the implementer of device A as to whether the files read from device B will be made available for examination by tools developed by the implementer of device B. It is recommended that record access files be stored on device A as a sequence of BACnet OCTET STRINGS.

16.X.2.5 Ending the Backup Procedure

When the all of the configuration files have been read, device A sends a ReinitializeDevice(ENDBACKUP, <password>) message to device B. Device B will perform whatever actions are required to complete the backup in order to place the device back into the state it was in before the backup procedure or into any other state as defined by the vendor. Device B must not remain in the BACKUP_IN_PROGRESS mode after the backup procedure has ended.

If device A needs to abort the backup for any reason (i.e., the user aborts the procedure, device B fails to allow reads from a configuration file, or device A detects any other condition that inhibits the backup procedure), device A shall attempt to send ReinitializeDevice(ENDBACKUP, <password>) to device B. Upon receipt of this message, device B shall end the backup procedure. If the backup procedure is aborted, device A should not assume that the configuration files are still valid and continue to read them.

The receipt of the ReinitializeDevice(ENDBACKUP, <password>) message shall cause device B to exit backup mode.

If device B does not receive any messages related to the backup procedure from device A for the number of seconds specified in the Backup_Failure_Timeout property of its Device object, device B should assume that the backup procedure has been aborted, and device B should exit backup mode. A message related to the backup procedure is defined to be any ReadProperty, WriteProperty, CreateObject, AtomicReadFile, or AtomicWriteFile request that directly accesses a configuration File object.

16.X.3 Restore

For the purposes of this discussion, the device performing the restore procedure will be referred to as device A, and the device being restored will be device B.

16.X.3.1 Initiation of the Restore Procedure

Device A sends a ReinitializeDevice(STARTRESTORE, <password>) message to device B. Device A will await a response from device B before continuing the restore procedure.

16.X.3.2 Preparation for Restore

Upon receipt of the restore request, if device B is able to perform a restore procedure, device B will prepare for the restore procedure and will respond with a 'Result(+)' to the ReinitializeDevice service request.

If device B is unable to perform a restore procedure, then it will respond to the ReinitializeDevice service request with a 'Result(-)' response. Assuming device B supports the restore procedure and the request was properly formulated, the valid Error Class:Error Codes that can be returned are:

DEVICE:CONFIGURATION_IN_PROGRESS – if device B is already processing a backup or a restore request.

SERVICES:SERVICE_REQUEST_DENIED – if the password that was provided was incorrect or if a password is required and one was not provided.

After device B responds to the ReinitializeDevice request with a 'Result(+)', the configuration File objects must exist in the device, or device B must be able to accept CreateObject requests from device A to create the configuration File objects. It is a local matter as to whether device B will respond to other requests while it is in restore mode. The exception to this is that device B must accept and fulfill read and write requests by device A that consist of accesses to device B's Device object and/or its configuration File objects. Any services that are rejected due to an in-progress backup procedure will be rejected with an error class of DEVICE and error code of CONFIGURATION_IN_PROGRESS.

Device B must be prepared to answer device A's requests for information from device B's Device object. If device B cannot service requests from devices other than device A, then device B shall reject those services with an error class of DEVICE and an error code of CONFIGURATION_IN_PROGRESS.

It is a local matter as to whether device B will continue to perform control actions while it is in restore mode. If device B changes its operational behavior during a restore procedure, then the System_Status property of the Device object shall be set to DOWNLOAD_IN_PROGRESS.

16.X.3.3 Restoring the Configuration Files

Device A will use the AtomicWriteFile service to write each configuration file to device B. If any of the files do not exist in device B, then device A will attempt to create the files using the CreateObject service. Any files that already exist in the device, and differ in size from the image being written to them, will be truncated by writing 0 to the File_Size property of the File object before the contents are written to the file.

The configuration files will be written as a stream of bytes, or as a sequence of records, depending on the value of the File_Access_Method property of the File object. Note that there is no standard file format for record-based files, whereas any file can be written as a stream of bytes.

Each configuration file written to the device should be a valid configuration file obtained from the vendor, from a vendor's configuration tool, or from a previous backup procedure. The files will be written to the device in the same order as they were retrieved during the backup procedure, or as specified by the vendor if the files were obtained from another source.

Device B is allowed to reject any write operation to the configuration file if it has determined that the content of the write is invalid (internal CRC error, Invalid type code, etc). If this is the case, device B will respond with an error class of SERVICES and an error code of INVALID_CONFIGURATION_DATA. It is a local matter as to whether device A will retry the request and how many times device A will retry, but device A should abort the restore procedure if device B continues to return an error.

16.X.3.4 Ending the Restore Procedure

When device A has completely written all of the configuration files to device B, device A will send ReinitializeDevice(ENDRESTORE, <password>). Device B will perform whatever actions are required to complete the restore procedure, which should include a validation of the restored configuration. If the validation fails, it is a local matter as to what device B will do beyond changing its System_State property to something other than DOWNLOAD_IN_PROGRESS.

If device A needs to abort the restore for any reason (i.e., the user aborts the procedure, device B fails to allow writes to a configuration file, or device A detects any other condition that inhibits the restore procedure), device A shall attempt to send ReinitializeDevice(ABORTRESTORE, <password>) to device B. Upon receipt of this message, device B shall abort the restore procedure.

If device B does not receive any messages related to the restore procedure from device A for the number of seconds specified in the Backup_Failure_Timeout property of its Device object, device B should assume that the restore procedure has been aborted, and device B should exit restore mode. A message related to the restore procedure is defined to be any ReadProperty, WriteProperty, CreateObject, or AtomicWriteFile request that directly accesses a configuration File object.

Once the restore procedure has ended, whether it was successful or not, device B must change its System_Status property to something other than DOWNLOAD_IN_PROGRESS.

If the restore is successful, no other actions by device A shall be required, and, device B will update the Last_Restore_Time property in its Device object.

If the restore failed or was aborted and device B is unable to recover its old configuration, or cannot establish a default configuration, device B shall set its System_State to DOWNLOAD_REQUIRED. Every attempt shall be made to leave device B in a state that will accept additional restore procedures.

[Addition to **18.6**, p. 315, by inserting in alphabetical order and renumbering other subclauses]

INVALID_CONFIGURATION_DATA – The configuration data provided was invalid or corrupt.

[Change to **21**, Error production, p. 363]

```
error-code  ENUMERATED {
            …
            invalid-array-index                 (42),
            invalid-configuration-data          (46),
            invalid-data-type                   (9),
            …
            write-access-denied                 (40),
            -- see invalid-configuration-data  (46)
            …
            }

        -- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
        -- 256-65535 may be used by others subject to the procedures and constraints described
        -- in Clause 23. The last enumeration used in this version is ~~42~~ 46.
```

[Change to **Table 12-11**]

Table 12-11. Properties of the Device Object Type

Property Identifier	Property Datatype	Conformance Code
…		
Device_Address_Binding	List of BACnetAddressBinding	R
Database_Revision	*Unsigned*	*R*
Configuration_Files	*BACnetARRAY[N] of BACnetObjectIdentifier*	*O*[5]
Last_Restore_Time	*BACnetDateTime*	*O*[5]
Backup_Failure_Timeout	*Unsigned16*	*O*[6]

[5] *These properties are required if the device supports the backup and restore procedures.*
[6] *This property must be present and writable if the device supports the backup and restore procedures.*

[Add **12.9.34...12.9.37**, p. 180 – **12.9.33** is ProtocolRevision from 135*b*]

12.9.34 Database_Revision

This property, of type Unsigned, is a logical revision number for the device's database. It is incremented when an object is created, an object is deleted, an object's name is changed, or a restore is performed.

12.9.35 Configuration_Files

This optional property is a BACnet Array of BACnet Object_Identifier. Entries in the array identify the files within the device that define the device's image that can be backed up. The contents of this property is only required to be valid during the backup procedure. This property must be supported if the device supports the BACnet backup and restore procedure as described in 16.X.

12.9.36 Last_Restore_Time

This optional property, of type BACnetTimeStamp, is the time at which the device's image was last restored as per Clause16.X. This property must be supported if the device supports the BACnet backup and restore procedures as described in Clause 16.X.

12.9.37 Backup_Failure_Timeout

This optional property, of type Unsigned16, is the time, in seconds, that the device being backed up or restored must wait before unilaterally ending the backup or restore procedure. This property must be writable with the intent that the device performing the backup, or the human operator, will configure this with an appropriate timeout.

[Add to **21**, p. 366]

BACnetDeviceStatus ::= ENUMERATED {

…
non-operational (4),
backup-in-progress (5),
…
}

[Change to **21**, BACnetPropertyIdentifier]

BACnetPropertyIdentifier ::= ENUMERATED {

…
archive (13),
backup-failure-timeout (153),
bias (14),
…
-- the property in this place was deleted (18),
configuration-files, *(154),*
controlled-variable-reference (19),
…
cov-increment (22),
database-revision (155),
datelist (23),
…
issue-confirmed-notifications (51),
last-restore-time (157),
limit-enable (52),
…
}

[Change to **Annex C**, DEVICE OBJECT encoding, pp. 414-415]

device-address-binding [30] SEQUENCE OF BACnetAddressBinding,
database-revision [155] Unsigned,
configuration-files [154] SEQUENCE OF BACnetObjectIdentifier,
last-restore-time [157] BACnetTimeStamp,
backup-failure-timeout [153] Unsigned16
}

[Change to **D.9**, Example 1, p. 425]

…

Property:	Device_Address_Binding =	(((Device, Instance 1), 1, X'01'), (Device, Instance 12), 1, X'23'), (Device, Instance 40), 2, X'02608C41A606'), ...)
Property:	*Database_Revision =*	*123*
Property:	*Configuration_Files =*	*((File, Instance 1), (File, Instance 2))*
Property:	*Last_Restore_Time =*	*(2, (29-SEP-1989, 01:00:00.0))*
Property:	*Backup_Failure_Timeout =*	*300*

[Change to **D.9**, Example 2, p. 425]

Property:	Device_Address_Binding =	(((Device, Instance 1), 1, X'01'))
Property:	*Database_Revision =*	*69*

[Change to **16.4**, p. 286]

The ReinitializeDevice service is used by a client BACnet-user to instruct a remote device to reboot itself (cold start), ~~or~~ reset itself to some predefined initial state (warm start), *or to control the backup or restore procedure*. ~~This service~~*Resetting or rebooting a device* is primarily initiated by a human operator for diagnostic purposes. *Use of this service during the backup or restore procedure is usually initiated on behalf of the user by the device controlling the backup or restore. Due* to the sensitive nature of this service, a password may be required from the responding BACnet-user prior to executing the service.

[Change to **16.4.1.1.1**, p. 286]

This parameter allows the client user to specify the desired state of the device after its reinitialization. The value of the parameter may be ~~either~~ *one of* COLDSTART, ~~or~~ WARMSTART*, STARTBACKUP, ENDBACKUP, STARTRESTORE, ENDRESTORE, or ABORTRESTORE*. WARMSTART shall mean to reboot the device and start over, retaining all data and programs that would normally be retained during a brief power outage. The precise interpretation of COLDSTART shall be defined by the vendor.

The use of the backup and restore commands are defined in 16.X.

[Change to **16.4.2**, p. 287]

After verifying the validity of the request, including the password, the responding BACnet-user shall pre-empt all other outstanding requests *and* respond with a 'Result(+)' primitive. ~~, and then~~ *If the request is WARMSTART or COLDSTART the responding BACnet-user will* immediately proceed to perform any applicable shutdown procedures prior to reinitializing the device as specified by the requesting BACnet-user in the request. If the service request is for WARMSTART and the device is not ready due to its initial characterization being in progress, a 'Result(-)' response primitive shall be issued. ~~If the password is invalid or is absent when one is required, a 'Result (-)' response primitive shall be issued.~~

If the requested state is one of STARTBACKUP, ENDBACKUP, STARTRESTORE, ENDRESTORE, or ABORTRESTORE, then the device shall behave as described in 16.X.

If the password is invalid or is absent when one is required, a 'Result (-)' response primitive shall be issued.

[Change to **21**, p. 357]

```
ReinitializeDevice-Request ::= SEQUENCE {
reinitializedStateOfDevice    [0]    ENUMERATED {
                                         coldstart       (0),
                                         warmstart       (1),
                                         startbackup     (2),
                                         endbackup       (3),
                                         startrestore    (4),
                                         endrestore      (5),
                                         abortrestore    (6)
                                         },
password                      [1]    CharacterString (SIZE (1..20)) OPTIONAL
}
```

135*c*-6. Define a new service, SubscribeCOVProperty, to allow COV notifications for arbitrary properties of an object with subscriber-specified COV increments.

[Add new service to **13**]

13.X SubscribeCOVProperty

The SubscribeCOVProperty service is used by a COV-client to subscribe for the receipt of notifications of changes that may occur to the properties of a particular object. Any object may optionally support COV reporting. If a standard object provides COV reporting, then changes of value of subscribed-to properties of the object, in some cases based on programmable increments, trigger COV notifications to be sent to one or more subscriber clients. Typically, COV notifications are sent to supervisory programs in BACnet client devices or to operators or logging devices.

The subscription establishes a connection between the change of value detection and reporting mechanism within the COV-server device and a "process" within the COV-client device. Notifications of changes are issued by the COV-server device when changes occur after the subscription has been established. The ConfirmedCOVNotification and UnconfirmedCOVNotification services are used by the COV-server device to convey change notifications. The choice of confirmed or unconfirmed service is made at the time the subscription is established. Any object, proprietary or standard, may support COV reporting for any property at the implementor's option.

The SubscribeCOVProperty service differs from the SubscribeCOV service in that it allows monitoring of properties other than those listed in Table 13-1.

13.X.1 Structure

The structure of the SubscribeCOVProperty service primitives is shown in Table 13-X. The terminology and symbology used in this table are explained in 5.6.

Table 13-X. Structure of SubscribeCOVProperty Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Subscriber Process Identifier	M	M(=)		
Monitored Object Identifier	M	M(=)		
Issue Confirmed Notifications	U	U(=)		
Lifetime	U	U(=)		
Monitored Property Identifier	M	M(=)		
COV Increment	U	U(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
Error Type			M	M(=)

13.X.1.1 Argument

This parameter shall convey the parameters for the SubscribeCOVProperty confirmed service request.

13.X.1.2 Subscriber Process Identifier

This parameter, of type Unsigned32, shall convey a numeric "handle" meaningful to the subscriber. This handle shall be used to match future re-subscriptions and cancellations from the subscriber with the COV context that exists within the COV-server device and with Confirmed or UnconfirmedCOVNotifications to identify the process within the COV-client that should receive them.

13.X.1.3 Monitored Object Identifier

This parameter, of type BACnetObjectIdentifier, shall convey the identifier of the object within the receiving device that contains the property for which a subscription is desired.

13.X.1.4 Issue Confirmed Notifications

This parameter, of type BOOLEAN, shall convey whether the COV-server device shall issue ConfirmedCOVNotifications (TRUE) or UnconfirmedCOVNotifications (FALSE) when changes occur. This parameter, if present, shall indicate a subscription or re-subscription is to occur and that the lifetime shall be refreshed to its initial state. If both the 'Issue Confirmed Notifications' and 'Lifetime' parameters are absent, then this shall indicate a cancellation request. If the 'Lifetime' parameter is present, then the 'Issue Confirmed Notifications' parameter shall be present.

13.X.1.5 Lifetime

This parameter, of type Unsigned, shall convey the desired lifetime of the subscription in seconds. A value of zero shall not be allowed. A non-zero value shall indicate the number of seconds that may elapse before the subscription shall be automatically cancelled. If both the 'Issue Confirmed Notifications' and 'Lifetime' parameters are absent, then this shall indicate a cancellation request. If the 'Issue Confirmed Notifications' parameter is present, then the 'Lifetime' parameter shall be present.

13.X.1.6 Monitored Property Identifier

This parameter, of type BACnetPropertyReference, shall convey the property identifier and optional array index for which a subscription is desired. If COV reporting is supported on an array property, it is at the implementor's discretion as to whether or not COV reporting on the whole array is supported.

13.X.1.7 COV Increment

This parameter, of type REAL, shall specify the minimum change in the monitored property that will cause a COVNotification to be issued to subscriber COV-clients. This parameter is ignored if the data type of the monitored property is not REAL. If the monitored data type is REAL and this parameter is not present, then the COV increment to use is taken from the COV_Increment property if it exists, and the monitored property is Present_Value; otherwise, it will be determined by the device executing the service. The intent is to allow the subscription to use the COV increment of another already existing subscription or to allow use of the COV_Increment property in the monitored object.

13.X.1.8 Result(+)

The 'Result(+)' parameter shall indicate that the requested service has succeeded.

13.X.1.9 Result(-)

The 'Result(-)' parameter shall indicate that the service request has failed. The reason for failure shall be specified by the 'Error Type' parameter.

13.X.1.9.1 Error Type

This parameter shall consist of two component parameters: (1) the 'Error Class' and (2) the 'Error Code'. See Clause 18.

13.X.2 Service Procedure

If neither 'Lifetime' nor 'Issue Confirmed Notifications' are present, then the request shall be considered to be a cancellation. Any COV context that already exists for the same BACnet address contained in the PDU that carries the SubscribeCOVProperty request and has the same 'Subscriber Process Identifier', 'Monitored Object Identifier' and 'Monitored Property Identifier' shall be disabled and a 'Result(+)' returned. Cancellations that are issued for which no matching COV context can be found shall succeed as if a context had existed, returning 'Result(+)'. If an existing COV context is found, it shall be removed from the Active_COV_Subscriptions property in the device object.

If the 'Lifetime' parameter is not present but the 'Issue Confirmed Notifications' parameter is present, then a value of zero (indefinite lifetime) shall be assumed for the lifetime. If the 'Issue Confirmed Notifications' parameter is present but the property to be monitored does not support COV reporting, then a 'Result(-)' shall be returned. If the property to be monitored does support COV reporting, then a check shall be made to locate an existing COV context for the same BACnet address contained in the PDU that carries the SubscribeCOVProperty request and has the same 'Subscriber Process Identifier', 'Monitored Object Identifier' and 'Monitored Property Identifier'. If an existing COV context is found, then the request shall be considered a re-subscription and shall succeed as if the subscription had been newly created.

If no COV context can be found that matches the request, then a new COV context shall be established that contains the BACnet address from the PDU that carries the SubscribeCOVProperty request and the same 'Subscriber Process Identifier', 'Monitored Object Identifier' and 'Monitored Property Identifier'. The new context shall be included in the Active_COV_Subscriptions property of the device object. If no context can be created, then a 'Result(-)' shall be returned.

If a new context is created, or a re-subscription is received, then the COV context shall be initialized and given a lifetime as specified by the 'Lifetime' parameter. The subscription shall be automatically cancelled after that many seconds have elapsed unless a re-subscription is received. A 'Result(+)' shall be returned and a ConfirmedCOVNotification or UnconfirmedCOVNotification shall be issued as soon as possible after the successful completion of a subscription or re-subscription request, as specified by the 'Issue Confirmed Notifications' parameter.

[Change to **13.1**, p. 218]

Change of value (COV) reporting allows a COV-client to subscribe with a COV-server, on a permanent or temporary basis, to receive reports of some changes of value of some referenced property based on fixed criteria. ~~Certain BACnet standard objects may optionally support COV reporting~~. If ~~a standard~~ *an* object provides COV reporting, then changes of value of ~~specific~~ *any subscribed-to* properties of the object, in some cases based on programmable increments, trigger COV notifications to be sent to subscribing clients. ~~Proprietary objects~~ *Any object, proprietary or standard,* may support COV reporting at the implementor's option.

COV subscriptions are established using the SubscribeCOV service *or the SubscribeCOVProperty service*. The subscription establishes a connection between the change of value detection and reporting mechanism within the COV-server device and a "process" within the COV-client device. Notifications of changes are issued by the COV-server when changes occur after the subscription has been established. The

ConfirmedCOVNotification and UnconfirmedCOVNotification services are used by the COV-server to convey change notifications. The choice of confirmed or unconfirmed service is specified in the subscription.

When a BACnet standard object, of a type listed in Table 13-1, supports COV reporting, it shall support COV reporting for the property as listed in Table 13-1. At the implementor's discretion, COV reporting may also be supported for any other property of the object. For properties listed in Table 13-1 that have a REAL data type, the COV increment used to determine when to generate notifications will be the COV_Increment property of the object unless a COV_Increment parameter is supplied in the SubscribeCOVProperty service. For other properties that have a REAL data type, the COV increment to use when not supplied with the SubscribeCOVProperty service shall be a local matter. This is to allow multiple subscribers that do not require a specific increment to use a common increment to allow for the reduction of the processing burden on the COV-server. The criteria for COV reporting for properties other than those listed in Table 13-1 is based on the data type of the property subscribed to and is described in Table 13-X.

Table 13-X. *Criteria Used for COV Reporting of Properties Other Than Those Listed in Table 13-1.*

Data Type	*Criteria*	*Properties Reported*
REAL	*If the property changes by the increment (from the service, if provided; otherwise, as determined by the device)* *or* *Status_Flags changes at all (if the object has a Status_Flags property)*	*The subscribed-to property, Status_Flags (if the object has a Status_Flags property)*
All other data types	*If the property changes at all* *or* *Status_Flags changes at all (if the object has a Status_Flags property)*	*The subscribed-to property, Status_Flags (if the object has a Status_Flags property)*

It is the responsibility of the COV-server to maintain the list of active subscriptions for each object that supports COV notification. This list of subscriptions shall be capable of holding at least a single subscription for each object that supports COV notification, although multiple subscriptions may be supported at the implementor's option. The list of subscriptions is ~~not~~ network-visible *through the device object's Active_COV_Subscriptions property*. Subscriptions may be created with finite lifetimes, meaning that the subscription may lapse and be automatically canceled after a period of time. Optionally, the lifetime may be specified as infinite, meaning that no automatic cancellation occurs. However, the COV-server is not required to guarantee preservation of subscriptions across power failures or "restarts." Periodic re-subscription is allowed and expected and shall simply succeed as if the subscription were new, extending the lifetime of the subscription.

~~Different object types~~ *The different standard objects that support standardized COV reporting use* different standardized criteria for determining that a "change of value" has occurred, *which are summarized in Table 13-1*. Proprietary object types, *or other standard object types not listed in Table 13-1,* that support COV ~~notification~~*reporting of the Present_Value property* should follow these criteria whenever possible. ~~The standardized~~ *Any* object~~s~~ may optionally provide COV support, and the change of value algorithms they may employ are summarized in Table 13-1 *and Table 13-X*.

[Change to **Table 12-11**]

Table 12-11. Properties of the Device Object Type

Property Identifier	Property Datatype	Conformance Code
…		
Device_Address_Binding	List of BACnetAddressBinding	R
Active_COV_Subscriptions	*List of BACnetCOVSubscription*	*O*[7]

[7] *This property is required if the device supports execution of either the SubscribeCOV or SubscribeCOVProperty service.*

[Addition of **12.9.X**, p. 180]

12.9.X Active_COV_Subscriptions

The Active_COV_Subscriptions property is a List of BACnetCOVSubscription, each of which consists of a Recipient, a Monitored Property Reference, an Issue Confirmed Notifications flag, a Time Remaining value and an optional COV Increment. This property provides a network-visible indication of those COV subscriptions that are active at any given time. Whenever a COV Subscription is created with the SubscribeCOV or SubscribeCOVProperty service, a new entry is added to the Active_COV_Subscriptions list. Similarly, whenever a COV Subscription is terminated, the corresponding entry shall be removed from the Active_COV_Subscriptions list.

This property is required if the device supports execution of either SubscribeCOV or SubscribeCOVProperty service.

[Add to **21**, BACnetConfirmedServiceChoice, p. 350]
(under Alarm and Event Services)]
subscribeCOVProperty (28),

(under Services added after 1995)
-- subscribeCOVProperty (28) see Alarm and Event Services

[Add to **21**, BACnet-Confirmed-Service-Request, p. 351]
(under Alarm and Event Services)]
subscribeCOVProperty [28] SubscribeCOVProperty-Request,

(under Services added after 1995)
-- subscribeCOVProperty [28] see Alarm and Event Services

[Add to **21**, under Confirmed Alarm and Event Services, pp. 352-354]

```
SubscribeCOVProperty-Request ::= SEQUENCE {
    subscriberProcessIdentifier     [0]  Unsigned32,
    monitoredObjectIdentifier       [1]  BACnetObjectIdentifier,
    issueConfirmedNotifications     [2]  BOOLEAN OPTIONAL,
    lifetime                        [3]  Unsigned OPTIONAL,
    monitoredPropertyIdentifier     [4]  BACnetPropertyReference,
    covIncrement                    [5]  REAL OPTIONAL
    }
```

[Add to **21**, BACnetError, p.361]

(under Alignment and Event Services)]
SubscribeCOVProperty [28] Error,

(under Services added after 1995)
--SubscribeCOVProperty [28] see Alarm and Event Services

[Change to **21**, BACnetServicesSupported, p. 379]

BACnetServicesSupported ::= BIT STRING {
-- Alarm and Event Services
...
subscribeCOV (5),
subscribeCOVProperty (38),

-- File Access Services
...
who-Is (34),

-- lifeSafetyOperation (37),
-- *subscribeCOVProperty (38)*
}

[Addition to **21**, p. 366]

BACnetCOVSubscription *::= SEQUENCE {*
Recipient [0] BACnetRecipientProcess,
MonitoredPropertyReference [1] BACnetObjectPropertyReference,
IssueConfirmedNotifications [2] BOOLEAN,
TimeRemaining [3] Unsigned,
COVIncrement [4] REAL OPTIONAL -- used only with monitored
-- properties with a data type of
-- REAL
}

[Change to **21**, BACnetPropertyIdentifier]

BACnetPropertyIdentifier ::= ENUMERATED {
…
active-vt-sessions (5),
active-cov-subscriptions (152),
alarm-value (6),
…
weekly-schedule (123),
-- *see active-cov-subscriptions (152)*
...
}

[Change to **Annex C**, DEVICE OBJECT encoding, pp. 414-415]

device-address-binding [30] SEQUENCE OF BACnetAddressBinding,
active-cov-subscriptions [152] SEQUENCE OF BACnetCOVSubscription
}

[Change to **D.9**, Example 1, p. 425]

Property:	Device_Address_Binding =	(((Device, Instance 1), 1, X'01'), (Device, Instance 12), 1, X'23'), (Device, Instance 40), 2, X'02608C41A606'), ...)
Property:	*Active_COV_Subscriptions =*	*((((0, (Device, Instance 12)), 300),* *((Analog Input, 1),Present_Value),TRUE,100,1.0),* *((((0, (Device, Instance 40)), 600),* *((Analog Input, 1),Present_Value),TRUE,3,1.5))*

[Addition to **E.1**, p. 437]

E.1.X Example of the SubscribeCOVProperty Service

This example illustrates the use of the SubscribeCOVProperty service to subscribe to COV notifications on the present-value of an Analog Input object.

Service =	*SubscribeCOVProperty*
'Subscriber Process Identifier' =	*18*
'Monitored Object Identifier' =	*(Analog Input, Instance 10)*
'Issue Confirmed Notifications' =	*TRUE*
'Lifetime' =	*60*
'Monitored Property' =	*(Present_Value)*
'COV Increment' =	*1.0*

[Addition to **F.1**, p. 459]

F.1.X Encoding for Example E.1.X - SubscribeCOVProperty

X'00'	*PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)*
X'02'	*Maximum APDU Size Accepted=206 octets*
X'0F'	*Invoke ID=15*
X'1C'	*Service Choice=28 (SubscribeCOVProperty-Request)*
X'09'	*SD Context Tag 0 (Subscriber Process Identifier, L=1)*
X'12'	*18*
X'1C'	*SD Context Tag 1 (Monitored Object Identifier, L=4)*
X'0000000A'	*Analog Input, Instance Number=10*
X'29'	*SD Context Tag 2 (Issue Confirmed Notifications, L=1)*
X'01'	*1 (TRUE)*
X'39'	*SD Context Tag 3 (Lifetime, L=1)*
X'3C'	*60*
X'4E'	*PD Opening Tag 4 (Monitored Property, L=1)*
X'09'	*SD Context Tag 0 (Property Identifier, L=1)*
X'55'	*85 (PRESENT_VALUE)*
X'4F'	*PD Closing Tag 4 (Property Identifier)*
X'59'	*SD Context Tag 5 (COV Increment, L=4)*
X''3F800000'	*1.0*

Assuming the service procedure executes correctly, a simple acknowledgement is returned:

X'20'	*PDU Type=2 (BACnet-SimpleACK-PDU)*
X'0F'	*Invoke ID=15*
X'1C'	*Service ACK Choice=28 (SubscribeCOVProperty)*

135*c*-7. Add Vendor ID to proprietary MS/TP frames.

[Add **9.3.9**, p. 82]

9.3.9 Frame Types 128 through 255: Proprietary Frames

These frames are available to vendors as proprietary (non-BACnet) frames. The first two octets of the Data field shall specify the unique vendor identification code, most significant octet first, for the type of vendor-proprietary frame to be conveyed. The length of the data portion of a Proprietary frame shall be in the range of 2 to 501 octets.

135*c*-8. Add a new service, GetEventInformation, that provides enough information to acknowledge alarms.

13. X GetEventInformation Service

The GetEventInformation service is used by a client BACnet-user to obtain a summary of all "active event states". The term "active event states" refers to all event-initiating objects that

a) have an Event_State property whose value is not equal to NORMAL,
   or
b) have an Acknowledged_Transitions property, which has at least one of the bits (TO_OFFNORMAL, TO_FAULT, TO_NORMAL) set to FALSE.

This service is intended to be implemented in all devices that generate event notifications.

13.X.1 Structure

The structure of the GetEventInformation service primitives is shown in Table 13-x. The terminology and symbology used in this table are explained in 5.6.

Table 13-x. Structure of GetEventInformation Service Primitives

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Last Received Object Identifier	U	U(=)		
Result(+)				
List of Event Summaries			M	M(=)
Object Identifier			M	M(=)
Event State			M	M(=)
Acknowledged Transitions			M	M(=)
Event Time Stamps			M	M(=)
Notify Type			M	M(=)
Event Enable			M	M(=)
Event Priorities			M	M(=)
More Events			M	M(=)
Result(-)			S	S(=)
Error Type			M	M(=)

13.X.1.1 Argument

This parameter indicates the GetEventInformation confirmed service request.

13.X.1.1.1 Last Received Object Identifier

This optional parameter, of type BACnetObjectIdentifier, shall specify the last Object Identifier received in a preceding GetEventInformation-ACK, if its 'More Events' parameter was TRUE. If this parameter is omitted, the returned summary shall start with the first object meeting the "active event states" criteria. A fixed object processing order is assumed, however the particular order is a local matter. If the Last Received Object Identifier has become invalid in the responding device (i.e., the object is no longer present), the service shall resume if it is possible to determine the object that would have been the successor of the object that is no longer present. Otherwise a Result(-) shall be returned with an error class of OBJECT and an error code of UNKNOWN_OBJECT.

13.X.1.2 Result(+)

The 'Result(+)' parameter shall indicate that the requested service has succeeded. A successful result includes the following parameters.

13.X.1.2.1 List of Event Summaries

The 'List of Event Summaries' shall contain zero or more Event Summaries. Each Event Summary shall consist of seven parameters: 'Object Identifier', 'Event State', 'Acknowledged Transitions', 'Event Time Stamps', 'Notify Type', 'Event Enable' and 'Event Priorities'. If the list is of length zero, then this shall be interpreted to mean that there are no event-initiating objects that have active event states in this device.

13.X.1.2.1.1 Object Identifier

This parameter, of type BACnetObjectIdentifier, shall identify the event-initiating object that has an Event_State property whose value is not equal to NORMAL or has an Acknowledged_Transitions property that has at least one of the following bits (TO_OFFNORMAL, TO_FAULT, TO_NORMAL) set to FALSE.

13.X.1.2.1.2 Event State

This parameter, of type BACnetEventState, indicates the value of the Event_State property of the object.

13.X.1.2.1.3 Acknowledged Transitions

This parameter, of type BACnetEventTransitionBits, indicates the value of the Acked_Transitions property of the object.

13.X.1.2.1.4 EventTimeStamps

This parameter, of type BACnetARRAY[3] of BACnetTimeStamps, shall convey the time stamps of the last event notifications for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events.

13.X.1.2.1.5 Notify Type

This parameter, of type BACnetNotifyType, shall convey the value of the Notify_Type property of this object.

13.X.1.2.1.6 Event Enable

This parameter, of type BACnetEventTransitionBits, shall convey the value of the Event_Enable property of the object.

13.X.1.2.1.7 Event Priorities

This parameter, of type BACnetARRAY[3] of Unsigned, shall convey the priorities specified in the Priority property of the associated Notification Class object. In the case where an Event Enrollment Object is used without an associated Notification Class Object, the three fields of this parameter shall all contain the value of the Priority property of the Event Enrollment Object.

13.X.1.2.2 More Events

This parameter, of type BOOLEAN, shall indicate whether (TRUE) or not (FALSE) more objects exist that meet the active event state criteria of the service request, but that could not be returned in the reply.

13.X.1.3 Result(-)

The 'Result(-)' parameter shall indicate that the service request has failed. The reason for failure shall be specified by the 'Error Type' parameter.

13.X.1.3.1 Error Type

This parameter shall consist of two component parameters: (1) the 'Error Class' and (2) the 'Error Code'. See Clause 18.

13.X.2 Service Procedure

After verifying the validity of the request, the responding BACnet-user shall search for all event-initiating objects that meet the following conditions, beginning with the object following (in whatever internal ordering of objects is used by the responding device) the object specified by the Last Received Object Identifier parameter, if present:

a) have an Event_State property whose value is not equal to NORMAL, or
b) have an Acknowledged_Transitions property that has at least one of the following bits (TO_OFFNORMAL, TO_FAULT, TO_NORMAL) set to FALSE.

A positive response containing the event summaries for objects found in this search shall be constructed. If no objects are found that meet these criteria, then a list of length zero shall be returned. As many of the included objects as can be returned within the APDU shall be returned. If more objects exist that meet the criteria but cannot be returned in the APDU, the 'More Events' parameter shall be set to TRUE, otherwise it shall be set to FALSE.

[Add to **21**, p. 350, BACnetConfirmedServiceChoice (under Alarm and Event Services)]

```
getEventInformation             (29),
```

[Add to **21**, p. 351, BACnet-Confirmed-Service-Request (under Alarm and Event Services)]

```
getEventInformation             [29] GetEventInformation-Request,
```

[Add to **21**, p. 352, BACnet-Confirmed-Service-ACK (under Object Access Services)]

```
getEventInformation             [29] GetEventInformation-ACK,
```

[Add to **21**, pp. 352-354, (under Confirmed Alarm and Event Services)]

```
GetEventInformation-Request ::= SEQUENCE {
    lastReceivedObjectIdentifier    [0]  BACnetObjectIdentifier OPTIONAL
        }
```

```
GetEventInformation-ACK ::= SEQUENCE {
    listOfEventSummaries   [0]  SEQUENCE OF SEQUENCE {
                                objectIdentifier          [0]  BACnetObjectIdentifier,
                                eventState                [1]  BACnetEventState,
                                acknowledgedTransitions   [2]  BACnetEventTransitionBits,
                                eventTimeStamps           [3]  SEQUENCE SIZE (3) OF BACnetTimeStamp,
                                notifyType                [4]  BACnetNotifyType,
                                eventEnable               [5]  BACnetEventTransitionBits,
                                eventPriorities           [6]  SEQUENCE SIZE (3) OF Unsigned
                                                          },
    moreEvents             [1]     BOOLEAN
        }
```

[Add to **21**, p. 361, BACnet-Error (under Alarm and Event Services)]

```
getEventInformation             [29] Error,
```

[Add to **21**, p. 379, BACnetServicesSupported (under Alarm and Event Services)]

```
 getEventInformation            (39),

-- Services added after 1995
-- getEventInformation          (39)      see Alarm and Event Services
```

[Add to **Annex E.1**, p. 435-438, **E 1.9**]

E.1.9 Example of the GetEventInformation Service

Assumed objects:	Object_Identifier	Object_Name	Object Type
	(Analog Input, Instance 2)	Zone1_Temp	ANALOG_INPUT
	(Analog Input, Instance 3)	Zone2_Temp	ANALOG_INPUT

A BACnet device attempting to obtain a list of active events from another BACnet device would issue the following service request:

Service = GetEventInformation

A typical response to this request would be a 'Result(+)' containing no parameters, indicating that there are no active events, or a 'Result(+)', conveying a list of active events as shown below.

'List of Event Summaries' = (((Analog Input, Instance 2), HIGH_LIMIT, B'011', ((7-Jun-99, 15:35:00.20), (*-*-*,*:*:*.*), (*-*-*,*:*:*.*)), ALARM, B'111', (15, 15, 20)),
((Analog Input, Instance 3), NORMAL, B'110', ((7-Jun-99, 15:40:00.00), (*-*-*,*:*:*:*), (15:45:30.30)), ALARM, B'111', (15, 15, 20)),
FALSE)

Notice that the to NORMAL transition from Zone2_Temp has not been acknowledged.

[Add to **Annex F**, p. 460, **F.1.9**]

F.1.9 Encoding for Example E.1.9 - GetEventInformation Service

X'00' PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=1)
X'02' Maximum APDU Size Accepted = 206 octets
X'01' Invoke ID = 1
X'1D' Service Choice = 29 (GetEventInformation)

Assuming the service procedure executes correctly, a complex acknowledgement is returned:

X'30' PDU Type = 3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'01' Invoke ID=01
X'1D' Service ACK Choice = 29, (GetEventInformation-ACK)

X'0E' PD opening Tag 0
X'0C' SD context Tag 0 (ObjectIdentifier, L=4)
X'00000002' Analog Input, Instance Number = 2
X'19' SD context Tag 1 (Enumerated, L=1)
X'03' 3 (HIGH_LIMIT)
X'2A' SD context Tag 2 (Bit String, L=2)
X'0560' 0,1,1 (FALSE, TRUE, TRUE)
X'3E' PD opening Tag 3
X'0D' SD context Tag 0 (Time L=4)
X'0F230014' Time 15:35:00.20
X'0D' SD context Tag 0 (Time L=4)
X'FFFFFFFF' Time unspecified
X'0D' SD context Tag 0 (Time L=4)
X'FFFFFFFF' Time unspecified
X'3F' PD closing Tag 3
X'49' SD context Tag 4 (Enumerated, L=1)
X'00' 0 (ALARM)
X'5A' SD context Tag 5 (Bit String, L=2)
X'05E0' 1,1,1 (TRUE, TRUE, TRUE)
X'6E' PD opening Tag 6
X'21' Application Tag 2 (Unsigned Integer, L=1)
X'0E' 15 (Priority)
X'21' Application Tag 2 (Unsigned Integer, L=1)
X'0E' 15 (Priority)
X'21' Application Tag 2 (Unsigned Integer, L=1)
X'14' 20 (Priority)
X'6F' PD closing Tag 6
X'0C' SD context Tag 0 (ObjectIdentifier, L=4)
X'00000003' Analog Input, Instance Number = 3
X'19' SD context Tag 1 (Enumerated, L=1)
X'00' 0 (NORMAL)
X'2A' SD context Tag 2 (Bit String, L=2)
X'05C0' 1,1,0 (TRUE, TRUE, FALSE)
X'3E' PD opening Tag 3
X'0D' SD context Tag 0 (Time L=4)
X'0F280000' Time 15:40:00.00
X'0D' SD context Tag 0 (Time L=4)
X'FFFFFFFF' Time unspecified
X'0D' SD context Tag 0 (Time L=4)
X'0F2D1E1E' Time 15:45:30.30
X'3F' PD closing Tag 3

X'49'		SD context Tag 4 (Enumerated, L=1)
X'00'		0 (ALARM)
X'5A'		SD context Tag 5 (Bit String, L=2)
X'05E0'		1,1,1 (TRUE, TRUE, TRUE)
X'6E'		PD opening Tag 6
	X'21'	Application Tag 2 (Unsigned Integer, L=1)
	X'0E'	15 (Priority)
	X'21'	Application Tag 2 (Unsigned Integer, L=1)
	X'0E'	15 (Priority)
	X'21'	Application Tag 2 (Unsigned Integer, L=1)
	X'14'	20 (Priority)
X'6F'		PD closing Tag 6
X'0F'		PD closing Tag 0
X'19'		SD context Tag 1 (Boolean, L=1)
X'00'		FALSE (More Events)

[Change **13.8**, GetAlarmSummary Service, p. 238]

The GetAlarmSummary service is used by a client BACnet-user to obtain a summary of "active alarms." The term "active alarm" refers to BACnet standard objects that have an Event_State property whose value is not equal to NORMAL and a Notify_Type property whose value is ALARM. ~~This is a very simple service intended to be implementable in all devices that are capable of alarm processing.~~ The GetEventEnrollmentSummary service provides a more sophisticated approach with various kinds of filters.

HISTORY OF REVISIONS

Protocol		*Summary of Changes to the Standard*
Version	*Revision*	
1	**NA**	ANSI/ASHRAE 135-1995
1	**NA**	Addendum *a* to ANSI/ASHRAE 135-1995 1. Add Annex J - BACnet/IP and supporting definitions
1	**1**	Addendum *b* to ANSI/ASHRAE 135-1995 1. Inconsistencies are eliminated in the definitions of the Analog and Binary Value object types 2. Any device that receives and executes UnconfirmedEventNotification service requests must support programmable process identifiers 3. Modify each event-generating object type to contain the last timestamp for each acknowledgeable transition 4. Modify the Notification Class object by requiring that the 'Notification Class' property be equivalent *to* the instance number of the Notification Class object 5. Modify the Event Notification services to make the 'To State' parameter mandatory for notifications of type ACK_NOTIFICATION 6. A new BACnetDeviceObjectPropertyReference production is added and its use in the Event Enrollment and Schedule object types is specified 7. Add a Multi-state Value object type 8. Add an Averaging object type 9. Change all 'Process Identifier' properties and parameters to Unsigned32 10. Change the Multi-state Input object type to correct flaws related to fault detection and reporting and achieve consistency with the proposed Multi-state Value object type 11. Add a Protocol_Revision property to the Device object type 12. The File object type is changed to allow truncation and partial deletion operations 13. A new ReadRange service is added to permit reading a range of data items from a property whose datatype is a list or array of lists 14. A new UTCTimeSynchronization service is introduced and related changes are made to properties in the Device object type 15. Add a Trend Log object type 16. The UnconfirmedCOVNotification service is extended to allow notifications without prior subscription as a means of distributing globally important data to a potentially large number of recipients 17. Add eight new BACnet engineering units.
1	**2**	Addendum *c* to ANSI/ASHRAE 135-1995 1. Add a new Life Safety Point object type that represents the characteristics of initiating and indicating devices in the fire, life safety, and security applications

		2. Add a new Life Safety Zone object type that represents the characteristics associated with an arbitrary group of BACnet Life Safety Point and Life Safety Zone objects 3. Add functionality to the existing BACnet alarm and event features needed to support the Life Safety Point and Life Safety Zone object types 4. Add a new LifeSafetyOperation service that provides silence and reset capabilities needed for life safety systems 5. Add a new subclause to 16 to describe the use of existing BACnet services to provide backup and restore capability 6. Define a new service, SubscribeCOVProperty, to allow COV notifications for arbitrary properties of an object with subscriber-specified COV increments 7. Add Vendor ID to proprietary MS/TP frames 8. Add a new service, GetEventInformation, that provides enough information to acknowledge alarms
1	**2**	Addendum *d* to ANSI/ASHRAE 135-1995 1. Replace Clause 22 with a new clause entitled "Conformance and Specification". 2. Update Annex A, "Protocol Implementation Conformance Statement". 3. Add a new Annex K entitled "BACnet Interoperability Building Blocks (BIBBs)". 4. Add a new Annex L entitled "Descriptions and Profiles of Standardized BACnet Devices".
1	**2**	Addendum *e* to ANSI/ASHRAE 135-1995 1. Define the PTP conection status when the half-router can and canot re-establish the connection. 2. Add Object Profiles and Extensions. 3. Add the capability for devices to advertise the maximum number of segments of a segmented APDU that they can receive.

NA = Not Applicable because the Protocol_Revision property was first defined in Addendum *b*