ANSI/ASHRAE Addendum j to
ANSI/ASHRAE Standard 135-2008

ASHRAE STANDARD

BACnet®—A Data Communication Protocol for Building Automation and Control Networks

Approved by the ASHRAE Standards Committee on June 20, 2009; by the ASHRAE Board of Directors on June 24, 2009; and by the American National Standards Institute on June 25, 2009.

ISSN 1041-2336

Approved American National Standard
ANSI
www.ansi.org

American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.
1791 Tullie Circle NE, Atlanta, GA 30329
www.ashrae.org

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 committee members may or may not be ASHRAE members, 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, or
d. permission to reprint portions of the Standard.

DISCLAIMER

ASHRAE uses its best efforts to promulgate Standards and Guidelines 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, installed, or operated in accordance with ASHRAE's Standards or Guidelines or that any tests conducted under its Standards or Guidelines will be nonhazardous or free from risk.

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 that may serve to guide the industry. The creation of ASHRAE Standards and Guidelines is determined by the need for them, and conformance to them is completely voluntary.

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

[This foreword and the "rationales" on the following pages are not part of this standard. They are merely informative and do not contain requirements necessary for conformance to the standard.]

FOREWORD

Addendum 135*j* to ANSI/ASHRAE Standard 135-2008 contains a number of changes to the current standard. These modifications are the result of change proposals made pursuant to the ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135. The changes are summarized below.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-2008 and Addenda is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are added, plain type is used throughout.

SSPC 135 wishes to recognize the efforts of the following people in developing this addendum: Corey Balfour, Howard Coleman, Clifford Copass, Stuart Donaldson, David Fisher, Philippe Goetz, John Hartman, Ryan Hughson, Les Mather, Kornelia Mergner, Hans-Joachim Mundt, Masahiro Ogawa, David Ritter, and Rob Zivney.

135-2008*j*-1. Add a new Access Point object type.

Rationale
Support for access control in BACnet includes requires a standardized object whose properties represent the externally visible characteristics associated with the authentication and authorization process of an access controlled point (e.g., door, gate, turnstile).

Addendum 135-2008*j*-1

[Add new clauses under Clause **3.2**, pp. 2-5, distributing them alphabetically and renumbering subsequent clauses]

3.2.x authentication factor: a data element of the credential which is used to verify a credential's identity.

3.2.x authorization (physical access control): the process of determining whether the access user is permitted to enter a protected zone through an access controlled point.

3.2.x access user (physical access control): the person or asset holding one or more credentials.

3.2.x access rights (physical access control): the access privileges granted to a credential.

3.2.x credential (physical access control): the combination of authentication factors and access rights.

3.2.x physical access control (PACS): an electronic system that controls the ability of people or vehicles to enter a protected area, by means of authentication and authorization at access control points.

3.2.x role-based access control (RBAC): access privileges that are assigned to specific roles. Access users acquire privileges through their assigned role.

[Change Clause 3.3, pp. 5-7]

3.3 Abbreviations and Acronyms Used in this Standard

...	
A	**application layer (prefix)**
ABA	*American Bankers Association*
AE	**application entity**
...	
DES	**Data Encryption Standard (FIPS 46-1)**
DESFire	*Data Encryption Standard Fast, Innovative, Reliable and Secure*
DID	**ARCNET destination MAC address**
...	
LPDU	**link protocol data unit**
LRC	*Longitudinal Redundancy Check*
LSAP	**link service access point (X'82' for BACnet)**
…	
SDU	service data unit
SIA	*Security Industry Association*
SID	ARCNET source MAC address
…	

[Add new Clause **12.X**, p. 288]

12.X Access Point Object Type

The Access Point object type defines a standardized object whose properties represent the externally visible characteristics associated with the authentication and authorization process of an access controlled point. (e.g., door, gate, turnstile). Access through this point is directional in that it represents access in one direction only. A door, in which access is controlled in both directions, is represented by two separate Access Point objects.

Authentication is the process of verifying the identity of an access user requesting access through an access-controlled door. This can be an authentication policy as simple as a single-factor authentication, in which one authentication factor (i.e., magnetic-stripe card, proximity-card, smart card) is used to identify a known user. In a multi-factor authentication policy, a combination of two or more authentication factors (e.g., card + PIN, card + biometric) are used to verify the identity of the access user. The Access Point object supports the definition of single-factor and multi-factor authentication policies, including the functionality to switch the policy in effect. On false attempts to authenticate, the Access Point may lock-down, for an infinite or specific amount of time.

Authorization is the process of determining whether the access user is permitted to access the zone that he or she has requested to enter. Once the access user has been authenticated successfully, a list of criteria is checked to determine whether access can be granted. If one or more of the authorization criteria fail, then the access user is denied access. Once the access user is granted access, the door will be commanded unlocked at the specified command priority and the access user can access the zone. The door which is controlled is specified in the Access Point. Authorization criteria supported by the Access Point are authorization modes, occupancy enforcement, external verification and threat level.

Authentication and authorization begins when an access user presents an authentication factor at the access controlled point. The process this object represents consumes the authentication factors from the corresponding Credential Data Input objects and performs the authentication and authorization functions. The result is to grant or deny access and to generate corresponding access events. If the object is out of service or not reliable, then these functions are not performed.

The Access Point object generates access events. Access events are stateless (i.e., NORMAL to NORMAL transitions only). A single access transaction, such as a request to enter or an operator action, can result in one or more access events. All access events that belong to the same access transaction have the same access event tag.

For intrinsic reporting, the ACCESS_EVENT algorithm is applied for both Access Alarm Events and Access Transaction Events:

Access Alarm Events: These are events requiring operator attention and handling, and the Access Point object may request human operator acknowledgement of these events.

Access Transaction Events: These are events that are to be logged, not requiring immediate operator attention. The Access Point object does not request human operator acknowledgement of these events.

Access Points which authorize entrance to an access controlled zone are entry points of that zone. Access Points which authorize exit from an access controlled zone are exit points of that zone. In the typical case a specific Access Point is an exit point from one zone and an entry point to an adjacent zone. If the Access Point leads from an Access Zone to no zone (i.e., outside), then the Access Point is an exit point only. If the Access Point leads from no zone (i.e., outside) to an Access Zone, then the Access Point is an entry point only. If the Access Point does not lead from or to an Access Zone (e.g., internal check point or muster point), then the Access Point is neither an entry nor an exit point.

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

Table 12-X. Properties of the Access Point Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Reliability	BACnetReliability	R
Out_Of_Service	BOOLEAN	R
Authentication_Status	BACnetAuthenticationStatus	R
Active_Authentication_Policy	Unsigned	R
Number_Of_Authentication_Policies	Unsigned	R
Authentication_Policy_List	BACnetARRAY[N]of BACnetAuthenticationPolicy	O[1]
Authentication_Policy_Names	BACnetARRAY[N] of CharacterString	O[1]
Authorization_Mode	BACnetAuthorizationMode	R
Verification_Time	Unsigned	O
Lockout	BOOLEAN	O[2]
Lockout_Relinquish_Time	Unsigned	O
Failed_Attempts	Unsigned	O
Failed_Attempt_Events	List of BACnetAccessEvent	O
Max_Failed_Attempts	Unsigned	O[3]
Failed_Attempts_Time	Unsigned	O[3]
Threat_Level	BACnetAccessThreatLevel	O
Occupancy_Upper_Limit_Enforced	BOOLEAN	O
Occupancy_Lower_Limit_Enforced	BOOLEAN	O
Occupancy_Count_Adjust	BOOLEAN	O
Accompaniment_Time	Unsigned	O
Access_Event	BACnetAccessEvent	R
Access_Event_Tag	Unsigned	R
Access_Event_Time	BACnetTimeStamp	R
Access_Event_Credential	BACnetDeviceObjectReference	R
Access_Event_Authentication_Factor	BACnetAuthenticationFactor	O
Access_Doors	BACnetARRAY[N] of BACnetDeviceObjectReference	R
Priority_For_Writing	Unsigned (1…16)	R
Muster_Point	BOOLEAN	O
Zone_To	BACnetDeviceObjectReference	O
Zone_From	BACnetDeviceObjectReference	O
Notification_Class	Unsigned	O[4]
Transaction_Notification_Class	Unsigned	O
Access_Alarm_Events	List of BACnetAccessEvent	O[4]
Access_Transaction_Events	List of BACnetAccessEvent	O[4]
Event_Enable	BACnetEventTransitionBits	O[4]
Acked_Transitions	BACnetEventTransitionBits	O[4]
Notify_Type	BACnetNotifyType	O[4]
Event_Time_Stamps	BACnetARRAY[3] of BACnetTimeStamp	O[4]
Profile_Name	CharacterString	O

[1] The size of this array shall equal the value of the Number_Of_Authentication_Policies property.
[2] This property is required to be present if Lockout_Relinquish_Time is present.

[3] If this property is present, then the Failed_Attempts property shall be present.
[4] These properties are required to be present if the object supports intrinsic reporting.

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 ACCESS_POINT.

12.X.4 Description

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

12.X.5 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of the Access Point. A more detailed status may 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 has a value of NORMAL, otherwise logical TRUE (1).
FAULT	Logical TRUE (1) if the Reliability is not NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	Logical TRUE (1) if the Access Point has been overridden by some mechanism local to the BACnet Device. Otherwise, the value is logical FALSE (0).
OUT_OF_SERVICE	Logical TRUE (1) if the Out_Of_Service property has a value of TRUE, otherwise logical FALSE (0).

12.X.6 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. If the object supports intrinsic reporting, then the Event_State property shall indicate the event state of the object. The allowed event states are NORMAL and FAULT. If the object does not support intrinsic reporting, then:

(a) if the Reliability is NO_FAULT_DETECTED, then Event_State shall be NORMAL, or
(b) if the Reliability is not NO_FAULT_DETECTED, then Event_State shall be FAULT.

12.X.7 Reliability

The Reliability property, of type BACnetReliability, provides an indication of whether the authentication and authorization process this object represents is "reliable" as far as the BACnet Device can determine and, if not, why. The

Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, PROCESS_ERROR, CONFIGURATION_ERROR, COMMUNICATION_FAILURE, UNRELIABLE_OTHER}.

If Reliability has a value other than NO_FAULT_DETECTED, the process that this object represents shall not perform any authentication or authorization. No access events are generated in this case.

12.X.7.1 Conditions for Generating a TO-FAULT Event

A TO-FAULT event is generated under these conditions:

(a) the Reliability property becomes not equal to NO_FAULT_DETECTED, and
(b) the TO-FAULT flag is set in the Event_Enable property.

12.X.7.2 Conditions for Generating a TO-NORMAL Event

A TO-NORMAL event is generated under these conditions:

(a) the Reliability property becomes equal to NO_FAULT_DETECTED, and
(b) the TO-NORMAL flag is set in the Event_Enable property.

12.X.8 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the authentication and authorization process this object represents is out of service. If out of service, then the process that this object represents shall not perform any authentication or authorization.

When this property changes from FALSE to TRUE, then the Access_Event property shall be set to OUT_OF_SERVICE. When this property changes from TRUE to FALSE, then the Access_Event property shall be set to OUT_OF_SERVICE_RELINQUISHED.

12.X.9 Authentication_Status

This property, of type BACnetAuthenticationStatus, shall indicate the current status of the authentication process. This is an enumeration with the following status values:

NOT_READY	The authentication process is not ready to perform a new authentication. This indicates a temporary condition due to processing of the current authentication factor, initialization during startup or other internal processing.
READY	The authentication process is ready to start a new authentication
DISABLED	The authentication process has been disabled. The property shall take on this status when the Out_Of_Service property is TRUE.
WAITING_FOR_AUTHENTICATION_FACTOR	The authentication process is waiting for an additional authentication factor for a multi-factor authentication.
WAITING_FOR_ACCOMPANIMENT	The authentication process is waiting for the authentication of the accompanying credential.

WAITING_FOR_VERIFICATION	The authentication process is waiting for the verification of the credential by an external process.
IN_PROGRESS	The authentication process is currently performing an authentication.

12.X.10 Active_Authentication_Policy

This property, of type Unsigned, shall specify the active authentication policy. The active authentication policy of this object shall be one of 'n' authentication policies, where 'n' is the number of authentication policies defined in the Number_Of_Authentication_Policies property.

The value of the Number_Of_Authentication_Policies property specifies the maximum value that can be written to Active_Authentication_Policy. If a value is written greater than the maximum value or specifies an index of an invalid entry in the Authentication_Policy_List property, if present, then the write attempt is denied and a Result(-) specifying an 'Error Class' of PROPERTY and an 'Error Code' of VALUE_OUT_OF_RANGE shall be returned.

If the Authentication_Policy_List property is present, then the value of Active_Authentication_Policy property is an array index that corresponds to the authentication policy as specified in the Authentication_Policy_List property. If no valid authentication policy exists or the current active authentication policy is not valid (i.e., the Authentication_Policy_List entry is empty or not well formed), then this property shall take a value of zero. If the value of the Number_Of_Authentication_Policies property becomes less than the value of this property, then this property shall take a value of zero.

If this property has the value of zero, then the Reliability property shall have the value CONFIGURATION_ERROR.

12.X.11 Number_Of_Authentication_Policies

This property, of type Unsigned, shall specify the number of specified authentication policies. This property shall always have a value greater than zero. If the value of this property is changed, the size of the Authentication_Policy_List array and the size of the Authentication_Policy_Names array, if present, shall also be changed to the same value.

12.X.12 Authentication_Policy_List

This optional property, of type BACnetARRAY[N] of BACnetAuthenticationPolicy, specifies the authentication policies defined for this Access Point.

Each element in the array defines an authentication policy for this access point. The BACnetAuthenticationPolicy structure contains the following fields:

Policy — This field is a list of sequence elements that specifies the Credential Data Input objects which are used for this authentication policy. Each element of the list has the following fields:

Credential-Data-Input	This field, of type BACnetDeviceObjectReference, contains a reference to an object of type Credential Data Input where the authentication factor value is read from the physical device.
Index	This field, of type Unsigned, indicates the order in which the authentication process expects to receive authentication factors from Credential Data Input objects. The value shall start with the value 1 and continue in increasing sequence. If two or more entries of the Policy list have the same index value this indicates that there is a choice between any of the authentication factors supported by the Credential Data Input objects referenced by these entries. In this case the user may present any one of these authentication factors.

Order-Enforced	If TRUE, then the ordering sequence, as specified by the Index fields of this Policy list, is enforced. If FALSE, then the order is not enforced.
Timeout	This field, of type Unsigned, specifies the maximum time in seconds for which all authentication factors, as defined by this policy, must be presented. A value of zero indicates an unlimited time to present all authentication factors. If not all authentication factors are presented in the allotted time, then a timeout occurs and authentication fails.

An Authentication_Policy_List array element shall be considered invalid if the Policy field is empty or if it is not well formed. In this case, the Reliability property shall have the value CONFIGURATION_ERROR. If a write to the Authentication_Policy_List property would cause an array element to be invalid, then a Result (-) shall be returned with an error class of PROPERTY and an error code of VALUE_OUT_OF_RANGE.

If this property is not present, then the authentication policies are a local matter.

The size of this array shall equal the value of the Number_Of_Authentication_Policies property.

12.X.12.1 Reading Authentication Factors

The authentication factors to be read are determined by the current authentication policy in effect. If the Authentication_Policy_List property is present, the authentication policy is explicitly defined and specifies which Credential Data Input objects the authentication factors are read from. When any authentication factor is read, then the Access_Event property shall be set to AUTHENTICATION_FACTOR_READ.

If the authentication factor read does not match any known authentication factor or the authentication factor read has an error (i.e., has a format type of ERROR), then authentication shall fail and access shall not be granted. In the case where the authentication factor is unknown the Access Event property shall be set to DENIED_UNKNOWN_CREDENTIAL. In the case where the authentication factor has an error, then the Access_Event property shall be set to DENIED_AUTHENTICATION_FACTOR_ERROR.

It may be possible with certain credential readers to signal a duress code when reading an authentication factor. Determining when a duress code has been read is a local matter. In this case the Access_Event property shall be set to DURESS.

12.X.12.1.1 Single-Factor Authentication

In single-factor authentication, only one authentication factor is required to identify and authenticate the access credential. Depending on the current authentication policy, the access user may have a choice of multiple credentials to use.

12.X.12.1.2 Multi-Factor Authentication

In multi-factor authentication, two or more authentication factors are used for authentication. Typically when multiple factors are used, the first authentication factor is used to identify the credential while the subsequent authentication factors are used to validate the identity. All authentication factors of a multi-factor authentication are expected to be configured in the same Access Credential object.

If a timeout is specified for the current authentication policy and not all authentication factors are read within that time, then authentication shall fail and access shall not be granted. In this case the Access_Event property is set to DENIED_AUTHENTICATION_FACTOR_TIMEOUT.

If one of the authentication factors presented is not the value expected or is presented in an incorrect order, then it is a local matter as to whether authentication fails immediately or whether the access user is given subsequent chances to present the correct authentication factor. If authentication fails immediately, then the Access_Event property shall be set to DENIED_INCORRECT_AUTHENTICATION_FACTOR. If the access user is allowed subsequent chances but fails to present the correct authentication factor within a certain number of attempts, then the Access_Event property shall be set to DENIED_MAX_ATTEMPTS. The maximum number of attempts the access user is allowed is a local matter.

12.X.12.1.3 External Authentication

If the authentication decision is made by an external process, such as a remote server, it may be possible that the authentication process becomes unavailable. When this occurs and when there is no secondary authentication process available, then the authentication shall fail and the Access_Event property shall be set to DENIED_AUTHENTICATION_UNAVAILABLE.

12.X.12.2 Initializing New Array Elements When the Array Size is Increased

If the size of the Authentication_Policy_List array is increased without initial values being provided, then the new array elements for which no initial value is provided shall be initialized with an empty Policy list, Order-Enforced shall be FALSE, and Timeout shall have the value zero.

12.X.13 Authentication_Policy_Names

This optional property, of type BACnetARRAY [N] of CharacterString, specifies the names of the defined authentication policies.

The size of this array shall equal the value of the Number_Of_Authentication_Policies property

12.X.14 Authorization_Mode

This property, of type BACnetAuthorizationMode, determines how authorization is performed at the Access Point. An Access Point object is not required to support all of these authorization modes but is required to support at least AUTHORIZE.

AUTHORIZE	The access rights of an active credential are evaluated, in addition to other possible authorization checks.
GRANT_ACTIVE	An active credential is granted access without evaluating the access rights assigned to the credential. Other authorization checks can still lead to denying access.
DENY_ALL	All credentials are denied access and the Access_Event property is set to DENIED_DENY_ALL. When a credential has a master exemption it is granted access unless other authorization checks fail.
VERIFICATION_REQUIRED	The access rights of an active credential are evaluated, in addition to other possible authorization checks. Granting access requires external verification. In this case the Access_Event property is set to VERIFICATION_REQUIRED and the access point waits for the external verification. The external verification process and the mechanism by which the verification result is provided to the access point is a local matter. If the external verification process denies access, then the Access_Event property shall be set to DENIED_VERIFICATION_FAILED. If there is no external verification result within the time specified by the Verification_Time property, then the Access_Event property shall be set to DENIED_VERIFICATION_TIMEOUT.

AUTHORIZATION_DELAYED	The access rights of an active credential are evaluated, in addition to other possible authorization checks. Granting access is delayed by the time specified by the Verification_Time property. This provides an external verification process the opportunity to deny access. In this case the Access_Event property is set to AUTHORIZATION_DELAYED and the access point waits for the external verification. The external verification process and the mechanism by which the verification result is provided to the access point is a local matter. If the external verification process denies access within the time specified in the Verification_Time property, then the Access_Event property shall be set to DENIED_VERIFICATION_FAILED. If there is no external verification result within the time specified by the Verification_Time property, then this authorization check succeeded.
NONE	No authorization functionality takes place at this access point and no authorization events (e.g., grant or any deny events) are generated. This may be used to implement special access point functionality, such as a guard tour or muster point, where authorization checks are not required.
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values to allow proprietary authorization modes other than those defined by the standard. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

12.X.14.1 Authorization Decision

Authorization is the process of determining whether or not the credential that has been used to request access at an access point will be permitted access. The authorization process completes when the access decision to grant or deny has been reached. Typically there are multiple authorization criteria used to determine if access will be granted. Examples of authorization criteria are checking access rights, checking passback violations, checking threat level, checking occupancy limits, etc. It is a local matter as to what order the authorization criteria are checked. The authorization criteria used to determine whether access is allowed may change according to the time of day, the location and the credential used.

If all authorization criteria are successful, then the credential is granted access at the access point. In this case, the Access_Event property shall be set to GRANTED.

If even one authorization check fails, then access is denied and the Access_Event property is set to the appropriate deny event. If there is no access event, either defined or proprietary, which is specific to the actual reason why access was denied, then the Access_Event property shall be set to DENIED_OTHER.

Access may be denied if the authorization process detects an inconsistency with the access request, such as when an access user requests access to a zone but there is no record of that access user being in the building. How the inconsistency is determined is a local matter. In this case access is denied and the Access_Event property shall be set to DENIED_UNEXPECTED_LOCATION_USAGE.

12.X.15 Verification_Time

This optional property, of type Unsigned, shall specify the time, in seconds, to wait for external verification when the Authorization_Mode property has a value of AUTHORIZATION_DELAYED or VERIFICATION_REQUIRED.

12.X.16 Lockout

This optional property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the access controlled point this object represents is in a lockout state. When the access point is in a lockout state, any access request shall always be denied, except for an active credential that has a master exemption. For each denied access request, the Access_Event property shall be set to DENIED_LOCKOUT. An Access Point object may be set to a lockout state due to too many failed access attempts, as defined in the Max_Failed_Attempts property, or by writing TRUE to this property.

When the property Lockout becomes TRUE due to too many failed access attempts, then the Access_Event property shall be set to LOCKOUT_MAX_ATTEMPTS. If TRUE is written to this property for any other reason, the Access_Event property shall be set to LOCKOUT_OTHER. When the Lockout property becomes FALSE, the Access_Event property shall be set to LOCKOUT_RELINQUISHED.

If the Lockout property is present, then the Lockout_Relinquish_Time property shall also be present.

12.X.17 Lockout_Relinquish_Time

This optional property, of type Unsigned, shall specify the time, in seconds, to delay after the Lockout property has taken on the value TRUE, before automatically relinquishing the lockout state. The lockout state is relinquished by setting the Lockout property to FALSE. A value of zero indicates that the lockout state will not automatically be relinquished.

If the Lockout_Relinquish_Time is present, then the Lockout property shall also be present.

12.X.18 Failed_Attempts

This optional property, of type Unsigned, shall indicate the current count of successive failed access attempts. Any successive failed access attempt shall increment the value of this property.

This property shall be set to zero when a successful access attempt occurs or when the property Lockout becomes FALSE.

12.X.19 Failed_Attempt_Events

This optional property, of type List of BACnetAccessEvent, specifies those access events that are counted as a failed access attempt.

If this property is not present, it is a local matter as to which access events are considered a failed attempt.

12.X.20 Max_Failed_Attempts

This optional property, of type Unsigned, shall specify the maximum number of successive failed access attempts before the Lockout property is set to TRUE. If the Failed_Attempts property becomes greater than or equal to the value of this property and this property is not zero, the Lockout property is set to TRUE. Zero indicates that the Lockout property is not set to TRUE as the result of failed access attempts.

If the Max_Failed_Attempts property is present, then the Failed_Attempts property shall also be present.

12.X.21 Failed_Attempts_Time

This optional property, of type Unsigned, shall specify the time, in seconds, to delay before setting the Failed_Attempts property to zero, after the last failed access attempt.

If the Failed_Attempts_Time is present, then the Failed_Attempts property shall also be present.

12.X.22 Threat_Level

This optional property, of type BACnetAccessThreatLevel, shall specify the current threat level for this Access Point. Zero is the lowest threat level, effectively disabling the threat level check, while 100 is the maximum threat level. If the threat authority of the authenticated credential is lower than the value of this property, then the authorization fails. In this case the Access_Event property shall be set to DENIED_THREAT_LEVEL.

12.X.23 Occupancy_Upper_Limit_Enforced

This optional property, of type BOOLEAN, indicates whether the upper occupancy limit of the access controlled zone, for which this object is an entry point, is enforced (TRUE) or not (FALSE). If enforced, authorization shall fail if the access controlled zone's occupancy is greater than or equal to its upper occupancy limit, unless the credential is exempted from this authorization check. When this authorization check fails, the Access_Event property shall be set to DENIED_UPPER_OCCUPANCY_LIMIT.

12.X.24 Occupancy_Lower_Limit_Enforced

This optional property, of type BOOLEAN, indicates whether the lower occupancy limit of the access controlled zone, for which this object is an exit point, is enforced (TRUE) or not (FALSE). If enforced, authorization shall fail if the access controlled zone's occupancy is lower than or equal to its lower occupancy limit, unless the credential is exempted from this authorization check. When this authorization check fails, the Access_Event property shall be set to DENIED_LOWER_OCCUPANCY_LIMIT.

12.X.25 Occupancy_Count_Adjust

This optional property, of type BOOLEAN, indicates whether (TRUE) this object will adjust the occupancy count of the zones for which it controls access, or not (FALSE). The occupancy count is decremented for the zone for which this Access Point is an exit point and incremented for the zone for which this Access Point is an entry point.

Occupancy count shall be adjusted if the credential holder passes through the access point. How this is determined is a local matter. The occupancy count of the zones is adjusted by writing a negative amount to the Adjust_Value property of the exit access zone and the corresponding positive amount to the Adjust_Value property of the entry access zone.

If this property is not supported, then the Access Point object behaves as if the value is FALSE.

12.X.26 Accompaniment_Time

This optional property, of type Unsigned, shall specify the time, in seconds, to wait for a second credential to be presented at this access point when the original credential requires accompaniment. If an accompanying credential is not presented within this time the authorization of the original credential shall fail and the Access_Event property shall be set to DENIED_NO_ACCOMPANIMENT.

12.X.27 Access_Event

This property, of type BACnetAccessEvent, indicates the last access event which occurred at this Access Point.

BACnetAccessEvent is an enumeration of authentication and authorization decisions, subsequent actions, and results of these actions. An Access Point is not required to support all values of this enumeration.

NONE	The access point did not yet determine any access event or the object is not generating access events. This is not a reported event. It is required to enable algorithmic ACCESS_EVENT reporting.
GRANTED	Access granted to the presented credential.
MUSTER	If the access point is a muster point, a muster event is generated when a credential is presented.
PASSBACK_DETECTED	A passback violation for the presented credential has been detected.
DURESS	A duress incident was detected at this access point.
TRACE	A traced credential has been presented.

LOCKOUT_MAX_ATTEMPTS	The access point is in a lockout state due to maximum failed access attempts.
LOCKOUT_OTHER	The access point is in a lockout state due to any reason other than maximum failed access attempts.
LOCKOUT_RELINQUISHED	The access point has relinquished the lockout state.
LOCKED_BY_HIGHER_PRIORITY	The controlled Access Door object is commanded at a higher priority.
OUT_OF_SERVICE	The Out_Of_Service flag of the Access Point object has been set to TRUE.
OUT_OF_SERVICE_RELINQUISHED	The Out_Of_Service flag of the Access Point object has been set to FALSE.
ACCOMPANIMENT_BY	The credential presented accompanies the previous credential.
AUTHENTICATION_FACTOR_READ	An authentication factor has been read. This event indicates a successful read of an authentication factor in single-factor or multi-factor authentication.
AUTHORIZATION_DELAYED	Authorization of a credential is delayed to allow time for an external process to deny access.
VERIFICATION_REQUIRED	Authorization of a credential requires verification from an external process.
NO_ENTRY_AFTER_GRANTED	Access was granted to the presented credential but the physical door was not opened.
DENIED_DENY_ALL	Access denied because the Authorization_Mode property of the Access Point object is set to DENY_ALL.
DENIED_UNKNOWN_CREDENTIAL	Access denied due to an unknown credential. The authentication factor presented did not match any known authentication factor.
DENIED_AUTHENTICATION_UNAVAILABLE	Access denied because the authentication and authorization decision is unavailable.
DENIED_AUTHENTICATION_FACTOR_TIMEOUT	Access denied due to required authentication factor for multi-factor authentication not being presented within time.
DENIED_INCORRECT_AUTHENTICATION_FACTOR	Access denied due to the authentication factor presented for a multi-factor-authentication not being the one expected.
DENIED_POINT_NO_ACCESS_RIGHTS	Access denied due to evaluation of TRUE of a negative access rule for the access point.
DENIED_ZONE_NO_ACCESS_RIGHTS	Access denied due to evaluation of TRUE of a negative access rule for the access zone.
DENIED_NO_ACCESS_RIGHTS	Access denied due to no positive access rule being found for the access zone or access point.
DENIED_OUT_OF_TIME_RANGE	Access denied due to the presented credential not being valid at this access point or access zone at this time.

DENIED_THREAT_LEVEL	Access denied due to insufficient threat authority for the presented credential.
DENIED_PASSBACK	Access denied due to a passback violation for the presented credential.
DENIED_UNEXPECTED_LOCATION_USAGE	Access denied due to the credential being used at a location which violates local consistency rules.
DENIED_MAX_ATTEMPTS	Access denied due to too many failed multi-factor access attempts at the access point.
DENIED_LOWER_OCCUPANCY_LIMIT	Exit from a zone for which this access point is an exit access point is denied due to zone occupancy being below or at the minimum limit.
DENIED_UPPER_OCCUPANCY_LIMIT	Access to a zone for which this access point is an entry access point is denied due to zone occupancy being at or above the maximum limit.
DENIED_AUTHENTICATION_FACTOR_LOST	Access denied due to the authentication factor used being reported as lost.
DENIED_AUTHENTICATION_FACTOR_STOLEN	Access denied due to the authentication factor used being reported as stolen.
DENIED_AUTHENTICATION_FACTOR_DAMAGED	Access denied due to the authentication factor used being reported as damaged.
DENIED_AUTHENTICATION_FACTOR_DESTROYED	Access denied due to the authentication factor used being reported as destroyed.
DENIED_AUTHENTICATION_FACTOR_DISABLED	Access denied due to the authentication factor used being disabled for unspecified or unknown reasons.
DENIED_AUTHENTICATION_FACTOR_ERROR	Access denied due to the authentication factor used having a read error.
DENIED_CREDENTIAL_UNASSIGNED	Access denied due to the credential used not having yet been assigned to an access user.
DENIED_CREDENTIAL_NOT_PROVISONED	Access denied due to the credential used not yet having been provisioned.
DENIED_CREDENTIAL_NOT_YET_ACTIVE	Access denied due to the credential used not yet being active.
DENIED_CREDENTIAL_EXPIRED	Access denied due to the credential used having expired.
DENIED_CREDENTIAL_MANUAL_DISABLE	Access denied due to the credential used having been manually disabled.
DENIED_CREDENTIAL_LOCKOUT	Access denied due to the credential used being locked out.
DENIED_CREDENTIAL_MAX_DAYS	Access denied due to the number of days the credential may be used having been exceeded.
DENIED_CREDENTIAL_MAX_USES	Access denied due to the number of allowed uses of the credential used has been exceeded.
DENIED_CREDENTIAL_INACTIVITY	Access denied due to the credential used being disabled after a period of inactivity.
DENIED_CREDENTIAL_DISABLED	Access denied due to the credential used being disabled

	for unspecified or unknown reasons.
DENIED_NO_ACCOMPANIMENT	Access denied due to the expected accompanying credential not being presented.
DENIED_INCORRECT_ACCOMPANIMENT	Access denied due to the accompanying credential presented being incorrect
DENIED_LOCKOUT	Access denied due to the access point being in lockout state.
DENIED_VERIFICATION_FAILED	Access denied due to an external process denying access when verification was required.
DENIED_VERIFICATION_TIMEOUT	Access denied due to an external process having failed to send a response, in the allotted time, when verification was required.
DENIED_OTHER	Access is denied for unspecified reasons.
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values to indicate Access Events other than those defined by this standard. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

12.X.27.1 Operations for setting the Access_Event property

When a new event occurs at the access point, the following series of operations shall be performed atomically:

(1) The value written to Access_Event shall be stored in the Access_Event property,
(2) If this event is the start of a new access transaction, the value of the Access_Event_Tag property shall be incremented.
(3) The current date and time shall be stored in the Access_Event_Time property.
(4) The reference to the Access Credential object that is associated with this event shall be stored in the Access_Event_Credential property. See Clause 12.X.30 for other conditions.
(5) The value of the authentication factor that is associated with this event shall be stored in the Access_Event_Authentication_Factor property. See Clause 12.X.31 for other conditions.

12.X.28 Access_Event_Tag

This property, of type Unsigned, is a numeric value which identifies the access transaction to which the current access event belongs. Multiple access events may be generated by a single access transaction.

The value of this property shall increase monotonically for each new access transaction. It may be implemented using modulo arithmetic. The initial value of this property before any access transaction has occurred shall be a local matter.

12.X.29 Access_Event_Time

This property, of type BACnetTimeStamp, indicates the most recent update time of the Access_Event property. This property shall update its value on each update of Access_Event. Update times of type Time or Date shall have X'FF' in each octet, and Sequence number update times shall have the value 0 if no update has yet occurred.

12.X.30 Access_Event_Credential

This property, of type BACnetDeviceObjectReference, shall specify the Access Credential object that corresponds to the access event specified in the Access_Event property, if applicable.
This property shall contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present, under the following conditions:

(a) there is no credential recognized up to now, or
(b) there is no credential that is associated to the current access event, or
(c) the credential of the authentication factor that is associated to the current event is unknown, or
(d) the device chooses not to expose the credential.

12.X.31 Access_Event_Authentication_Factor

This optional property, of type BACnetAuthenticationFactor, shall specify the authentication factor that corresponds to the access event specified in the Access_Event property, if applicable. Otherwise it shall contain a value of format type UNDEFINED.

This property shall contain a value of format type UNDEFINED under the following conditions:

(a) there was no authentication factor read up to now, or
(b) there is no authentication factor that is associated to the current access event, or
(c) the device chooses not to expose the authentication factor.

12.X.32 Access_Doors

This property, of type BACnetARRAY[N] of BACnetDeviceObjectReference, shall specify the references to those Access Door objects whose Present_Value properties are commanded after successful authorization. If this Access Point object does not command Access Door objects (e.g., muster point), or is used to control access to other resources or functions, or commands other objects, then this array shall be empty.

12.X.32.1 Commanding Access Doors

After successful authorization the following actions occur:

(1) The Access_Event property shall be set to GRANTED, and

(2) The physical doors, as specified in the Access_Doors property, are commanded with either a PULSE_UNLOCK or EXTENDED_PULSE_UNLOCK door command, at the priority specified by the Priority_For_Writing property.

Commanding the doors may fail due to a higher priority command in effect in the Access Door object. In this case the Access _Event property shall be set to LOCKED_BY_HIGHER_PRIORITY.

The respective Access Doors may be monitored to verify that they are opened and access takes place. If no access takes place, the Access_Event property shall be set to NO_ENTRY_AFTER_GRANTED.

12.X.33 Priority_For_Writing

This property, of type Unsigned (1..16), defines the priority at which the referenced Access Door object's Present_Value properties are commanded. It corresponds to the 'Priority' parameter of the WriteProperty service. The value 1 is considered the highest priority and 16 the lowest. See Clause 19.2.

12.X.34 Muster_Point

This optional property, of type BOOLEAN, indicates whether this Access Point generates muster access events (TRUE) or not (FALSE).

A muster event is generated by setting the Access_Event property to MUSTER after an access credential has been presented at the access point. It is a local matter as to whether a muster event is generated for unknown credentials.

12.X.35 Zone_To

This optional property, of type BACnetDeviceObjectReference, shall specify the Access Zone object for which this Access Point object is an entry access controlled point, allowing entrance to the zone. This property shall not reference the same Access Zone object as the Zone_From property. If the Access Point is not an entry point to an access controlled zone, then this property shall contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present.

12.X.36 Zone_From

This optional property, of type BACnetDeviceObjectReference, shall specify the Access Zone object for which this Access Point object is an exit access controlled point, allowing exit from the zone. This property shall not reference the same Access Zone object as the Zone_To property. If the Access Point is not an exit point from an access controlled zone, then this property shall contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present.

12.X.37 Notification_Class

This optional property, of type Unsigned, shall specify the notification class to be used when generating Access Alarm 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.38 Transaction_Notification_Class

This optional property, of type Unsigned, shall specify the notification class of Access Transaction Events. The Transaction_Notification_Class property implicitly refers to a Notification Class object that has a Notification_Class property with the same value. If this property is not present, then the Notification Class specified by the property Notification_Class shall be used for Access Transaction Events.

12.X.39 Access_Alarm_Events

This optional property, of type List of BACnetAccessEvent, shall specify any access events which are reported as Access Alarm Events. This property is required if intrinsic reporting is supported by this object.

An Access Alarm Event is reported when the following conditions are true:

(a) The Access_Event is updated and the updated value is equal to one of the values in Access_Alarm_Events, and
(b) the TO-NORMAL flag is enabled in the Event_Enable property.

The notification is sent with Notify Type as specified by the property Notify_Type.

The Notification Class object referenced by the Notification_Class property is used to report Access Alarm Events.

12.X.40 Access_Transaction_Events

This optional property, of type List of BACnetAccessEvent, shall specify any access events that are reported as Access Transaction Events. This property is required if intrinsic reporting is supported by this object.

An Access Transaction Event is reported when the following conditions are true:

(a) The Access_Event is updated and the updated value is equal to one of the values in Access_Transaction_Events, and
(b) the TO-NORMAL flag is set in the Event_Enable property.

The value of Notify_Type is ignored and the notification is sent with a Notify Type of EVENT. The Event_Time_Stamps TO-NORMAL element is not affected. The Acked_Transitions TO_NORMAL bit is not affected.

The Notification Class object referenced by the Transaction_Notification_Class property is used to report Access Transaction Events. If Transaction_Notification_Class is not present, the Notification Class object referenced by Notification_Class is used. The Ack_Required property of the respective Notification Class object is ignored and the value FALSE is conveyed in the AckRequired parameter of the event notification message.

12.X.41 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.

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

12.X.42 Acked_Transitions

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

(a) upon receipt of the corresponding acknowledgment;
(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 acknowledgment 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 acknowledgment is expected).

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

12.X.43 Notify_Type

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

Access Transaction Events generated by the object are always of type EVENT, regardless of the value of this property.

12.X.44 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 X'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.45 Profile_Name

This optional property, of type CharacterString, is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name shall begin with a vendor identifier code (see Clause 23) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified here.

This standard defines only the format of the names of profiles. The definition of the profiles themselves is outside the scope of this standard.

[Change **Table 13-1**, p. 291]

Table 13-1. Standardized Objects That May Support COV Reporting

Object Type	Criteria	Properties Reported
…	…	…
Access Point [1]	*If Access_Event_Time changes at all* *or* *Status_Flags changes at all*	*Access_Event,* *Status_Flags,* *Access_Event_Tag,* *Access_Event_Time,* *Access_Event_Credential,* *Access_Event_Authentication_Factor (if present)*
…		

[1] *For COV contexts for this object type in Active_COV_Subscriptions in the Device object, the Monitored Property Reference shall contain the Access_Event property identifier.*

[Add new Clause **19.2.7**, p. 404]

19.2.7 Prioritization for Access Point Objects

Access Point objects interact with the Present_Value property of Access Door objects; however, if the Access Door objects are local to the device, they will not use BACnet services to do so. Each Access Point object has a Priority_For_Writing property that designates the priority to be used to command the Access Door objects.

[Add new **BACnetAccessEvent** production to Clause **21**, p. 448]

```
BACnetAccessEvent ::= ENUMERATED {
    none                                    (0),
    granted                                 (1),
    muster                                  (2),
    passback-detected                       (3),
    duress                                  (4),
    trace                                   (5),
    lockout-max-attempts                    (6),
    lockout-other                           (7),
    lockout-relinquished                    (8),
    locked-by-higher-priority               (9),
    out-of-service                          (10),
    out-of-service-relinquished             (11),
    accompaniment-by                        (12),
    authentication-factor-read              (13),
    authorization-delayed                   (14),
    verification-required                   (15),
    -- Enumerated values 128-511 are used for events which indicate that access has been denied.
    denied-deny-all                         (128),
    denied-unknown-credential               (129),
    denied-authentication-unavailable       (130),
    denied-authentication-factor-timeout    (131),
    denied-incorrect-authentication-factor  (132),
    denied-zone-no-access-rights            (133),
    denied-point-no-access-rights           (134),
    denied-no-access-rights                 (135),
    denied-out-of-time-range                (136),
    denied-threat-level                     (137),
    denied-passback                             (138),
    denied-unexpected-location-usage            (139),
    denied-max-attempts                         (140),
    denied-lower-occupancy-limit                (141),
    denied-upper-occupancy-limit                (142),
    denied-authentication-factor-lost           (143),
    denied-authentication-factor-stolen         (144),
    denied-authentication-factor-damaged        (145),
    denied-authentication-factor-destroyed      (146),
    denied-authentication-factor-disabled       (147),
    denied-authentication-factor-error          (148),
    denied-credential-unassigned                (149),
    denied-credential-not-provisioned           (150),
    denied-credential-not-yet-active            (151),
    denied-credential-expired                   (152),
    denied-credential-manual-disable            (153),
    denied-credential-lockout                   (154),
    denied-credential-max-days                  (155),
    denied-credential-max-uses                  (156),
    denied-credential-inactivity                (157),
    denied-credential-disabled                  (158),
    denied-no-accompaniment                     (159),
    denied-incorrect-accompaniment              (160),
    denied-lockout                              (161),
    denied-verification-failed                  (162),
    denied-verification-timeout                 (163),
    denied-other                                (164),
    ...
    }
    -- Enumerated values 0-511 are reserved for definition by ASHRAE. Enumerated values
    -- 512-65535 may be used by others subject to the procedures and constraints described
    -- in Clause 23.
```

[Add new **BACnetAccessThreatLevel** production to Clause **21**, p. 448]

```
    BACnetAccessThreatLevel ::= Unsigned(0..100)
```

[Note: **BACnetAuthenticationFactor** is defined in Addendum 135-2008*j*-6.]

[Note: **BACnetAuthenticationFactorType** is defined in Addendum 135-2008*j*-6.]

[Add new **BACnetAuthenticationPolicy** production to Clause **21**, p. 449]

```
    BACnetAuthenticationPolicy ::= SEQUENCE {
        policy            [0] SEQUENCE OF SEQUENCE {
                              credential-data-input       [0] BACnetDeviceObjectReference,
                              index                       [1] Unsigned
                              },
        order-enforced    [1] BOOLEAN,
        timeout           [2] Unsigned
        }
```

[Add new **BACnetAuthenticationStatus** production to Clause **21**, p. 449]

```
    BACnetAuthenticationStatus ::= ENUMERATED {
        not-ready                                (0),
```

ready (1),
disabled (2),
waiting-for-authentication-factor (3),
waiting-for-accompaniment (4),
waiting-for-verification (5),
in-progress (6)
}

[Add new **BACnetAuthorizationMode** production to Clause **21**, p. 449]

BACnetAuthorizationMode ::= ENUMERATED {
authorize (0),
grant-active (1),
deny-all (2),
verification-required (3),
authorization-delayed (4),
none (5),
...
}
-- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
-- 64-65535 may be used by others subject to the procedures and constraints described
-- in Clause 23.

[Change **BACnetObjectType** production in Clause **21**, p. 463]

BACnetObjectType ::= ENUMERATED {
access-credential *(32),*
access-door (30),
access-point *(33),*
access-rights *(34),*
access-user *(35),*
access-zone *(36),*
accumulator (23),
...
command (7),
credential-data-input *(37),*
device (8),
...
-- see access-door (30),
-- *see access-credential* *(32),*
-- *see access-point* *(33),*
-- *see access-rights* *(34),*
-- *see access-user* *(35),*
-- *see access-zone* *(36),*
-- *see credential-data-input* *(37),*
...
}

[Change **BACnetObjectTypesSupported** production in Clause **21**, pp. 463-464]

BACnetObjectTypesSupported ::= BIT STRING {
-- *access-credential* *(32),*
-- access-door (30),
-- *access-point* *(33),*
-- *access-rights* *(34),*
-- *access-user* *(35),*

-- *access-zone* *(36),*
-- accumulator (23),
...
command (7),
credential-data-input *(37),*
device (8),
...
access-door ~~(30)~~ *(30),*
-- *Objects added after 2008*
access-credential *(32),*
access-point *(33),*
access-rights *(34),*
access-user *(35),*
access-zone *(36),*
credential-data-input *(37)*
}

[Change **BACnetPropertyIdentifier** production, **Clause 21**, pp. 465-471]

BACnetPropertyIdentifier ::= ENUMERATED {
absentee-limit *(244),*
accepted-modes (175),
access-alarm-events *(245),*
access-doors *(246),*
access-event *(247),*
access-event-authentication-factor *(248),*
access-event-credential *(249),*
access-event-tag *(322),*
access-event-time *(250),*
access-transaction-events *(251),*
accompaniment *(252),*
accompaniment-time *(253),*
activation-time *(254),*
active-authentication-policy *(255),*
acked-transitions (0),
...
archive (13),
assigned-access-rights *(256),*
attempted-samples (124),
authentication-factors *(257),*
authentication-policy-list *(258),*
authentication-policy-names *(259),*
authentication-status *(260),*
authorization-mode *(261),*
auto-slave-discovery (169),
...
backup-failure-timeout (153),
belongs-to *(262),*
bias (14),
…
cov-resubscription-interval (128),
credential-disable *(263),*
credential-status *(264),*
credentials *(265),*
credentials-in-zone *(266),*
-- current-notify-time (129), This property was deleted in version 1 revision 3.

...		
daylight-savings-status	(24),	
days-remaining	*(267),*	
deadband	(25),	
...		
elapsed-active-time	(33),	
entry-points	*(268),*	
enable	(133),	-- renamed from previous version
...		
exception-schedule	(38),	
exit-points	*(269),*	
expected-shed-level	(214),	
expiry-time	*(270),*	
extended-time-enable	*(271),*	
failed-attempt-events	*(272),*	
failed-attempts	*(273),*	
failed-attempts-time	*(274),*	
fault-values	(39),	
...		
full-duty-baseline	(215),	
global-identifier	*(323),*	
high-limit	(45),	
...		
-- issue-confirmed-notifications	(51), This property was deleted in version 1 revision 4.	
last-access-event	*(275),*	
last-access-point	*(276),*	
last-credential-added	*(277),*	
last-credential-added-time	*(278),*	
last-credential-removed	*(279),*	
last-credential-removed-time	*(280),*	
last-notify-record	(173),	
…		
last-restore-time	(157),	
last-use-time	*(281),*	
life-safety-alarm-values	(166),	
...		
lock-status	(233),	
lockout	*(282),*	
lockout-relinquish-time	*(283),*	
log-buffer	(131),	
...		
masked-alarm-values	(234),	
master-exemption	*(284),*	
maximum-output	(61),	
...		
max-apdu-length-accepted	(62),	
max-failed-attempts	*(285),*	
max-info-frames	(63),	
...		
member-of	(159),	
members	*(286),*	
minimum-off-time	(66),	
...		
modification-date	(71),	
muster-point	*(287),*	
negative-access-rules	*(288),*	

node-subtype	(207),
...	
number-of-APDU-retries	(73),
number-of-authentication-policies	*(289),*
number-of-states	(74),
...	
object-type	(79),
occupancy-count	*(290),*
occupancy-count-adjust	*(291),*
occupancy-count-enable	*(292),*
occupancy-exemption	*(293),*
occupancy-lower-limit	*(294),*
occupancy-lower-limit-enforced	*(295),*
occupancy-state	*(296),*
occupancy-upper-limit	*(297),*
occupancy-upper-limit-enforced	*(298),*
operation-expected	(161),
...	
-- see event-parameters	(83),
passback-exemption	*(299),*
passback-mode	*(300),*
passback-timeout	*(301),*
polarity	(84),
positive-access-rules	*(302),*
prescale	(185),
...	
read-only	(99),
reason-for-disable	*(303),*
reason-for-halt	(100),
...	
subordinate-list	(211),
supported-formats	*(304),*
supported-format-classes	*(305),*
system-status	(112),
threat-authority	*(306),*
threat-level	*(307),*
time-delay	(113),
...	
total-record-count	(145),
trace-flag	*(308),*
tracking-value	(164),
transaction-notification-class	*(309),*
trigger	(205),
...	
update-time	(189),
user-external-identifier	*(310),*
user-information-reference	*(311),*
user-name	*(317),*
user-type	*(318),*
uses-remaining	*(319),*
utc-offset	(119),
...	
vendor-name	(121),
verification-time	*(326),*
vt-classes-supported	(122),
...	

```
window-samples                               (148),
zone-from                                    (320),
zone-members                                 (165),
zone-to                                      (321),
-- see maximum-value-timestamp               (149),
...
-- see secured-status                        (235),
                                             -- enumeration values 236-243 reserved for future addenda
-- see absentee-limit                        (244),
-- see access-alarm-events                   (245),
-- see access-doors                          (246),
-- see access-event                          (247),
-- see access-event-authentication-factor    (248),
-- see access-event-credential               (249),
-- see access-event-time                     (250),
-- see access-transaction–events             (251),
-- see accompaniment                         (252),
-- see accompaniment-time                    (253),
-- see activation-time                       (254),
-- see active-authentication-policy          (255),
-- see assigned-access-rights                (256),
-- see authentication-factors                (257),
-- see authentication-policy-list            (258),
-- see authentication-policy-names           (259),
-- see authentication-status                 (260),
-- see authorization-mode                    (261),
-- see belongs-to                            (262),
-- see credential-disable                    (263),
-- see credential-status                     (264),
-- see credentials                           (265),
-- see credentials-in-zone                   (266),
-- see days-remaining                        (267),
-- see entry-points                          (268),
-- see exit-points                           (269),
-- see expiry-time                           (270),
-- see extended-time-enable                  (271),
-- see failed-attempt-events                 (272),
-- see failed-attempts                       (273),
-- see failed-attempts-time                  (274),
-- see last-access-event                     (275),
-- see last-access-point                     (276),
-- see last-credential-added                 (277),
-- see last-credential-added-time            (278),
-- see last-credential-removed               (279),
-- see last-credential-removed-time          (280),
-- see last-use-time                         (281),
-- see lockout                               (282),
-- see lockout-relinquish-time               (283),
-- see master-exemption                      (284),
-- see max-failed-attempts                   (285),
-- see members                               (286),
-- see muster-point                          (287),
-- see negative-access-rules                 (288),
-- see number-of-authentication-policies     (289),
-- see occupancy-count                       (290),
-- see occupancy-count-adjust                (291),
-- see occupancy-count-enable              (292),
-- see occupancy-exemption                 (293),
-- see occupancy-lower-limit               (294),
-- see occupancy-lower-limit-enforced      (295),
-- see occupancy-state                     (296),
-- see occupancy-upper-limit               (297),
-- see occupancy-upper-limit-enforced      (298),
-- see passback-exemption                  (299),
-- see passback-mode                       (300),
-- see passback-timeout                    (301),
-- see positive-access-rules               (302),
-- see reason-for-disable                  (303),
-- see supported-formats                   (304),
-- see supported-format-classes            (305),
-- see threat-authority                    (306),
-- see threat-level                        (307),
-- see trace-flag                          (308),
-- see transaction-notification-class      (309),
-- see user-external-identifier            (310),
-- see user-information-reference          (311),
                                           -- enumeration values 312-316 reserved for future addenda
-- see user-name                           (317),
-- see user-type                           (318),
-- see uses-remaining                      (319),
-- see zone-from                           (320),
-- see zone-to                             (321),
-- see access-event-tag                    (322),
-- see global-identifier                   (323),
                                           -- enumeration values 324-325 reserved for future addenda
-- see verification-time                   (326)
}
```

[Change **Clause 21**, **BACnetPropertyStates** production, pp. 471-472]

[Note: BACnetAccessZoneOccupancyState is defined in Addendum 135-2008j-2]
[Note: BACnetAccessCredentialDisableReason is defined in Addendum 135-2008j-5]
[Note: BACnetAccessCredentialDisable is defined in Addendum 135-2008j-5]

```
BACnetPropertyStates ::= CHOICE {
    ...
    door-alarm-state                    [15] BACnetDoorAlarmState,
                                        -- context tags 16-29 reserved for future addenda
    access-event                        [30] BACnetAccessEvent,
    zone-occupancy-state                [31] BACnetAccessZoneOccupancyState,
    access-credential-disable-reason    [32] BACnetAccessCredentialDisableReason,
    access-credential-disable           [33] BACnetAccessCredentialDisable,
    authentication-status               [34] BACnetAuthenticationStatus,
    ...
    }
```

[Change **Table 23-1**, p. 480**]**

Table 23-1. Extensible Enumerations

Enumeration Name	Reserved Range	Maximum Value
error-class	0-63	65535
error-code	0-255	65535
BACnetAbortReason	0-63	255
BACnetAccessAuthenticationFactorDisable	*0-63*	*65535*
BACnetAccessCredentialDisable	*0-63*	*65535*
BACnetAccessCredentialDisableReason	*0-63*	*65535*
BACnetAccessEvent	*0-511*	*65535*
BACnetAccessUserType	*0-63*	*65535*
BACnetAccessZoneOccupancyState	*0-63*	*65535*
BACnetAuthorizationMode	*0-63*	*65535*
BACnetDeviceStatus	0-63	65535
…		

[Add new object type structure to **ANNEX C**, p. 497]

```
ACCESS-POINT ::= SEQUENCE {
    object-identifier                   [75]    BACnetObjectIdentifier,
    object-name                         [77]    CharacterString,
    object-type                         [79]    BACnetObjectType,
    description                         [28]    CharacterString   OPTIONAL,
    status-flags                        [111]   BACnetStatusFlags,
    event-state                         [36]    BACnetEventState,
    reliability                         [103]   BACnetReliability,
    out-of-service                      [81]    BOOLEAN,
    authentication-status               [260]   BACnetAuthenticationStatus,
    active-authentication-policy        [255]   Unsigned,
    number-of-authentication-policies   [289]   Unsigned,
    authentication-policy-list          [258]   SEQUENCE OF BACnetAuthenticationPolicy OPTIONAL,
                                                -- accessed as a BACnetARRAY
    authentication-policy-names         [259]   SEQUENCE OF CharacterString OPTIONAL,
                                                -- accessed as a BACnetARRAY
    authorization-mode                  [261]   BACnetAuthorizationMode,
    verification-time                   [326]   Unsigned OPTIONAL,
    lockout                             [282]   BOOLEAN OPTIONAL,
    lockout-relinquish-time             [283]   Unsigned OPTIONAL,
    failed-attempts                     [273]   Unsigned OPTIONAL,
    failed-attempt-events               [272]   SEQUENCE OF BACnetAccessEvent OPTIONAL,
    max-failed-attempts                 [285]   Unsigned OPTIONAL,
    failed-attempts-time                [274]   Unsigned OPTIONAL,
    threat-level                        [307]   BACnetAccessThreatLevel OPTIONAL,
    occupancy-upper-limit-enforced      [298]   BOOLEAN OPTIONAL,
    occupancy-lower-limit-enforced      [295]   BOOLEAN OPTIONAL,
    occupancy-count-adjust              [291]   BOOLEAN OPTIONAL,
    accompaniment-time                  [253]   Unsigned OPTIONAL,
    access-event                        [247]   BACnetAccessEvent,
    access-event-tag                    [322]   Unsigned,
    access-event-time                   [250]   BACnetTimeStamp,
    access-event-credential             [249]   BACnetDeviceObjectReference,
    access-event-authentication-factor  [248]   BACnetAuthenticationFactor OPTIONAL,
    access-doors                        [246]   SEQUENCE OF  BACnetDeviceObjectReference,
                                                -- accessed as a BACnetARRAY
    priority-for-writing                [88]    Unsigned(1..16),
    muster-point                     [287]   BOOLEAN OPTIONAL,
    zone-to                          [321]   BACnetDeviceObjectReference    OPTIONAL,
    zone-from                        [320]   BACnetDeviceObjectReference    OPTIONAL,
    notification-class               [17]    Unsigned    OPTIONAL,
    transaction-notification-class   [309]   Unsigned    OPTIONAL,
    access-alarm-events              [245]   SEQUENCE OF BACnetAccessEvent OPTIONAL,
    access-transaction-events        [251]   SEQUENCE OF  BACnetAccessEvent 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
    profile-name                     [168]   CharacterString   OPTIONAL
    }
```

[Add new **Clause D.X**, p. 534]

D.X Example of an Access Point object

In this example, authentication and authorization at an entrance to a secured zone is represented as an Access Point object "Main Entrance 1". The secured zone to enter is assumed to be represented by an Access Zone object instance 23.

There are three Credential Data Input objects used, all located in remote Device 12:
- (a) Proximity Card Reader – Instance 3
- (b) PIN Keypad – Instance 4
- (c) Iris Scanner – Instance 5

Three different authentication policies are defined and can be activated:
- (a) OFFICE HOURS – Low Security – Proximity Card Reader or PIN Keypad
- (b) LATE SHIFT – Medium Security   Proximity Card Reader and PIN Keypad
- (c) NIGHT – High Security – Proximity Card Reader and Iris Scanner

The current authorization mode is AUTHORIZE, authentication and authorization is in effect. The current authentication policy is set to "OFFICE-HOURS".

The Access Point is currently not locked down. The Access Point will go into a locked down state after three failed attempts. A failed attempt is specified to occur when one of the following access events are generated:
1. DENIED_UNKNOWN_CREDENTIAL - the credential is unknown
2. DENIED_ZONE_NO_ACCESS_RIGHTS – the credential does not have any access rights at the zone
3. DENIED_POINT_NO_ACCESS_RIGHTS – the credential does not have any access rights at the access point
4. DENIED_THREAT_LEVEL -the credential does not have a sufficient threat authority to enter at this access point

The current threat level is 20.

Access Door objects 44 and 45, local in the device, are controlled at a command priority 12.

Occupancy lower limit is not checked but upper limit is checked if present at the Access Zone. This access point will adjust the occupancy count of both the exit and entry zones when a credential holder passes through this access point.

The last event to occur at this access point was a passback violation which happened on March 21, 2008, at 6:50 PM as part of access transaction 1116. The authentication factor used is a 32-bit simple number and has a value of 1334234499. This authentication factor is contained in the credential object with instance 41445.

Property:	Object_Identifier =	(Access Point, Instance 2)
Property:	Object_Name =	"MAIN-ENTRANCE-01"
Property:	Object_Type =	ACCESS_POINT

Property:	Description =	"Main Entrance 1"
Property:	Status_Flags =	{FALSE, FALSE, FALSE, FALSE}
Property:	Event_State =	(NORMAL)
Property:	Reliability =	NO_FAULT_DETECTED
Property:	Out_Of_Service =	FALSE
Property:	Authentication_Status	READY
Property:	Active_Authentication_Policy =	1
Property:	Number_Of_Authentication_Policies =	3
Property:	Authentication_Policy_List =	{
		((((Device, Instance 12) (Credential Data Input, Instance 3)),1),
		(((Device, Instance 12) (Credential Data Input, Instance 4)),1),
		FALSE, 0),
		((((Device, Instance 12) (Credential Data Input, Instance 3)),1),
		(((Device, Instance 12) (Credential Data Input, Instance 4)),2),
		TRUE, 0),
		((((Device, Instance 12) (Credential Data Input, Instance 3)),1),
		(((Device, Instance 12) (Credential Data Input, Instance 5)),2),
		TRUE, 30)
		}
Property:	Authentication_Policy_Names =	("OFFICE_HOURS", "LATE_SHIFT", "NIGHT")
Property:	Authorization_Mode =	AUTHORIZE
Property:	Verification_Time =	15
Property:	Lockout =	FALSE
Property:	Lockout_Relinquish_Time =	60
Property:	Failed_Attempts =	0
Property:	Failed_Attempt_Events =	(DENIED_UNKNOWN_CREDENTIAL,
		DENIED_ZONE_NO_ACCESS_RIGHTS,
		DENIED_POINT_NO_ACCESS_RIGHTS,
		DENIED_THREAT_LEVEL)
Property:	Max_Failed_Attempts =	3
Property:	Failed_Attempts_Time =	10
Property:	Threat_Level =	20
Property:	Occupancy_Upper_Limit_Enforced =	TRUE
Property:	Occupancy_Lower_Limit_Enforced =	FALSE
Property:	Occupancy_Count_Adjust =	TRUE
Property:	Accompaniment_Time =	15
Property:	Access_Event =	PASSBACK_DETECTED
Property:	Access_Event_Tag	1116
Property:	Access_Event_Time =	((21 MAR 2008, FRIDAY), 18:50:21.2)
Property:	Access_Event_Credential =	(, (Access Credential, Instance 41445))
Property:	Access_Event_Authentication_Factor =	(SIMPLE_NUMBER32, 1334234499)
Property:	Access_Doors =	((, (Access Door, Instance 44)),
		(, (Access Door, Instance 45)))
Property:	Priority_For_Writing =	12
Property:	Muster_Point =	FALSE
Property:	Zone_To =	(, (Access Zone, Instance 23))
Property:	Zone_From =	(, (Access Zone, Instance 1))
Property:	Notification_Class =	39
Property:	Transaction_Notification_Class =	48
Property:	Access_Alarm_Events =	(DURESS, PASSBACK_DETECTED)
Property:	Access_Transaction_Events =	(GRANTED, DENIED_PASSBACK,
		AUTHENTICATION_FACTOR_READ,
		DENIED_UNKNOWN_CREDENTIAL,
		DENIED_ZONE_NO_ACCESS_RIGHTS,
		DENIED_POINT_NO_ACCESS_RIGHTS)
Property:	Event_Enable =	{TRUE, TRUE, TRUE}

Property:	Acked_Transitions =	{TRUE, TRUE, FALSE}
Property:	Notify_Type =	EVENT
Property:	Event_Time_Stamps =	((*-*-*, *:*:*.*), (*-*-*, *:*:*.*), ((21 MAR 2008, FRIDAY), 18:50:21.3))

135-2008*j*-2. Add a new Access Zone object type.

Rationale
Support for access control in BACnet requires a standardized object whose properties represent the externally visible characteristics associated with a secured geographical zone for which authentication and authorization of a credential takes place to obtain physical access.

Addendum 135-2008*j*-2

[Insert new Clause **12.X**, p. 288]

12.X Access Zone Object Type

The Access Zone object type defines a standardized object whose properties represent the externally visible characteristics associated with a secured geographical zone for which authentication and authorization of a credential takes place to obtain physical access. Entrance to the zone takes place through entry access controlled points while the zone is exited through exit access controlled points. These access controlled points are represented by Access Point objects.

The Access Zone object may optionally support occupancy counting and the specification of occupancy limits. Access may be denied if limits are violated. The enforcement rules are specified at the corresponding entry and/or exit Access Point objects. The Access Zone object's Occupancy_State is the state of occupancy. This state is derived from the actual occupancy count and occupancy limits.

Intrinsic reporting of this object is based on the Occupancy_State property and uses the CHANGE_OF_STATE algorithm.

"Who's in" reporting is supported through a list of the credentials which are currently in the zone. Credentials are added on successful entrance through an access controlled entry point and removed on successful exit through an access controlled exit point. This list may also be maintained based on time conditions or other local methods.

The Access Zone object supports passback detection and allows the selection of hard, soft or no-passback enforcement. A passback violation occurs when entrance to this zone is requested at an access controlled point while the credential is assumed to be in this zone. The list of credentials in this zone may be used to detect a passback violation.

A specific access controlled zone may be represented by a single Access Zone object in a single device, or in multiple devices by one Access Zone object per device. When an access controlled zone is represented in multiple devices, the representing Access Zone objects may not have the same Object_Identifier in each device; however, they may be identified using the Global_Identifier property. It is a local matter as to how these objects are synchronized.

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

Table 12-X. Properties of the Access Zone Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Global_Identifier	Unsigned32	W
Occupancy_State	BACnetAccessZoneOccupancyState	R
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Reliability	BACnetReliability	R[1]
Out_Of_Service	BOOLEAN	R

Occupancy_Count	Unsigned	O[1,3,4]
Occupancy_Count_Enable	BOOLEAN	O[3,4]
Adjust_Value	INTEGER	O[3,4,5]
Occupancy_Upper_Limit	Unsigned	O
Occupancy_Lower_Limit	Unsigned	O
Credentials_In_Zone	List of BACnetDeviceObjectReference	O
Last_Credential_Added	BACnetDeviceObjectReference	O
Last_Credential_Added_Time	BACnetDateTime	O
Last_Credential_Removed	BACnetDeviceObjectReference	O
Last_Credential_Removed_Time	BACnetDateTime	O
Passback_Mode	BACnetAccessPassbackMode	O
Passback_Timeout	Unsigned	O[2]
Entry_Points	List of BACnetDeviceObjectReference	R
Exit_Points	List of BACnetDeviceObjectReference	R
Time_Delay	Unsigned	O[3]
Notification_Class	Unsigned	O[3]
Alarm_Values	List of BACnetAccessZoneOccupancyState	O[3]
Event_Enable	BACnetEventTransitionBits	O[3]
Acked_Transitions	BACnetEventTransitionBits	O[3]
Notify_Type	BACnetNotifyType	O[3]
Event_Time_Stamps	BACnetARRAY[3] of BACnetTimeStamp	O[3]
Profile_Name	CharacterString	O

[1] These properties, if present, shall be writeable when Out_Of_Service is TRUE.
[2] If this property is present, then Passback_Mode shall be present.
[3] These properties are required if the object supports intrinsic reporting.
[4] These properties are required if the object supports occupancy counting.
[5] The Adjust_Value property shall be writeable if 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 ACCESS_ZONE.

12.X.4 Description

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

12.X.5 Global_Identifier

This property, of type Unsigned32, is a unique identifier which is used to globally identify the access controlled zone this object represents. This value may be used to identify Access Zone objects in multiple devices that represent the same access controlled zone.

If this value is assigned, it shall be unique internetwork-wide and all Access Zone objects in all devices that represent this access controlled zone shall have this value. A value of zero indicates that no global identifier is assigned.

12.X.6 Occupancy_State

This property, of type BACnetAccessZoneOccupancyState, reflects the occupancy state of the zone.

BACnetAccessZoneOccupancyState is an enumeration of possible occupancy states.

NORMAL	This is the occupancy state when occupancy counting is enabled and no other standard or proprietary states are applicable.
BELOW_LOWER_LIMIT	If Occupancy_Lower_Limit property is present and the Occupancy_Count property is lower than this value.
AT_LOWER_LIMIT	If Occupancy_Lower_Limit property is present and the Occupancy_Count property is equal to this value.
AT_UPPER_LIMIT	If Occupancy_Upper_Limit property is present and the Occupancy_Count property is equal to this value.
ABOVE_UPPER_LIMIT	If Occupancy_Upper_Limit property is present and the Occupancy_Count property is greater than this value.
DISABLED	This is the occupancy state when occupancy counting is disabled for this object. Occupancy counting is disabled when the Occupancy_Count_Enable property is FALSE.
NOT_SUPPORTED	This is the occupancy state when occupancy counting is not supported by this object.
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values to indicate other states based on the Occupancy_Count property other than those defined by this standard. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

12.X.7 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of the Access Zone object. A more detailed status may 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 has a value of NORMAL, otherwise logical TRUE (1).

FAULT	Logical TRUE (1) if the Reliability is not NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	The value of this flag shall be logical FALSE (0).
OUT_OF_SERVICE	Logical TRUE (1) if the Out_Of_Service property has a value of TRUE, otherwise logical FALSE (0).

12.X.8 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. If the object supports intrinsic reporting, then the Event_State property shall indicate the event state of the object. If the object does not support intrinsic reporting, then:

(a) if the Reliability is NO_FAULT_DETECTED, then Event_State shall be NORMAL, or
(b) if the Reliability is not NO_FAULT_DETECTED, then Event_State shall be FAULT.

12.X.9 Reliability

The Reliability property, of type BACnetReliability, provides an indication of whether the Occupancy_State, Occupancy_Count and/or Credentials_In_Zone properties of this object are "reliable" as far as the BACnet Device can determine and, if not, why. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, CONFIGURATION_ERROR, UNRELIABLE_OTHER}.

The Reliability property shall be writable when Out_Of_Service is TRUE.

12.X.9.1 Conditions for Generating a TO-FAULT Event

A TO-FAULT event is generated under these conditions:

(a) the Reliability property becomes not equal to NO_FAULT_DETECTED, and
(b) the TO-FAULT flag is enabled in the Event_Enable property.

12.X.9.2 Conditions for Generating a TO-NORMAL Event

A TO-NORMAL event is generated under these conditions:

(a) the Reliability property becomes equal to NO_FAULT_DETECTED, and
(b) the TO-NORMAL flag is set in the Event_Enable property.

12.X.10 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the object is out of service.

When the object is out of service, the Reliability property and the corresponding state of the FAULT flag of the Status_Flags property shall be decoupled and the Reliability property 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 Reliability property shall respond to changes made to this property while Out_Of_Service is TRUE.

If occupancy counting is supported and the object is out of service, then the Occupancy_Count property is decoupled from the processing of occupancy counting. In addition, writing to the Adjust_Value property shall not modify the Occupancy_Count. The Occupancy_Count property 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 Occupancy_State property shall respond to changes made to this property while Out_Of_Service is TRUE.

12.X.11 Occupancy_Count

This optional property, of type Unsigned, is used to indicate the actual occupancy count of a zone. If the value of the Occupancy_Count_Enable property is FALSE, then this property shall have a value of zero. The value of the Occupancy_Count property may be adjusted by writing to the Adjust_Value property. The Occupancy_Count property shall be writable when Out_Of_Service is TRUE. When Out_Of_Service becomes FALSE, it is a local matter as to what value this property is set to.

If occupancy counting is supported by this object, then this property shall be present.

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

12.X.12 Occupancy_Count_Enable

This optional property, of type BOOLEAN, indicates whether occupancy counting is enabled (TRUE) or not (FALSE).

If this property has a value of FALSE, then the Occupancy_State property shall have a value of DISABLED.

When this property changes from FALSE to TRUE it is a local matter as to what value the Occupancy_Count property is set to.

If occupancy counting is supported by this object, then this property shall be present.

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

12.X.13 Adjust_Value

This optional property, of type INTEGER, shall adjust the Occupancy_Count property when written.

The following series of operations shall be performed atomically when this property is written and the value of the Occupancy_Count_Enable property is TRUE:

(1) The value written to Adjust_Value shall be stored in the Adjust_Value property.

(2) If the value written is non-zero, then this value shall be added to the value of the Occupancy_Count property. If the value written is negative and the resulting value of the Occupancy_Count property would be less than zero, then the Occupancy_Count property shall be set to zero. If the value written is zero, then the value of the Occupancy_Count property shall be set to zero.

When this property is written and the value of the Occupancy_Count_Enable property is FALSE, then the Adjust_Value property shall be set to zero.

If Adjust_Value has never been written or the Occupancy_Count_Enable property is FALSE, then this property shall have a value of zero.

If occupancy counting is supported by this object, then this property shall be present and writable.

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

12.X.14 Occupancy_Upper_Limit

This optional property, of type Unsigned, specifies the occupancy upper limit of the zone. If this property has a value of zero, then there is no upper limit. If this value is not zero, it shall be greater than the value of the Occupancy_Lower_Limit, if present.

12.X.15 Occupancy_Lower_Limit

This optional property, of type Unsigned, specifies the occupancy lower limit of the zone. If this property has a value of zero, then there is no lower limit.

12.X.16 Credentials_In_Zone

This optional property, of type List of BACnetDeviceObjectReference, is used to list references to those Access Credential objects that represent credentials assumed to be in this zone. This information may be used to verify whether a specific credential is already in the zone for passback detection purposes. If the zone does not support listing credentials, then this list, if present, shall be empty. It is a local matter as to how this list is updated.

12.X.17 Last_Credential_Added

This optional property, of type BACnetDeviceObjectReference, indicates the reference to the Access Credential object which has last been added to the Credentials_In_Zone property. If no credential has been added yet, then this reference shall contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present. If COV property subscriptions for this property are present, then any update, even one with the same value, is reported by a COV notification.

12.X.18 Last_Credential_Added_Time

This optional property, of type BACnetDateTime, indicates the date and time when a reference to an Access Credential object has last been added to the Credentials_In_Zone property. If this property is present, but no credential has yet been added, then this property shall not convey an actual time and shall contain a value of X'FF' in all octets.

12.X.19 Last_Credential_Removed

This optional property, of type BACnetDeviceObjectReference, indicates the reference to the Access Credential object which has last been removed from the Credentials_In_Zone property. If no credential has been removed yet, then this reference shall contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present. If COV property subscriptions for this property are present, then any update, even one with the same value, is reported by a COV notification.

12.X.20 Last_Credential_Removed_Time

This optional property, of type BACnetDateTime, indicates the date and time when a reference to an Access Credential object has last been removed from the Credentials_In_Zone property. If this property is present, but no credential has yet been removed, then this property shall not convey an actual time and shall contain a value of X'FF' in all octets.

12.X.21 Passback_Mode

This optional property, of type BACnetAccessPassbackMode, specifies how all Access Point objects that represent entry points to the access controlled zone this object represents shall handle passback violations. Passback modes are:

PASSBACK_OFF	Passback violations are not checked.

HARD_PASSBACK	Passback violations are checked, enforced and reported. When a passback violation is detected, the Access_Event Property of the corresponding Access Point object shall be set to DENIED_PASSBACK and the authorization for the credential shall fail.
SOFT_PASSBACK	Passback violations are checked and reported but not enforced. When a passback violation is detected, the Access_Event Property of the corresponding Access Point object shall be set to PASSBACK_DETECTED.

12.X.22 Passback_Timeout

This optional property, of type Unsigned, specifies the passback timeout in minutes. The timeout is evaluated individually for every credential used to enter the zone. The timeout period for a particular credential begins at the time of successful access to the zone. After the timeout has expired for a particular credential, a passback violation of this credential will no longer be detected. A value of zero or absence of this property indicates passback violations will never time out.

If Passback_Timeout is present, Passback_Mode shall be present.

12.X.23 Entry_Points

This property, of type List of BACnetDeviceObjectReference, references all Access Point objects that lead into the zone.

12.X.24 Exit_Points

This property, of type List of BACnetDeviceObjectReference, references all Access Point objects that lead out of the zone.

12.X.25 Time_Delay

This optional property, of type Unsigned, shall specify the minimum period of time in seconds that the Occupancy_State remains:

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

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

12.X.26 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.27 Alarm_Values

This optional property, of type List of BACnetAccessZoneOccupancyState, shall specify any states the Occupancy_State shall equal before a TO-OFFNORMAL event is generated. This property is required if intrinsic reporting is supported by this object.

12.X.27.1 Conditions for Generating a TO-OFFNORMAL Event

A TO-OFFNORMAL event is generated under these conditions:

(a) the Occupancy_State equals at least one of the values in the Alarm_Values list, and

(b) the Occupancy_State remains equal to any value within the Alarm_Values list for a minimum period of time specified by the Time_Delay property, and
(c) the TO-OFFNORMAL flag is enabled in the Event_Enable property.

12.X.27.2 Conditions for Generating a TO-NORMAL Event

Once equal, if the Occupancy_State becomes not equal to any of the states in this property, then a TO-NORMAL event is generated under these conditions:

(a) the Occupancy_State remains not equal to any of the states in the Alarm_Values property for a minimum period of time specified by the Time_Delay property, and
(b) the TO-NORMAL flag is enabled in the Event_Enable property.

12.X.28 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. This property is required if intrinsic reporting is supported by this object.

12.X.29 Acked_Transitions

This optional property, of type BACnetEventTransitionBits, shall convey three flags that separately indicate the receipt of acknowledgments 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 acknowledgment;
(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 acknowledgment 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 acknowledgment is expected).

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

12.X.30 Notify_Type

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

12.X.31 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 X'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.32 Profile_Name

This optional property, of type CharacterString, is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name begins with a vendor identifier code (see Clause 23) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified here. This standard defines only the format of the names of profiles. The definition of the profiles themselves is outside the scope of this standard.

[Change Clause **13.2**, p. 292]

…

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.

In the case of the Access Zone object, the Alarm_Values property lists each of the possible Occupancy_State states that shall be interpreted as OFFNORMAL. All other Occupancy_State 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** and **Table 13-3**, pp. 293-294]

Table 13-2. Standard Objects That May Support Intrinsic Reporting

Object Type	Criteria	Event Type
...		
Access Zone	*If Occupancy_State becomes equal to one of the values contained in Alarm_Values AND remains equal to any value within the Alarm_Values list for longer than Time_Delay AND the new transition is enabled in Event_Enable*	*CHANGE_OF_STATE*
…		

Table 13-3. Standard Object Property Values Returned in Notifications

Object	Event Type	Notification Parameters	Referenced Object's Properties
...			
Access Zone	*CHANGE_OF_STATE*	*New_State* *Status_Flags*	*Occupancy_State* *Status_Flags*
….			

[Add new **BACnetAccesssPassbackMode** production to Clause **21**, p. 448]

```
BACnetAccessPassbackMode ::= ENUMERATED {
    passback-off      (0),
    hard-passback     (1),
    soft-passback     (2)
    }
```

[Add new **BACnetAccesssZoneOccupancyState** production to Clause **21**, p. 448]

```
BACnetAccessZoneOccupancyState ::= ENUMERATED {
    normal                 (0),
    below-lower-limit      (1),
    at-lower-limit         (2),
    at-upper-limit         (3),
    above-upper-limit      (4),
    disabled               (5),
    not-supported          (6),
    …
}
-- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
-- 64-65535 may be used by others subject to the procedures and constraints described
-- in Clause 23.
```

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetObjectTypesSuppported** BITSTRING appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to **Clause 21**, **BACnetPropertyStates** production appear in Addendum 135-2008*j*-1]

[Note: changes to **Table 23-1 Extensible Enumerations** appear in Addendum 135-2008*j*-1.]

[Add new object type structure to **ANNEX C**, p. 497]

```
ACCESS-ZONE ::= SEQUENCE {
    object-identifier            [75]    BACnetObjectIdentifier,
    object-name                  [77]    CharacterString,
    object-type                  [79]    BACnetObjectType,
    description                  [28]    CharacterString OPTIONAL,
    global-identifier            [323]   Unsigned32,
    occupancy-state              [296]   BACnetAccessZoneOccupancyState,
    status-flags                 [111]   BACnetStatusFlags,
    event-state                  [36]    BACnetEventState,
    reliability                  [103]   BACnetReliability,
    out-of-service               [81]    BOOLEAN,
    occupancy-count              [290]   Unsigned OPTIONAL,
    occupancy-count-enable       [292]   BOOLEAN OPTIONAL,
    adjust-value                 [176]   INTEGER OPTIONAL,
    occupancy-upper-limit        [297]   Unsigned OPTIONAL,
    occupancy-lower-limit        [294]   Unsigned OPTIONAL,
    credentials-in-zone          [266]   SEQUENCE OF BACnetDeviceObjectReference OPTIONAL,
    last-credential-added        [277]   BACnetDeviceObjectReference OPTIONAL,
    last-credential-added-time   [278]   BACnetDateTime OPTIONAL,
    last-credential-removed      [279]   BACnetDeviceObjectReference OPTIONAL,
    last-credential-removed-time [280]   BACnetDateTime OPTIONAL,
    passback-mode                [300]   BACnetAccessPassbackMode OPTIONAL,
    passback-timeout             [301]   Unsigned OPTIONAL,
    entry-points                 [268]   SEQUENCE OF BACnetDeviceObjectReference,
    exit-points                  [269]   SEQUENCE OF BACnetDeviceObjectReference,
    time-delay                   [113]   Unsigned OPTIONAL,
    notification-class           [17]    Unsigned OPTIONAL,
    alarm-values                 [7]     SEQUENCE OF BACnetAccessZoneOccupancyState 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
    profile-name                 [168]   CharacterString OPTIONAL
}
```

[Add new **Clause D.X**, p. 534]

D.X Example of an Access Zone object

In this example, an access controlled zone is represented as an Access Zone object. The same access controlled zone is represented in multiple devices and its global identifier is 23.

This Access Zone has an upper occupancy limit of 100. The zone will report an AT_UPPER_LIMIT alarm if the occupancy count becomes equal to the upper occupancy limit and will report an ABOVE_UPPER_LIMIT alarm if the count goes above the upper occupancy limit. The state of this zone is NORMAL because the current occupancy is below the occupancy upper limit. There are two access credentials currently in the zone. Passback detection is switched off.

The entry Access Points (Local Device, Instance 2 and 3) are leading into the zone and the exit Access Points (Local Device, Instance 12 and 13) are leading out of the Access Zone.

Property:	Object_Identifier =	(Access Zone, Instance 23)
Property:	Object_Name =	"OFFICE_MAIN_FLOOR"
Property:	Object_Type =	ACCESS_ZONE
Property:	Description =	"Office Main Floor"
Property:	Global_Identifier =	23
Property:	Occupancy_State =	NORMAL
Property:	Status_Flags =	{FALSE, FALSE, FALSE, FALSE}
Property:	Event_State =	NORMAL
Property:	Reliability =	NO_FAULT_DETECTED
Property:	Out_Of_Service =	FALSE
Property:	Occupancy_Count =	2
Property:	Occupancy_Count_Enable =	TRUE
Property:	Adjust_Value =	0
Property:	Occupancy_Upper_Limit =	100
Property:	Occupancy_Lower_Limit =	0
Property:	Credentials_In_Zone =	((,(Access Credential, Instance 12110)), (,(Access Credential, Instance 12245)))
Property:	Last_Credential_Added =	(,(Access Credential, Instance 12110))
Property:	Last_Credential_Added_Time =	((16 APRIL 2008, WEDNESDAY), 09:05:23.1))
Property:	Last_Credential_Removed =	(,(Access Credential, Instance 14430))
Property:	Last_Credential_Removed_Time =	((16 APRIL 2008, WEDNESDAY), 08:59:46.8))
Property:	Passback_Mode =	PASSBACK_OFF
Property:	Passback_Timeout =	10
Property:	Entry_Points =	((,(Access Point, Instance 2)), (,(Access Point, Instance 3)))
Property:	Exit_Points =	((,(Access Point, Instance 12)), (,(Access Point, Instance 13)))
Property:	Time_Delay =	10
Property:	Notification_Class =	39
Property:	Alarm_Values =	(AT_UPPER_LIMIT, ABOVE_UPPER_LIMIT)
Property:	Event_Enable =	{TRUE, TRUE, TRUE}
Property:	Acked_Transitions =	{TRUE, TRUE, TRUE}
Property:	Notify_Type =	EVENT
Property:	Event_Time_Stamps =	((16 APRIL 2008, WEDNESDAY), 09:11:14.3)), (*-*-*,*:*:*.*), ((16 APRIL 2008, WEDNESDAY), 11:13:21.3)))

135-2008*j*-3. Add a new Access User object type.

Rationale
Support for access control in BACnet requires a standardized object whose properties represent the externally visible characteristics associated with a user of a physical access control system, which may be an individual person, a group of users, or an asset.

Addendum 135-2008*j*-3

[Insert new Clause **12.X** and Table **12-X**, p. 288]

12.X Access User Object Type

The Access User object type defines a standardized object whose properties represent the externally visible characteristics associated with a user of a physical access control system.

The Access User object is used to represent an individual person, a group of users, or an asset. Relationships among access users are supported for representation of hierarchical organizations (e.g., companies, departments, or groups of any kind) or for representing ownership of assets.

The Access User object is not directly involved in authentication and authorization. It is used for informational purposes. It can hold a name, a reference number and a reference to an external system providing details of the access user.

The Access User object can have Access Credential objects assigned. This information can be used for administrative purposes (e.g., disabling all credentials of a person).

When a user is represented in multiple devices, the representing Access User objects may not have the same Object_Identifier in each device; however, they may be identified using the Global_Identifier property. It is a local matter as to how these objects are synchronized.

The Access User object type and its properties are summarized in Table 12-X and described in detail in this subclause.

Table 12-X. Properties of the Access User Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Global_Identifier	Unsigned32	W
Status_Flags	BACnetStatusFlags	R
Reliability	BACnetReliability	R
User_Type	BACnetAccessUserType	R
User_Name	CharacterString	O
User_External_Identifier	CharacterString	O
User_Information_Reference	CharacterString	O
Members	List of BACnetDeviceObjectReference	O
Member_Of	List of BACnetDeviceObjectReference	O
Credentials	List of BACnetDeviceObjectReference	R
Profile_Name	CharacterString	O

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 ACCESS_USER.

12.X.4 Description

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

12.X.5 Global_Identifier

This property, of type Unsigned32, is a unique identifier which is used to globally identify the access user this object represents. This value may be used to identify Access User objects in multiple devices which represent the same access user.

If this value is assigned, it shall be unique internetwork-wide and all Access User objects in all devices which represent this access user shall have this value. A value of zero indicates that no global identifier is assigned.

12.X.6 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of this object. A more detailed status may 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	The value of this flag shall be logical FALSE (0).
FAULT	Logical TRUE (1) if the Reliability is not NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	The value of this flag shall be logical FALSE (0).
OUT_OF_SERVICE	The value of this flag shall be logical FALSE (0).

12.X.7 Reliability

The Reliability property, of type BACnetReliability, provides an indication of whether this object is "reliable" as far as the BACnet Device can determine and, if not, why. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, CONFIGURATION_ERROR, UNRELIABLE_OTHER}.

12.X.8 User_Type

This property, of type BACnetAccessUserType, specifies the access user type this object represents. The following user types are defined:

ASSET	The Access User object represents a physical item.
GROUP	The Access User object represents a group of access users.
PERSON	The Access User object represents an individual person.
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values to allow proprietary access user types other than those defined by this standard. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

12.X.9 User_Name

This optional property, of type CharacterString, is a string of printable characters which specifies the name of the access user. The content is not restricted and can contain multiple lines.

12.X.10 User_External_Identifier

This optional property, of type CharacterString, specifies an external identifier associated with the access user. While the content is typically unique, its interpretation is a local matter.

12.X.11 User_Information_Reference

This optional property, of type CharacterString, specifies a reference to an external system where additional information of the user can be found. The interpretation of the content is a local matter.

12.X.12 Members

This optional property, of type List of BACnetDeviceObjectReference, references the Access User objects that represent the associated access users. Each object referenced shall be an Access User object.

12.X.13 Member_Of

This optional property, of type List of BACnetDeviceObjectReference, references the Access User objects that represent the access users to which this access user is associated. Each object referenced shall be an Access User object.

12.X.14 Credentials

This property, of type List of BACnetDeviceObjectReference, references all Access Credential objects that represent those credentials which are owned by this access user. Each object referenced shall be an Access Credential object.

12.X.15 Profile_Name

This optional property, of type CharacterString, is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name begins with a vendor identifier code (see Clause 23) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified here. This standard defines only the format of the names of profiles. The definition of the profiles themselves is outside the scope of this standard.

[Add new **BACnetAccessUserType** enumeration to Clause **21**, p. 448]

```
BACnetAccessUserType ::= ENUMERATED {
    asset     (0),
    group     (1),
    person    (2),
    ...
    }
    -- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
    -- 64-65535 may be used by others subject to the procedures and constraints described
    -- in Clause 23.
```

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetObjectTypesSupported** BITSTRING appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to **Table 23-1 Extensible Enumerations** appear in Addendum 135-2008*j*-1.]

[Add new object type structure to **ANNEX C**, p. 497]

```
ACCESS-USER ::= SEQUENCE {
    object-identifier              [75]     BACnetObjectIdentifier,
    object-name                    [77]     CharacterString,
    object-type                    [79]     BACnetObjectType,
    description                    [28]     CharacterString OPTIONAL,
    global-identifier              [323]    Unsigned32,
    status-flags                   [111]    BACnetStatusFlags,
    reliability                    [103]    BACnetReliability,
    user-type                      [318]    BACnetAccessUserType,
    user-name                      [317]    CharacterString OPTIONAL,
    user-external-identifier       [310]    CharacterString OPTIONAL,
    user-information-reference     [311]    CharacterString OPTIONAL,
    members                        [286]    SEQUENCE OF BACnetDeviceObjectReference OPTIONAL,
    member-of                      [159]    SEQUENCE OF BACnetDeviceObjectReference OPTIONAL,
    credentials                    [265]    SEQUENCE OF BACnetDeviceObjectReference,
    profile-name                   [168]    CharacterString OPTIONAL
    }
```

[Add new **Clause D.X**, p. 534]

D.X Example of an Access User object

In this example, the person Dorothy H. Miller is represented as an Access User object. Note that this is a fictitious person. The same access user is represented in multiple devices and its global identifier is 22457.

```
Property:    Object_Identifier =           (Access User, Instance 2)
Property:    Object_Name =                 "Miller37"
Property:    Object_Type =                 ACCESS_USER
```

Property:	Description =	"Dorothy H. Miller, CEO"
Property:	Global_Identifier =	22457
Property:	Status_Flags =	{FALSE, FALSE, FALSE, FALSE}
Property:	Reliability =	NO_FAULT_DETECTED
Property:	User_Type =	PERSON
Property:	User_Name =	"Dorothy H. Miller"
Property:	User_External_Identifier =	"SSN-575-99-1234"
Property:	User_Information_Reference =	"HRMS:D93345666"
Property:	Members =	((Access User, Instance 734), (Access User, Instance 41))
Property:	Member_Of =	((Access User, Instance 12), (Access User, Instance 1))
Property:	Credentials =	((Access Credential, Instance 12110), (Access Credential, Instance 41445))

135-2008*j*-4. Add a new Access Rights object type.

Rationale
Support for access control in BACnet requires a standardized object whose properties represent the externally visible characteristics associated with access rights for physical access control. This object is a collection of individual access rule specifications that define privileges for entering and leaving access controlled zones or for accessing other resources or functions.

Addendum 135-2008*j*-4

[Insert new Clause **12.X** and Table **12-X**, p. 288]

12.X Access Rights Object Type

The Access Rights object type defines a standardized object whose properties represent the externally visible characteristics associated with access rights for physical access control.

The Access Rights object is a collection of individual access rule specifications which define privileges for entering and leaving access controlled zones or for accessing other resources or functions. One or many credentials can share this collection of access rules. This object type supports role-based access control models.

The Access Rights object contains a collection of negative and positive access rules. A negative access rule specifies where and when access shall be denied. A positive access rule specifies where and when access may be granted. Negative access rules take precedence over positive access rules. All negative access rules of all Access Rights objects assigned to a credential are evaluated before any positive access rule.

Each access rule, whether positive or negative, specifies the location of access, which is an access controlled point or a zone, a condition which determines whether the rule applies at this time, and a flag which indicates whether the rule is enabled. In the most typical case the condition is determined by evaluating the Present_Value of a Schedule object that specifies time ranges. In addition, each of these access rules can be enabled or disabled individually.

The Access Rights object can specify an accompaniment requirement that defines the access user that owns the accompanying credential, the credential required to accompany, or the access rights required to be assigned to the accompanying credential.

When a specific access rights collection is represented in multiple devices, the representing Access Rights objects may not have the same Object_Identifier in each device; however, they may be identified using the Global_Identifier property. It is a local matter as to how these objects are synchronized.

The Access Rights object type and its properties are summarized in Table 12-X and described in detail in this subclause.

Table 12-X. Properties of the Access Rights Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Global_Identifier	Unsigned32	W
Status_Flags	BACnetStatusFlags	R
Reliability	BACnetReliability	R
Enable	BOOLEAN	R
Negative_Access_Rules	BACnetARRAY[N] of BACnetAccessRule	R
Positive_Access_Rules	BACnetARRAY[N] of BACnetAccessRule	R
Accompaniment	BACnetDeviceObjectReference	O
Profile_Name	CharacterString	O

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 ACCESS_ RIGHTS.

12.X.4 Description

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

12.X.5 Global_Identifier

This property, of type Unsigned32, is a unique identifier which is used to globally identify the collection of access rights this object represents. This value may be used to identify Access Rights objects in multiple devices which represent the same collection of access rights.

If this value is assigned, it shall be unique internetwork-wide and all Access Rights objects in all devices which represent this collection of access rights shall have this value. A value of zero indicates that no global identifier is assigned.

12.X.6 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of this object. A more detailed status may 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	The value of this flag shall be logical FALSE (0).
FAULT	Logical TRUE (1) if the Reliability is not NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	The value of this flag shall be logical FALSE (0).
OUT_OF_SERVICE	The value of this flag shall be logical FALSE (0).

12.X.7 Reliability

The Reliability property, of type BACnetReliability, provides an indication of whether this object is "reliable" as far as the BACnet Device can determine and, if not, why. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, CONFIGURATION_ERROR, UNRELIABLE_OTHER}.

12.X.8 Enable

This property, of type BOOLEAN, indicates whether this object is enabled (TRUE) or disabled (FALSE). When this object is disabled all the access rules specified in the Positive_Access_Rules and Negative_Access_Rules properties are disabled.

12.X.9 Negative_Access_Rules

This property, of type BACnetARRAY[N] of BACnetAccessRule, specifies the negative access rules. Each element of the array is evaluated as described in Clause 12.X.9.1.

To determine how access rules are evaluated, see Clause 12.X.9.2.

12.X.9.1 Access Rule Specification

An access rule specification is of type BACnetAccessRule. This is a structure with the following fields:

Time-Range-Specifier — This field is an enumeration that specifies the evaluation of the Time-Range field:

SPECIFIED	Time-Range references a property that will be evaluated to TRUE or FALSE as defined for the Time-Range field.
ALWAYS	The value of the Time-Range field is ignored and always evaluates to TRUE.

Time-Range — This optional field, of type BACnetDeviceObjectPropertyReference, references a property that can be evaluated to TRUE or FALSE, which defines whether the rule is valid (TRUE) or not (FALSE).

The Time-Range reference shall be considered *unspecified* if it contains 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present.

This field shall be present if Time-Range-Specifier is SPECIFIED. If Time-Range-Specifier is ALWAYS and this field is present, then this reference shall be unspecified.

	If Time-Range-Specifier is SPECIFIED and this field references a property of type other than BOOLEAN, then the following evaluations apply: If the value of the referenced property is of type Unsigned, a value of zero shall evaluate to FALSE, while any other value shall evaluate to TRUE. If the value of the referenced property is of type INTEGER, a value of less than or equal to zero shall evaluate to FALSE, while any value greater than zero shall evaluate to TRUE. If the value of the referenced property is of type BACnetBinaryPV, then INACTIVE shall evaluate to FALSE, while ACTIVE evaluates to TRUE. If the referenced property does not exist or is unspecified, or if its value cannot be retrieved or is of type NULL, the Time-Range evaluates to FALSE. If the reference property is of any other type, then the evaluation is a local matter. Note: This field can reference a Schedule object Present_Value property for the specification of time ranges.
Location-Specifier	This field is an enumeration that specifies how the Location field is evaluated: SPECIFIED — Location references a specific Access Point or Access Zone object and is evaluated as specified for the Location field. ALL — The value of the Location field is ignored and matches any access controlled point.
Location	This optional field, of type BACnetDeviceObjectReference, refers to the Access Point or Access Zone that this access rule is valid for. The Location reference shall be considered *unspecified* if it contains 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present. This field shall be present if Location-Specifier is SPECIFIED. If Location-Specifier is ALL and this field is present, then this reference shall be unspecified. If Location-Specifier is SPECIFIED, then the following evaluations apply: When Location refers to an Access Point object, this access controlled point is required to be the location where the credential used to request access has been authenticated. When Location refers to an Access Zone object, the access controlled point where the credential used to request access has been authenticated is required to be an entry point to this zone. If the referenced object does not exist, is unspecified, or cannot be retrieved, then the Location evaluates to FALSE.
Enable	This field, of type BOOLEAN, specifies whether this rule is enabled (TRUE) or

not (FALSE).

An access rule evaluates to true when all of the following conditions are met:

(a) Time_Range field evaluates to TRUE, and
(b) the Location field matches the location of authentication, and
(c) the Enable field is TRUE.

12.X.9.2 Access Rules Authorization Check

The access rules authorization check is performed on the access rules assigned to a credential.

All the negative access rules of all the Access Rights objects referenced by the respective Access Credential object shall be evaluated before the positive access rules. If any enabled negative rule evaluates to true, then this authorization check fails and access shall be denied. In this case, the Access_Event property of the Access Point object shall be set to DENIED_POINT_NO_ACCESS_RIGHTS if the negative rule prohibits access through the access point, or to DENIED_ZONE_NO_ACCESS_RIGHTS if the negative rule prohibits access to the zone.

If no negative access rule evaluates to true, then the positive access rules of all the Access Rights objects referenced by the respective Access Credential object shall be evaluated. When the first enabled positive access rule is found that evaluates to true, then this authorization check succeeds. Access may subsequently be denied or granted based on other authorization checks.

If all positive access rules of all the Access Rights objects referenced by this credential evaluate to false, then this authorization check shall fail. In this case, if the credential has access through this access point or to the access zone at a different time, then the Access_Event property of the Access Point object shall be set to DENIED_OUT_OF_TIME_RANGE. Otherwise, the Access_Event property shall be set to DENIED_NO_ACCESS_RIGHTS.

If the respective Access Credential object has a master exemption, then this authorization check is not performed and always considered successful.

12.X.9.3 Initializing New Array Elements When the Array Size is Increased

If the size of the Negative_Access_Rules array is increased without initial values being provided, then the new array elements, for which no initial value is provided, shall be initialized to contain SPECIFIED for the Time-Range-Specifier field, an unspecified reference in the Time-Range field, SPECIFIED for the Location-Specifier field, an unspecified reference in the Location field, and TRUE for the Enable field.

12.X.10 Positive_Access_Rules

This property, of type BACnetARRAY[N] of BACnetAccessRule, specifies the positive access rules. Each element of the array is evaluated as described in Clause 12.X.9.1.

To determine how access rules are evaluated, see Clause 12.X.9.2

12.X.10.1 Initializing New Array Elements When the Array Size is Increased

If the size of the Positive_Access_Rules array is increased without initial values being provided, then the new array elements, for which no initial value is provided, shall be initialized to contain SPECIFIED for the Time-Range-Specifier field, an unspecified reference in the Time-Range field, SPECIFIED for the Location-Specifier field, an unspecified reference in the Location field, and TRUE for the Enable field.

12.X.11 Accompaniment

This optional property, of type BACnetDeviceObjectReference, specifies that the access rights, which this object represents, may be evaluated successfully only if the original credential, which has this Access Rights object assigned, is accompanied by a second credential that meets the accompaniment criteria and is presented at the same access point. The accompanying credential must also have valid access rights to the Access Point where both credentials are presented. It is a local matter as to whether the accompanying credential is required to be presented by the access user before or after the original credential.

If the accompanying credential is not presented within the amount of time, specified by the Accompaniment_Time property of the Access Point object, then the authorization of the original credential will fail. If this time is not specified then the amount of time to wait for the accompanying credential is a local matter. When the expected accompaniment is not received, the Access_Event property of the Access Point object shall be set to DENIED_NO_ACCOMPANIMENT.

When an accompaniment is presented, whether valid or not, the Access_Event property of the Access Point object shall be set to ACCOMPANIMENT_BY.

The accompaniment criteria are specified as:

(a) If this property refers to an Access Rights object, then the accompanying credential is required to have that Access Rights object assigned.
(b) If this property refers to an Access Credential object, then this object is required to represent the accompanying credential.
(c) If this property refers to an Access User object, then this object is required to represent the access user which owns the accompanying credential.

If an invalid accompaniment is provided, then the Access_Event property of the Access Point object shall be set to DENIED_INCORRECT_ACCOMPANIMENT.

If no accompaniment requirement is specified then this reference shall contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present.

12.X.12 Profile_Name

This optional property, of type CharacterString, is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name begins with a vendor identifier code (see Clause 23) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified here. This standard defines only the format of the names of profiles. The definition of the profiles themselves is outside the scope of this standard.

[Add new **BACnetAccessRule** production to Clause **21**, p. 448]

```
BACnetAccessRule ::= SEQUENCE {
    timeRangeSpecifier   [0]  ENUMERATED {
                                  specified        (0),
                                  always           (1)
                                  },
    timeRange            [1]  BACnetDeviceObjectPropertyReference OPTIONAL, -- to be present if
                                                                            -- timeRangeSpecifier has the
                                                                            -- value "specified"
    locationSpecifier    [2]  ENUMERATED {
                                  specified        (0),
                                  all              (1)
                                  },
    location             [3]  BACnetDeviceObjectReference OPTIONAL,     -- to be present if locationSpecifier has
```

```
                                                                   -- the value "specified"
    enable                  [4]  BOOLEAN
    }
```

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetObjectTypesSuppported** BITSTRING appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2008*j*-1.]

[Add new object type structure to **ANNEX C**, p. 497]

```
ACCESS-RIGHTS ::= SEQUENCE {
    object-identifier           [75]    BACnetObjectIdentifier,
    object-name                 [77]    CharacterString,
    object-type                 [79]    BACnetObjectType,
    description                 [28]    CharacterString OPTIONAL,
    global-identifier           [323]   Unsigned32,
    status-flags                [111]   BACnetStatusFlags,
    reliability                 [103]   BACnetReliability,
    enable                      [133]   BOOLEAN,
    negative-access-rules       [288]   SEQUENCE OF BACnetAccessRule,   -- accessed as a BACnetARRAY
    positive-access-rules       [302]   SEQUENCE OF BACnetAccessRule,   -- accessed as a BACnetARRAY
    accompaniment               [252]   BACnetDeviceObjectReference OPTIONAL,
    profile-name                [168]   CharacterString OPTIONAL
    }
```

[Add new **Clause D.X**, p. 534]

D.X Examples of Access Rights objects

Assumed objects:

Object_Identifier	Object_Name	Object_Type
(Device 12, Access Point, Instance 1)	Main Production Area Door	ACCESS_POINT
(Device 14, Access Point, Instance 7)	North Emergency Exit Door	ACCESS_POINT
(Access Zone, Instance 23)	Perimeter Doors	ACCESS_ZONE
(Schedule, Instance 44)	Night Shift Hours	SCHEDULE
(Access Credential, Instance 33)	Bob Smith	ACCESS_CREDENTIAL

In this first example, this Access Rights object defines the access rights of night shift workers of the facility. The rules state that credentials having this access right may access any of the perimeter doors (Access Zone, Instance 23) at any time with the exception of the north emergency exit door (Device 12, Access Point, Instance 7), for which access is never allowed. In addition, the main production area door (Device 14, Access Point, Instance 1) may be accessed during times when the night shift hours schedule (Schedule, Instance 44) is active. This access rights collection is represented only in a single device and therefore its global identifier is zero.

Property:	Object_Identifier =	(Access Rights, Instance 2)
Property:	Object_Name =	"Night Shift Rights"
Property:	Object_Type =	ACCESS_RIGHTS
Property:	Description =	"Access Rights for main production area at night shift"
Property:	Global_Identifier	0
Property:	Status_Flags =	{FALSE, FALSE, FALSE, FALSE}
Property:	Reliability =	NO_FAULT_DETECTED
Property:	Enable =	TRUE

Property:	Negative_Access_Rules =	((ALWAYS, ((*, Instance 4194303), *,,), SPECIFIED, ((Device, Instance 12), (Access Point, Instance 7)), TRUE))
Property:	Positive_Access_Rules =	((SPECIFIED, ((Schedule, Instance 44), Present_Value,,), SPECIFIED, ((Device, Instance 14), (Access Point, Instance 1)), TRUE), (ALWAYS, ((*, Instance 4194303), *,,), SPECIFIED, (,(Access Zone, Instance 23)), TRUE))

In this second example, this Access Rights object defines the access rules for visitors to this facility. The rules state that a visitor may have access to any door in the facility but that they must always be accompanied by Bob Smith (Access Credential, Instance 33).

Property:	Object_Identifier =	(Access Rights, Instance 9)
Property:	Object_Name =	"Visitor Access Rights"
Property:	Object_Type =	ACCESS_RIGHTS
Property:	Description =	"Access rights for accompanied visitors"
Property:	Enable =	TRUE
Property:	Negative_Access_Rules =	()
Property:	Positive_Access_Rules =	((ALWAYS, ((*, Instance 4194303), *,,), ALL, (,(*, Instance 4194303)), TRUE))
Property:	Accompaniment =	(,(Access Credential, Instance 33))

135-2008*j*-5. Add a new Access Credential object type.

Rationale
Support for access control in BACnet requires a standardized object whose properties represent the externally visible characteristics of a credential that is used for authentication and authorization when requesting access. The credential can be owned by an access user of any type; ownership is represented by a reference to the Access User object that represents the owning access user.

Addendum 135-2008*j*-5

[Insert new Clause **12.X**, p. 288]

12.X Access Credential Object Type

The Access Credential object type defines a standardized object whose properties represent the externally visible characteristics of a credential that is used for authentication and authorization when requesting access.

The credential can be owned by an access user of any type. Access user ownership is represented by a reference to an Access User object.

The Access Credential object is a container of related authentication factors. Each authentication factor in the credential can be individually enabled or disabled. An Access Credential object can represent a single authentication factor, a group of authentication factors each having identical access rights, or multiple authentication factors required for multi-factor-authentications.

The access rights assigned to the credential are specified by referencing Access Rights objects. Each reference can be individually enabled or disabled.

The Credential_Status indicates the validity of this credential for authentication. The status is derived from other properties of this object or can be set from an external process.

The credential can be restricted in its use for authentication. It can be restricted based on activation and expiry dates, the number of days it can be used or the number of uses. It can be disabled if it is not used for a specified number of days. The credential can be exempted from authorization checks such as passback violation enforcement and occupancy enforcements. It can indicate whether an extended time is required to pass through a door.

A threat authority can be specified for the credential. If this value is lower than the threat level at the access controlled point, then access is denied.

The credential can be flagged to be traced. Any access controlled point recognizing this credential shall generate a corresponding TRACE access event.

When a credential is represented in multiple devices, the representing Access Credential objects may not have the same Object_Identifier in each device; however, they may be identified using the Global_Identifier property. It is a local matter as to how these objects are synchronized.

The Access Credential object type and its properties are summarized in Table 12-X and described in detail in this subclause.

Table 12-X. Properties of the Access Credential Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Global_Identifier	Unsigned32	W
Status_Flags	BACnetStatusFlags	R
Reliability	BACnetReliability	R
Credential_Status	BACnetBinaryPV	R
Reason_For_Disable	List of BACnetAccessCredentialDisableReason	R
Authentication_Factors	BACnetARRAY[N] of BACnetCredentialAuthenticationFactor	R
Activation_Time	BACnetDateTime	R
Expiry_Time	BACnetDateTime	R
Credential_Disable	BACnetAccessCredentialDisable	R
Days_Remaining	INTEGER	O[1]
Uses_Remaining	INTEGER	O
Absentee_Limit	Unsigned	O[1]
Belongs_To	BACnetDeviceObjectReference	O
Assigned_Access_Rights	BACnetARRAY[N] of BACnetAssignedAccessRights	R
Last_Access_Point	BACnetDeviceObjectReference	O
Last_Access_Event	BACnetAccessEvent	O
Last_Use_Time	BACnetDateTime	O
Trace_Flag	BOOLEAN	O
Threat_Authority	BACnetAccessThreatLevel	O
Extended_Time_Enable	BOOLEAN	O
Master_Exemption	BOOLEAN	O
Passback_Exemption	BOOLEAN	O
Occupancy_Exemption	BOOLEAN	O
Profile_Name	CharacterString	O

[1] If this property is present, then the property Last_Use_Time shall also 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 ACCESS_CREDENTIAL.

12.X.4 Description

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

12.X.5 Global_Identifier

This property, of type Unsigned32, is a unique identifier which is used to globally identify the credential this object represents. This value may be used to identify Access Credential objects in multiple devices which represent the same credential.

If this value is assigned, it shall be unique internetwork-wide and all Access Credential objects in all devices which represent this credential shall have this value. A value of zero indicates that no global identifier is assigned.

12.X.6 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of this object. A more detailed status may 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	The value of this flag shall be logical FALSE (0).
FAULT	Logical TRUE (1) if the Reliability is not NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	The value of this flag shall be logical FALSE (0).
OUT_OF_SERVICE	The value of this flag shall be logical FALSE (0).

12.X.7 Reliability

The Reliability property, of type BACnetReliability, provides an indication of whether this object is "reliable" as far as the BACnet Device can determine and, if not, why. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, CONFIGURATION_ERROR, UNRELIABLE_OTHER}.

12.X.8 Credential_Status

This property, of type BACnetBinaryPV, specifies whether the credential is active or inactive. Only the value ACTIVE enables the credential to be used for authentication. While the list in property Reason_For_Disable is nonempty, the status of the credential shall be INACTIVE, otherwise it shall be ACTIVE.

When an inactive credential is used, the authentication of this credential shall fail and access to the access point shall be denied. In this case, the Access_Event property of the Access Point object where the credential has attempted access shall be set to the value which corresponds to the reason this credential is disabled, as specified in the Reason_For_Disable property. See Clause 12.X.9.1.

12.X.9 Reason_For_Disable

This property, of type List of BACnetAccessCredentialDisableReason, contains a list of disable-reasons why the credential has been disabled. The credential can be disabled for multiple reasons at the same time. While the Credential_Status property has a value INACTIVE, this list shall not be empty. When an entry is removed from this list that results in the list becoming empty, the Credential_Status shall be set to ACTIVE.

The disable-reasons for which the credential can be disabled are as follows:

DISABLED	The credential is disabled for unspecified reasons.
DISABLED_NEEDS_PROVISIONING	The credential needs further provisioning, which may include vendor proprietary data.
DISABLED_UNASSIGNED	The credential is not currently assigned to any access user. This status is assigned only if the property Belongs_To is present and contains instance 4194303 in the object identifier.
DISABLED_NOT_YET_ACTIVE	The credential is not yet valid at this time. The current time is before the Activation_Time.
DISABLED_EXPIRED	The credential is no longer valid. The current time is after the Expiry_Time.
DISABLED_LOCKOUT	Too many retries in multi-factor authentications have been performed.
DISABLED_MAX_DAYS	The maximum number of days for which this credential is valid for has been exceeded.
DISABLED_MAX_USES	The maximum number of uses for which this credential is valid for has been exceeded.
DISABLED_INACTIVITY	The credential has exceeded the allowed period of inactivity.
DISABLED_MANUAL	The credential is commanded to be disabled by a human operator.
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values to indicate disable reasons other than those defined by this standard. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

12.X.9.1 Conditions for setting the Access_Event property of the Access Point object

When access is requested using a credential that is inactive, access shall not be granted. In this case the Access Point object representing the access point where access was requested shall set its Access_Event property as defined in Table 12-X+1:

Table 12-X+1. Credential Disable Reasons and Applicable Access Events

Credential Disable Reason	Applicable Access Event
DISABLED_NEEDS_PROVISIONING	DENIED_CREDENTIAL_NOT_PROVISONED
DISABLED_UNASSIGNED	DENIED_CREDENTIAL_UNASSIGNED
DISABLED_NOT_YET_ACTIVE	DENIED_CREDENTIAL_NOT_YET_ACTIVE
DISABLED_LOCKOUT	DENIED_CREDENTIAL_LOCKOUT
DISABLED_MAX_DAYS	DENIED_CREDENTIAL_MAX_DAYS
DISABLED_MAX_USES	DENIED_CREDENTIAL_MAX_USES
DISABLED_INACTIVITY	DENIED_CREDENTIAL_INACTIVITY
DISABLED_MANUAL	DENIED_CREDENTIAL_MANUAL_DISABLE
DISABLED	DENIED_CREDENTIAL_DISABLED

If Reason_For_Disable contains multiple values, it is a local matter as to which corresponding access event the

Access_Event property is set to.

12.X.10 Authentication_Factors

This property, of type BACnetARRAY[N] of BACnetCredentialAuthenticationFactor, specifies the authentication factors that belong to this credential. Each element of the array has two fields:

Disable	This field, of type BACnetAccessAuthenticationFactorDisable, specifies whether the corresponding authentication factor is disabled or not. Any value other than NONE indicates that the authentication factor is not valid for authentication.

The following authentication factor disable values are defined:

DISABLED	The physical authentication factor is disabled for unspecified reasons.
DISABLED_LOST	The physical authentication factor is reported to be lost.
DISABLED_STOLEN	The physical authentication factor is reported to be stolen.
DISABLED_DAMAGED	The physical authentication factor is reported to be damaged.
DISABLED_DESTROYED	The physical authentication factor is reported to be destroyed.
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values to specify disable values other than those defined by this standard. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

Authentication-Factor	This field, of type BACnetAuthenticationFactor, specifies the authentication factor that belongs to this credential.

Any access attempt using an authentication factor which is disabled shall fail. In this case, the Access_Event property of the Access Point object where this authentication factor was used shall be set to the value corresponding to the reason why it was disabled. See Table 12-X+2.

Table 12-X+2. Authentication Factor Disable and Applicable Access Events

Authentication Factor Disable	Applicable Access Event
DISABLED	DENIED_AUTHENTICATION_FACTOR_DISABLED
DISABLED_LOST	DENIED_AUTHENTICATION_FACTOR_LOST
DISABLED_STOLEN	DENIED_AUTHENTICATION_FACTOR_STOLEN
DISABLED_DAMAGED	DENIED_AUTHENTICATION_FACTOR_DAMAGED
DISABLED_DESTROYED	DENIED_AUTHENTICATION_FACTOR_DESTROYED

12.X.10.1 Initializing New Array Elements When the Array Size is Increased

If the size of the Authentication _Factors array is increased without initial values being provided, then the new array elements for which no initial value is provided shall be initialized to contain DISABLE for the Disable field and an authentication factor with format type UNDEFINED for the Authentication-Factor field.

12.X.11 Activation_Time

This property, of type BACnetDateTime, specifies the date and time at or after which the credential becomes active. If the current time is before the activation time, the credential shall be disabled and the value DISABLED_NOT_YET_ACTIVE shall be added to the Reason_For_Disable list. The value DISABLED_NOT_YET_ACTIVE shall be removed from the list when this condition no longer applies. If all of the octets of the BACnetDateTime value contain a value of X'FF', then the credential has an activation time of 'start of time'.

12.X.12 Expiry_Time

This property, of type BACnetDateTime, specifies the date and time after which the credential will expire. This defines the end of the validity period of the credential. If the current time is after the expiry time, the credential shall be disabled and the value DISABLED_EXPIRED shall be added to the Reason_For_Disable list. The value DISABLED_EXPIRED shall be removed from the list when this condition no longer applies. If all of the fields of the BACnetDateTime value contain a value of X'FF', then the credential has an expiry time of 'end-of-time'.

12.X.13 Credential_Disable

This property, of type BACnetAccessCredentialDisable, contains a value that disables a credential for reasons external to this object. If this property is writable, then it is the mechanism by which an operator or external process may disable the credential.

When this property is changed, any disable reason added to the Reason_For_Disable list as a result of a previous change of this property shall be removed from that list. When this property takes on any value other than NONE, the corresponding disable-reason value shall be added to the Reason_For_Disable list.

The following credential disable values are defined:

NONE	The credential has not been disabled by an operator or external process.
DISABLE	The credential has been disabled for unspecified reasons. The disable-reason value DISABLED shall be added to the Reason_For_Disable property.
DISABLE_MANUAL	The credential has been disabled by a human operator. The disable-reason value DISABLED_MANUAL shall be added to the Reason_For_Disable property.
DISABLE_LOCKOUT	The credential is disabled because it has been locked out by an external process. The disable-reason value DISABLED_LOCKOUT shall be added to the Reason_For_Disable property.
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values for disabling a credential other than those defined by this standard. A disable-reason value shall be added to the Reason_For_Disable property. It is a local matter which disable reason is added. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

12.X.14 Days_Remaining

This optional property, of type INTEGER, specifies the number of remaining days for which the credential can be used. If this property has a value greater than zero, its value shall be decremented by one when the credential this object represents is granted access at an access controlled point, and the current date is more recent than the date indicated in the property Last_Use_Time. If this property becomes zero, the Access Credential shall be disabled and the value

DISABLED_MAX_DAYS shall be added to the Reason_For_Disable property. The value DISABLED_MAX_DAYS shall be removed from the Reason_For_Disable property when this property is set to a value greater than zero.

If this property is present and the credential this object represents is not limited in the days it can be used, then the value of this property shall be -1 and DISABLED_MAX_USES shall never be added to the Reason_For_Disable property.

If Days_Remaining is present, then Last_Use_Time shall also be present.

12.X.15 Uses_Remaining

This optional property, of type INTEGER, specifies the number of remaining uses that the credential can be used for authentication. If this property has a value greater than zero and access is granted at an access controlled point, then the value of this property shall be decremented by one. If this property becomes zero, then the Access Credential shall be disabled and the value DISABLED_MAX_USES shall be added to the Reason_For_Disable property. The value DISABLED_MAX_USES shall be removed from the Reason_For_Disable property when this property is set to a value greater than zero.

If this property is present and the credential this object represents is not limited in the number of uses, then the value of this property shall be -1 and DISABLED_MAX_USES shall never be added to the Reason_For_Disable property.

12.X.16 Absentee_Limit

This optional property, of type Unsigned, specifies the maximum number of consecutive days for which the credential can remain inactive (i.e. unused) before it becomes disabled. The calculation of inactivity duration is based on the time of last use as indicated by the property Last_Use_Time. If Last_Use_Time does not have a valid time and date, then the absentee limit shall be considered to not be exceeded. When the absentee limit is exceeded, the Access Credential shall be disabled and the value DISABLED_INACTIVITY shall be added to the Reason_For_Disable list. The value DISABLED_INACTIVITY shall be removed from the list when this condition no longer applies.

If Absentee_Limit is present, Last_Use_Time shall also be present.

12.X.17 Belongs_To

This optional property, of type BACnetDeviceObjectReference, references an Access User object that represents the owning access user (i.e., person, group, or asset). If this property is present and the credential is not assigned to an access user, this property shall contain an instance number of 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present. The determination of whether the credential is valid for authentication, based on the value of this property, is a local matter. If the credential has not been assigned to an access user and the policy of the site requires that it be assigned, then the credential shall be disabled and the value DISABLED_UNASSIGNED shall be added to the Reason_For_Disable list. The value DISABLED_UNASSIGNED shall be removed from the list when this condition no longer applies.

12.X.18 Assigned_Access_Rights

This property, of type BACnetARRAY [N] of BACnetAssignedAccessRights, specifies the access rights assigned to this credential. The structure has two fields:

Assigned-Access-Rights	This field, of type BACnetDeviceObjectReference, refers to an Access Rights object that defines access rights assigned to this credential. Each object referenced in this field shall be an Access Rights object. Any entry which references a non-existent Access Rights object shall be ignored. If no access rights are specified, then this reference shall contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present.
Enable	This field, of type BOOLEAN, specifies whether the access rights specified in the assigned-access-rights field is enabled (TRUE) or not (FALSE) for the credential this

object represents.

12.X.18.1 Initializing New Array Elements When the Array Size is Increased

If the size of the Assigned_Access_Rights array is increased without initial values being provided, then the new array elements for which no initial value is provided shall be initialized to contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present, for the Assigned-Access-Rights field, and the value FALSE for the Enable field.

12.X.19 Last_Access_Point

This optional property, of type BACnetDeviceObjectReference, refers to the last Access Point object where one of the authentication factors of the credential has been used. If property level COV is in effect for this property, any update of this property shall cause a COV notification to be issued, regardless of whether the value of this property changes. If the credential this object represents has never been used, then this property shall contain 4194303 in the instance part of the object identifier and in the device instance part of the device identifier, if present.

12.X.20 Last_Access_Event

This optional property, of type BACnetAccessEvent, shall specify the last access event generated at an access controlled point on use of this credential. If the credential this object represents has never been used, then this property shall have a value of NONE.

12.X.21 Last_Use_Time

This optional property, of type BACnetDateTime, specifies the date and time of the last use of the credential at an access controlled point, independent of whether access was granted or denied. If the credential this object represents has never been used, then this property shall have the value X'FF' for all date and time octets.

12.X.22 Trace_Flag

This optional property, of type BOOLEAN, specifies whether the credential is being traced. When a traced credential is used at an access point, the Access_Event property of the corresponding Access Point object shall be set to TRACE.

12.X.23 Threat_Authority

This optional property, of type BACnetAccessThreatLevel, specifies the maximum threat level for which this credential is valid. If this value is less than the Threat_Level property of the Access Point object where the access credential is used, access is denied. If this property is not present, the threat authority of this credential is assumed to be zero.

12.X.24 Extended_Time_Enable

This optional property, of type BOOLEAN, specifies which command of type BACnetDoorValue shall be used to command the access door when access is granted. If extended time is enabled (TRUE), EXTENDED_PULSE_UNLOCK is used, otherwise (FALSE) PULSE_UNLOCK is used.

12.X.25 Master_Exemption

This optional property, of type BOOLEAN, specifies a master exemption from authorization checks. Once authenticated, the credential, if active, is exempted from all standard authorization checks if master exemption is enabled (TRUE). It is a local matter as to whether the credential is exempted from proprietary authorization checks.

12.X.26 Passback_Exemption

This optional property, of type BOOLEAN, specifies an exemption from passback enforcement. If passback exemption is enabled (TRUE), then the credential is not denied access due to passback violations.

12.X.27 Occupancy_Exemption

This optional property, of type BOOLEAN, specifies an exemption from occupancy enforcement. If occupancy exemption is enabled (TRUE), then the occupancy count in the Access Zone object shall be updated as normal; however, the access credential shall not be denied access due to occupancy limit enforcement.

12.X.28 Profile_Name

This optional property, of type CharacterString, is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name begins with a vendor identifier code (see Clause 23) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified here. This standard defines only the format of the names of profiles. The definition of the profiles themselves is outside the scope of this standard.

[Add new **BACnetAssignedAccessRights** production to Clause **21**, p. 448]

```
BACnetAssignedAccessRights ::= SEQUENCE {
    assigned-access-rights    [0] BACnetDeviceObjectReference,
    enable                    [1] BOOLEAN
    }
```

[Add new **BACnetAccessAuthenticationFactorDisable** enumeration to Clause **21**, p. 448]

```
BACnetAccessAuthenticationFactorDisable ::= ENUMERATED {
    none                  (0),
    disabled              (1),
    disabled-lost         (2),
    disabled-stolen       (3),
    disabled-damaged      (4),
    disabled-destroyed    (5),
    …
    }
    -- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
    -- 64-65535 may be used by others subject to the procedures and constraints described
    -- in Clause 23.
```

[Add new **BACnetAccessCredentialDisable** enumeration to Clause **21**, p. 448]

```
BACnetAccessCredentialDisable ::= ENUMERATED {
    none                  (0),
    disable               (1),
    disable-manual        (2),
    disable-lockout       (3),
    …
    }
    -- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
    -- 64-65535 may be used by others subject to the procedures and constraints described
    -- in Clause 23.
```

[Add new **BACnetAccessCredentialDisableReason** enumeration to Clause **21**, p. 448]

```
BACnetAccessCredentialDisableReason ::= ENUMERATED {
    disabled                        (0),
    disabled-needs-provisioning     (1),
    disabled-unassigned             (2),
    disabled-not-yet-active         (3),
    disabled-expired                (4),
    disabled-lockout                (5),
    disabled-max-days               (6),
    disabled-max-uses               (7),
    disabled-inactivity             (8),
    disabled-manual                 (9),
    …
    }
    -- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
    -- 64-65535 may be used by others subject to the procedures and constraints described
    -- in Clause 23.
```

[Note: **BACnetAccessThreatLeve**l is defined in Addendum 135-2008*j*-1.]

[Note: **BACnetAuthenticationFactor** is defined in Addendum 135-2008*j*-6.]

[Note: **BACnetAccessEvent** is defined in Addendum 135-2008*j*-1.]

[Add new **BACnetCredentialAuthenticationFactor** production to Clause **21**, p. 448]

```
BACnetCredentialAuthenticationFactor ::= SEQUENCE {
    disable                     [0] BACnetAccessAuthenticationFactorDisable,
    authentication-factor       [1] BACnetAuthenticationFactor
    }
```

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetObjectTypesSuppported** BITSTRING appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to **Clause 21**, **BACnetPropertyStates** production appear in Addendum 135-2008*j*-1]

[Note: changes to **Table 23-1 Extensible Enumerations** appear in Addendum 135-2008*j*-1.]

[Add new object type structure to **ANNEX C**, p. 497]

```
ACCESS-CREDENTIAL ::= SEQUENCE {
    object-identifier           [75]    BACnetObjectIdentifier,
    object-name                 [77]    CharacterString,
    object-type                 [79]    BACnetObjectType,
    description                 [28]    CharacterString OPTIONAL,
    global-identifier           [323]   Unsigned32,
    status-flags                [111]   BACnetStatusFlags,
    reliability                 [103]   BACnetReliability,
    credential-status           [264]   BACnetBinaryPV,
    reason-for-disable          [303]   SEQUENCE OF BACnetAccessCredentialDisableReason,
    authentication-factors      [257]   SEQUENCE OF BACnetCredentialAuthenticationFactor,
                                        -- accessed as a BACnetARRAY
    activation-time             [254]   BACnetDateTime,
    expiry-time                 [270]   BACnetDateTime ,
    credential-disable          [263]   BACnetAccessCredentialDisable,
    days-remaining              [267]   INTEGER OPTIONAL,
    uses-remaining              [319]   INTEGER OPTIONAL,
    absentee-limit              [244]   Unsigned OPTIONAL,
    belongs-to                  [262]   BACnetDeviceObjectReference OPTIONAL,
    assigned-access-rights      [256]   SEQUENCE OF BACnetAssignedAccessRights,
                                        -- accessed as a BACnetARRAY
    last-access-point           [276]   BACnetDeviceObjectReference OPTIONAL,
    last-access-event           [275]   BACnetAccessEvent OPTIONAL,
    last-use-time               [281]   BACnetDateTime OPTIONAL,
    trace-flag                  [308]   BOOLEAN OPTIONAL,
    threat-authority            [306]   BACnetAccessThreatLevel OPTIONAL,
    extended-time-enable        [271]   BOOLEAN OPTIONAL,
    master-exemption            [284]   BOOLEAN OPTIONAL,
    passback-exemption          [299]   BOOLEAN OPTIONAL,
    occupancy-exemption         [293]   BOOLEAN OPTIONAL,
    profile-name                [168]   CharacterString OPTIONAL
    }
```

[Add new **Clause D.X**, p. 534]

D.X Example of an Access Credential object

This example of an Access Credential object represents the authentication factors assigned to a specific user who has the identical access rights. The credential is active and has two authentication factors assigned to it:

1. Wiegand proximity card with facility code =131 (X'83') and card number 77 (X'004D'). This authentication factor has been disabled because it was lost.
2. A FASC-N smart card with agency code = 9700 (X'25E4'), system site code = 1234 (X'04D2') and credential number = 123456 (X'0001E240').

The credential is valid from 1-MAR-2008 09:00 AM until 30-MAR-2009 11:59 PM and is valid for 30 uses. If the card is not used for 14 consecutive days, it will automatically become disabled.

The credential belongs to an Access User Instance 2 and has access rights, as specified by the local Access Rights object instance 1 and 77 assigned to it. The credential also has the local Access Rights object instance 219 assigned to it, but it has been disabled for this credential.

The credential has the extended access time flag enabled, but neither master exemption, passback exemption nor occupancy count exemption is set for this credential. The same credential is represented in multiple devices and its global identifier is 805281245.

Property:	Object_Identifier =	(Access Credential, Instance 33)
Property:	Object_Name =	"Manager Credential 33"
Property:	Object_Type =	ACCESS_CREDENTIAL
Property:	Description =	"Credential 1 Miller37"
Property:	Global_Identifier	805281245
Property:	Status_Flags =	{FALSE, FALSE, FALSE, FALSE}
Property:	Reliability =	NO_FAULT_DETECTED
Property:	Credential_Status =	ACTIVE
Property:	Reason_For_Disable =	( )
Property:	Authentication_Factors =	((DISABLE_LOST,(WIEGAND26, 0, X'83004D')), (NONE, (FASC_N, 0, X'25E404D20001E240')))
Property:	Activation_Time =	((01 MAR 2008, SATURDAY), 09:00:00.0))
Property:	Expiry_Time =	((30 MAR 2009, MONDAY), 23:59:59.9))
Property:	Credential_Disable =	NONE
Property:	Days_Remaining =	-1
Property:	Uses_Remaining =	30
Property:	Absentee_Limit =	14
Property:	Belongs_To =	(Access User, Instance 2)
Property:	Assigned_Access_Rights =	(((,(Access Rights, Instance 1)), TRUE), ((,(Access Rights, Instance 77)), TRUE), ((,(Access Rights, Instance 219)), FALSE))
Property:	Last_Access_Point =	(,(Access Point, Instance 2))
Property:	Last_Access_Event =	PASSBACK_DETECTED
Property:	Last_Use_Time =	((8 APRIL 2008, TUESDAY), 11:11:33.2))
Property:	Trace_Flag =	FALSE
Property:	Threat_Authority =	50
Property:	Extended_Time_Enable =	TRUE
Property:	Master_Exemption =	FALSE
Property:	Passback_Exemption =	FALSE
Property:	Occupancy_ Exemption =	FALSE

135-2008*j*-6. Add a new Credential Data Input object type.

Rationale

Support for access control in BACnet requires a standardized object whose properties represent the externally visible characteristics of credential data input devices, such as card readers, keypads, biometric readers, etc. The value of this object is an authentication factor that is a unique digital identifier used to verify the identity of a credential.

Addendum 135-2008*j*-6

[Add new Clause **12.X**, p 288]

12.X Credential Data Input Object Type

The Credential Data Input object type defines a standardized object whose properties represent the externally visible characteristics of a process that provides authentication factors read by a physical device. An authentication factor is a data element of a credential that is a unique digital identifier used to verify the identity of a credential. A credential may have multiple authentication factors.

Examples of physical devices that may be represented by this object type are card readers, keypads, biometric readers, etc.

A single physical credential reader which supports multiple authentication factor formats may be represented by multiple Credential Data Input objects when the authentication factor formats are not functionally equivalent or cannot be used interchangeably. An example of a device of this type is a credential reader that contains both a card and biometric reader. In this case two specific Credential Data Input objects are used; one for the card reader function and one for the biometric reader function respectively.

Alternatively, a single physical credential reader that supports multiple authentication factor formats may be represented by a single Credential Data Input object when the authentication factor formats are functionally equivalent and may be used interchangeably. An example of a device of this type is a credential reader that can read multiple Wiegand formats. It is recommended that a single Credential Data Input object that supports multiple authentication factor formats be associated with a single physical device.

The Credential Data Input object type and its properties are summarized in Table 12-X and described in detail in this subclause.

Table 12-X. Properties of the Credential Data Input Object

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	BACnetAuthenticationFactor	R[1]
Description	CharacterString	O
Status_Flags	BACnetStatusFlags	R
Reliability	BACnetReliability	R[1]
Out_Of_Service	BOOLEAN	R
Supported_Formats	BACnetARRAY[N] of BACnetAuthenticationFactorFormat	R
Supported_Format_Classes	BACnetARRAY[N] of Unsigned	O[2]
Update_Time	BACnetTimeStamp	R
Profile_Name	CharacterString	O

[1] This property is required to be writable when Out_Of_Service is TRUE.
[2] The size of this array shall be the same as the size of the Supported_Formats array.

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 CREDENTIAL_DATA_INPUT.

12.X.4 Present_Value

This property, of type BACnetAuthenticationFactor, is a structure that encapsulates the authentication factor value. The structure has three fields, which are defined as follows:

Format-Type	This field, of type BACnetAuthenticationFactorType, specifies the format of the authentication factor value in the Value field. The value of this field shall be one of the format types specified in the Supported_Formats property. If there is no current authentication factor value read by this object, then this field shall take on the value UNDEFINED. In addition, if this field contains a value that is not specified in the Supported_Formats property, such as after a modification to the Supported_Formats property or after the Out_Of_Service property changes from TRUE to FALSE, then this field shall take on the value UNDEFINED. If an authentication factor is read that contains errors or that cannot be interpreted as one of the specified format types, then this field shall take on the value ERROR.
Format-Class	This field, of type Unsigned, shall contain the value specified in the Supported_Format_Classes array field that corresponds to the authentication format type in the Format-Type field. If the Supported_Format_Classes property is not present, this field shall always have a value of zero. If Format-Type has a value of UNDEFINED, then this field shall have a value of zero.
Value	This field, of type OCTET STRING, holds the authentication factor value data. The encoding of this value is specified in the Format-Type field and defined in Table P-1 of ANNEX P.

The Present_Value property shall be writable when Out_Of_Service is TRUE.

12.X.5 Description

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

12.X.6 Status_Flags

This property, of type BACnetStatusFlags, represents four Boolean flags that indicate the general "health" of the Credential Data Input 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	The value of this flag shall be logical FALSE (0).
FAULT	Logical TRUE (1) if the Reliability is not NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	The value of this flag shall be logical FALSE (0).
OUT_OF_SERVICE	Logical TRUE (1) if the Out_Of_Service property has a value of TRUE, otherwise logical FALSE (0).

12.X.7 Reliability

The Reliability property, of type BACnetReliability, provides an indication of whether the Present_Value is "reliable" as far as the BACnet Device or operator can determine. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, CONFIGURATION_ERROR, OPEN_LOOP, SHORTED_LOOP, PROCESS_ERROR, COMMUNICATION_FAILURE, UNRELIABLE_OTHER}.

The Reliability property shall be writable when Out_Of_Service is TRUE.

12.X.8 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the Present_Value of the Credential Data Input object is prevented from being modified by some process local to the BACnet device in which the object resides.

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 in the input.

12.X.9 Supported_Formats

This property, of type BACnetARRAY of BACnetAuthenticationFactorFormat, is used to specify which authentication factor formats are supported by this object. The structure of an element of this array has three fields that are defined as follows:

Format-Type	This field, of type BACnetAuthenticationFactorType, specifies a supported authentication factor format type.
Vendor-ID	This optional field, of type Unsigned16, is required when Format-Type field has a value of CUSTOM. It shall contain the BACnet vendor identifier of the vendor which defined the custom format. This value may differ from the Vendor_Identifier property value in the Device object, which identifies the device manufacturer. If the Format-Type field does not have a value of CUSTOM and this field is present it shall

	have a value of zero.
Vendor-Format	This optional field of type Unsigned16 is required when Format-Type field has a value of CUSTOM. It shall contain a unique identifier that identifies a specific custom authentication factor format as defined by the BACnet vendor in the Vendor-ID field. If the Format-Type field does not have a value of CUSTOM and this field is present, then it shall have a value of zero.

12.X.9.1 Resizing Supported_Formats Array and Supported_Format_Classes Array, by Writing Any of these Properties

The size of the Supported Formats array and Supported_Format_Classes arrays shall be maintained so that both have the same size. If either of these arrays is present and writable and the number of elements of one is reduced, then each of the arrays shall be truncated to the new reduced size. If either of these arrays is present and writable and the number of elements of one array is increased, then the other array shall be increased to the new expanded size and the new array elements initialized according to the requirements of each property. See Clauses 12.X.9.2 and 12.X.10.2.

12.X.9.2 Initializing New Array Elements When the Array Size is Increased

If the size of the Supported_Formats array is increased without entry values being provided, then the new array entries shall be initialized with the Format-Type having a value of UNDEFINED. If Vendor-ID and Vendor-Format are present, they shall be initialized with a value of zero.

12.X.10 Supported_Format_Classes

This optional property, of type BACnetARRAY of Unsigned, specifies the values that the Format-Class field of the Present_Value may take on. The value of the ith element of this array shall be used when an authentication factor is read that is of the format defined in the ith element of the Supported_Formats array.

This property is used to distinguish between multiple different supported authentication factor formats, used on a site, of which two or more use the same authentication factor format type and may have colliding value ranges. A value of zero is used as the default where no differentiation is required. Otherwise, the value is site specific and can be any non-zero value.

12.X.10.1 Resizing Supported_Formats Array and Supported_Format_Classes Array by Writing Any of these Properties

See Clause 12.X.9.1

12.X.10.2 Initializing New Array Elements When the Array Size is Increased

If the size of the Supported_Format_Classes array is increased without entry values being provided, then the new array entries shall be initialized with a value of zero.

12.X.11 Update_Time

This property, of type BACnetTimeStamp, indicates the most recent update time when the Present_Value was updated. This property shall update its value on each update of the Present_Value. If no update has yet occurred, update times of type Time or Date shall have X'FF' in each octet, and Sequence number update times shall have the value 0.

12.X.12 Profile_Name

This property, of type CharacterString, is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name shall begin with a vendor identifier code (see Clause 23) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified here. This standard defines only the format of the names of profiles. The definition of the profiles themselves is outside the scope of this standard.

[Change Table **13-1**, p. 254]

Table 13-1. Standardized Objects That May Support COV Reporting

Object Type	Criteria	Properties Reported
…	…	…
Credential Data Input	*If Update_Time changes at all or Status_Flags changes at all*	*Present_Value, Status_Flags, Update_Time*
…		

[Add new **BACnetAuthenticationFactor** production to Clause **21**, p. 448]

```
BACnetAuthenticationFactor ::= SEQUENCE {
    format-type     [0] BACnetAuthenticationFactorType,
    format-class    [1] Unsigned,
    value           [2] OCTET STRING
                -- for encoding of values into this octet string see ANNEX P.
    }
```

[Add new **BACnetAuthenticationFactorFormat** production to Clause **21**, p. 448]

```
BACnetAuthenticationFactorFormat ::= SEQUENCE {
    format-type         [0] BACnetAuthenticationFactorType,
    vendor-id           [1] Unsigned16 OPTIONAL,
    vendor-format       [2] Unsigned16 OPTIONAL
    }
```

[Add new **BACnetAuthenticationFactorType** enumeration to Clause **21**, p. 448]

```
BACnetAuthenticationFactorType ::= ENUMERATED {
     undefined                  (0),
     error                      (1),
     custom                     (2),
     simple-number16            (3),
     simple-number32            (4),
     simple-number56            (5),
     simple-alpha-numeric       (6),
     aba-track2                 (7),
     wiegand26                  (8),
     wiegand37                  (9),
     wiegand37-facility         (10),
     facility16-card32          (11),
     facility32-card32          (12),
     fasc-n                     (13),
     fasc-n-bcd                 (14),
     fasc-n-large               (15),
     fasc-n-large-bcd           (16),
     gsa75                   (17),
     chuid                   (18),
     chuid-full              (19),
     guid                    (20),
     cbeff-A                 (21),
     cbeff-B                 (22),
     cbeff-C                 (23),
     user-password           (24)
     }
```

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetObjectTypesSuppported** BITSTRING appear in Addendum 135-2008*j*-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2008*j*-1.]

[Note: changes to **Clause 21**, **BACnetPropertyStates** production appear in Addendum 135-2008*j*-1]

[Add new object type structure to **ANNEX C**, p. 497]

```
CREDENTIAL-DATA-INPUT::= SEQUENCE {
    object-identifier        [75]    BACnetObjectIdentifier,
    object-name              [77]    CharacterString,
    object-type              [79]    BACnetObjectType,
    present-value            [85]    BACnetAuthenticationFactor,
    description              [28]    CharacterString OPTIONAL,
    status-flags             [111]   BACnetStatusFlags,
    reliability              [103]   BACnetReliability OPTIONAL,
    out-of-service           [81]    BOOLEAN,
    supported-formats        [304]   SEQUENCE OF BACnetAuthenticationFactorFormat,
                         -- accessed as BACnetARRAY
    supported-format-classes [305]   SEQUENCE OF Unsigned OPTIONAL,
                         -- accessed as BACnetARRAY
    update-time              [189]   BACnetTimeStamp,
    profile-name             [168]   CharacterString OPTIONAL
    }
```

[Add new **Clause D.X**, p. 534]

D.X Example of an Credential Data Input object

This example of a credential data input object represents a card reader on the unsecure side of the door. This object supports two variants of the 16-bit facility and 32-bit card number format (FACILITY16_CARD32). These two variants are distinguished from each other by having a format class of 89 and 99 respectively. This object also supports the 37-bit Wiegand format (WIEGAND37).

A valid card was recently read with a facility code of 121 (X'0079') and a card number of 20926 (X'000051BE').

Property:	Object_Identifier =	(Credential Data Input, Instance 1)
Property:	Object_Name =	"Main Entrance Card Reader"
Property:	Object_Type =	CREDENTIAL_DATA_INPUT
Property:	Present_Value =	(FACILITY16_CARD32, 89, X'0079000051BE')
Property:	Description =	"Main entrance south building multi-format card reader"
Property:	Status_Flags =	(FALSE, FALSE, FALSE, FALSE)
Property:	Reliability =	NO_FAULT_DETECTED
Property:	Out_Of_Service =	FALSE

Property:	Supported_Formats =	((FACILITY16_CARD32, 0, 0), (FACILITY16_CARD32, 0, 0), (WIEGAND37, 0, 0))
Property:	Supported_Format_Classes =	(89,99,0)
Property:	Update_Time =	((7 JAN 2008, MONDAY), 15:30:51.70)

135-2008*j*-7. Add a new ACCESS_EVENT event algorithm.

Rationale

A new event algorithm is needed for access control support in BACnet that is related to user actions, authentication and authorization decisions at an Access Point.

Addendum 135-2008*j*-7

[Change **Clause 12.12.5** and **Table 12-15**, in the Event Enrollment object, pp. 188-189]

12.12.5 Event_Type

This read only property, of type BACnetEventType, indicates the type of event algorithm that is to be used to detect the occurrence of events and report to enrolled devices. 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, EXTENDED, UNSIGNED-RANGE, *ACCESS_EVENT*}.

There is a specific relationship between each event algorithm, the parameter list, and the event types that are valid for the event. The Event_Type reflects the algorithm that is used to determine the state of an event. The algorithm for each Event_Type is specified in Clause 13. The Event_Parameters property provides the parameters needed by the algorithm.

The valid combinations of Event_Type, Event_State, and Event_Parameters values are summarized in Table 12-15.

Table 12-15. Event_Types, Event_States, and their Parameters

Event_Type	Event_State	Event_Parameters
…	…	…
ACCESS_EVENT	*NORMAL*	*List_Of_Access_Events* *Access_Event_Time_Reference*

[Change **Clause 12.12.7**, pp. 189-190]

12.12.7 Event_Parameters

The Event_Parameters property, of type BACnetEventParameter, determines the algorithm used to monitor the referenced object and provides the parameter values needed for this algorithm. The meaning of each value in the Event_Parameters depends on the algorithm as indicated by the Event_Type column in Table 12-15. Each of the possible parameters is described below.

…	
Mode_Property_Reference	This parameter, of type BACnetDeviceObjectPropertyReference, applies to the CHANGE_OF_LIFE_SAFETY algorithm. It identifies the 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.
List_Of_Access_Events	*This parameter is a list of BACnetAccessEvent values that applies to the ACCESS_EVENT algorithm. If the value of the referenced property is updated to one of the values in the List_Of_Access_Events, then the Event_State property of the Event Enrollment object makes a transition TO-NORMAL and appropriate notifications are sent.*

Access_Event_Time_Reference	*This parameter, of type BACnetDeviceObjectPropertyReference, applies to the ACCESS_EVENT algorithm. It identifies the object and property (normally the Access_Event_Time property) that provides the last update time of the referenced property that is monitored for access events. This parameter may only reference object properties that are of type BACnetTimeStamp.*

…

[Change **Clause 13.2** and **Table 13-2** through **Table 3-4**, pp. 292-293]

13.2 Intrinsic Reporting

…

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.

In the case of Access Point, the ACCESS_EVENT algorithm is applied for both Access Alarm Events and Access Transaction Events:

(1) Access Alarm Events

The Access_Alarm_Events property of the Access Point lists each of the Access_Event values that shall be reported, when Access_Event_Time changes, as EVENT or ALARM notifications according to the Notify_Type, if the TO-NORMAL bit of Event_Enable is TRUE. The Event_State remains NORMAL.

The Notification Class object referenced by Notification_Class is used to distribute Access Alarm Events.

(2) Access Transaction Events

The Access_Transaction_Events property of the Access Point lists each of the Access_Event values that shall be reported when Access_Event_Time changes, if the TO-NORMAL bit of Event_Enable is TRUE, as EVENT notifications ignoring Notify_Type. The Event_State remains NORMAL. The Event_Time_Stamps TO-NORMAL element is not affected. Acked_Transitions is not affected.

The Notification Class object referenced by Transaction_Notification_Class is used to distribute Access Transaction Events. If Transaction_Notification_Class is not present in the Access Point object, then the Notification Class object referenced by Notification_Class is used. Ack_Required of the respective Notification Class object is ignored and the value FALSE is conveyed in the AckRequired parameter of the event notification message.

Table 13-2. Standard Objects That May Support Intrinsic Reporting

Object Type	Criteria	Event Type
…	…	…
Access Point	*(Access Alarm Events) If Access_Event_Time changes and Access_Event is equal to one of the values in the Access_Alarm_Events list*	*ACCESS_EVENT*
Access Point	*(Access Transaction Events) If Access_Event_Time changes and Access_Event is equal to one of the values in the Access_Transaction_Events list*	*ACCESS_EVENT*

Table 13-3. Standard Object Property Values Returned in Notifications

Object	Event Type	Notification Parameters	Referenced Object's Properties
…	…	…	…
Access Point	*ACCESS_EVENT*	*Access_Event* *Status_Flags* *Access_Event_Tag* *Access_Event_Time* *Access_Credential* *Authentication_Factor(if present)*	*Access_Event* *Status_Flags* *Access_Event_Tag* *Access_Event_Time* *Access_Event_Credential* *Access_Event_Authentication_Factor(if present)*

[1] This parameter conveys a reference to the Log_Buffer property of the Trend Log object.

Table 13-4. Notification Parameters for Standard Event Types

Event Type	Notification Parameters	Description
…	…	…
ACCESS_EVENT	*Access_Event* *Status_Flags* *Access_Event_Tag* *Access_Event_Time* *Access_Credential* *Authentication_Factor (if present)*	*The new value of the referenced property* *The Status_Flags of the referenced object* *The Access_Event_Tag of the referenced object* *The new value of the referenced Access_Event_Time property* *The Access_Event_Credential of the referenced object* *The Access_Event_Authentication_Factor of the referenced object (if present)*

[Change Clause **13.3**, p. 296]

13.3 Algorithmic Change reporting

…

The following event type algorithms are specified in this standard because of their widespread occurrence in building automation and control systems. They are

…

(i) UNSIGNED_RANGE

(j) ACCESS_EVENT

[Add new Clause **13.3.X**, p. 303]

13.3.X ACCESS_EVENT Algorithm

An ACCESS_EVENT event occurs when the value of the property referred to by Access_Event_Time_Reference changes and the referenced property is equal to one of the values contained in the List_Of_Access_Events. This type of event may be applied only to properties that are of type BACnetAccessEvent. For the purposes of event notification, ACCESS_EVENT events generate a TO-NORMAL transition.

[Add new Figure 13-X, p. 303]

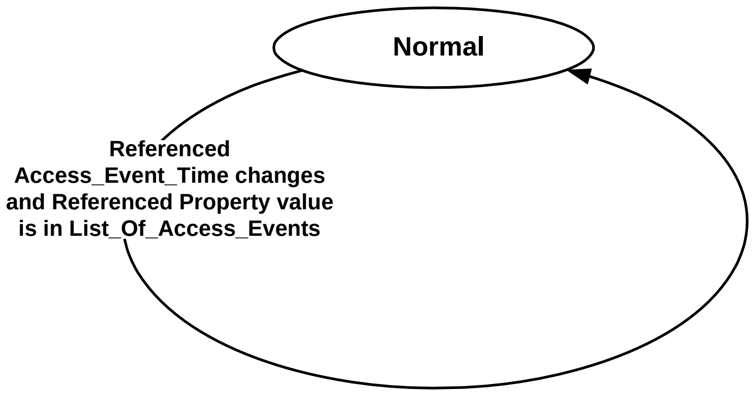
Normal
Referenced Access_Event_Time changes and Referenced Property value is in List_Of_Access_Events

Figure 13-X. ACCESS_EVENT Algorithm

[Change **BACnetEventParameter** production, Clause **21**, pp. 456-457]

```
    BACnetEventParameter ::= CHOICE {
        …
        unsigned-range  [11] SEQUENCE {
                             time-delay    [0] Unsigned,
                             low-limit     [1] Unsigned,
                             high-limit    [2] Unsigned
        ~~}~~},
                              -- context tag 12 is reserved for future addenda
        access-event          [13] SEQUENCE {
                              list-of-access-events        [0] SEQUENCE OF BACnetAccessEvent,
                              access-event-time-reference    [1] BACnetDeviceObjectPropertyReference
                              }
        }
```

[Change **BACnetEventType** enumeration, Clause **21**, p. 458]

```
    BACnetEventType ::= ENUMERATED {
        …
        unsigned-range         (11),
                               -- enumeration value 12 is reserved for future addenda
        access-event           (13),
        …
        }
    -- Enumerated values 0-63 are reserved for definition by ASHRAE. Enumerated values
    -- 64-65535 may be used by others subject to the procedures and constraints described
    -- in Clause 23. It is expected that these enumerated values will correspond to the use of the
    -- complex-event-type CHOICE [6] of the BACnetNotificationParameters production.
~~--    The last enumeration used in this version is 11.~~
```

[Change **BACnetNotificationParameters**, Clause **21**, pp. 461-462]

[Note: **BACnetAuthenticationFactor** is defined in Addendum 135-2008*j*-6.]

BACnetNotificationParameters ::= CHOICE {

```
        …
        unsigned-range    [11] SEQUENCE {
                exceeding-value       [0] Unsigned,
                status-flags          [1] BACnetStatusFlags,
                exceeded-limit        [2] Unsigned
        ~~}~~},
                          -- context tag 12 is reserved for future addenda
        access-event      [13] SEQUENCE {
            access-event              [0] BACnetAccessEvent,
            status-flags              [1] BACnetStatusFlags,
            access-event-tag          [2] Unsigned,
            access-event-time         [3] BACnetTimeStamp,
            access-credential         [4] BACnetDeviceObjectReference,
            authentication-factor     [5] BACnetAuthenticationFactor OPTIONAL
            }
    }
```

135-2008*j*-8. Add a new ANNEX P, BACnet encoding rules for authentication factor values.

Rationale
A new normative annex is needed to specify how standard and other authentication factors are encoded in the value field of the BACnetAuthenticationFactorType structure.

Addendum 135-2008*j*-8

[Add new **ANNEX P**, p. 683]

ANNEX P – BACnet ENCODING OF STANDARD AUTHENTICATION FACTOR FORMATS (NORMATIVE)

(This annex is part of this standard and is required for its use.)

In the physical access control industry there are a number of established standards and defacto standards for authentication factor formats as well as numerous proprietary formats that are widely used. Because the access control industry is rapidly changing and evolving due to government mandates and the emergence of new technology, it is expected that new authentication factor formats will continue to emerge on a regular basis.

Due to the wide variety of authentication factor formats, a specific BACnet ASN.1 encoding for each format is not practical. In addition, because of the vast variety in the size and structure of the authentication factor formats, a single common structure that defines all different formats is considered too complex and therefore not feasible.

The BACnet structure BACnetAuthenticationFactor is used to encapsulate an authentication factor value. Additional attributes are included that identify the authentication factor format and encoding scheme. The structure has the following fields:

Format-Type	This enumeration (BACnetAuthenticationFactorType) specifies the internal representation of the authentication factor value in the 'value' field. The value of this field defines how the authentication factor data in the 'value' field is encoded.
Format-Class	This is a site-specific identifier used to distinguish between different authentication factor value formats that have the same format type. When Format-Type is UNDEFINED, Format_Class shall have a value of zero.
Value	This is an octet string that holds the authentication factor value. The internal format used is specified by the value in the format-type field. The encoding of the authentication factor value is defined in the corresponding entry in Table X-1.

Encapsulating the authentication factor values in an octet string is advantageous because it allows BACnet devices to read, write and use authentication factors without having explicit knowledge of the encoding/decoding of all or any of the authentication factor formats. The rules for encoding and decoding are typically required only by the credential reader and the access credential provisioning device (e.g., a workstation).

Table P-1: Authentication Factor Value Encoding Rules

Format Type (BACnetAuthenticationFactorType)	Authentication Factor Format Description	Authentication Factor Value Encoding[1]
UNDEFINED	Undefined – no authentication factor value is specified	Octet String Size = 0

Format Type (BACnetAuthenticationFactorType)	Authentication Factor Format Description	Authentication Factor Value Encoding[1]
ERROR	Error – this is used when the authentication factor value is not the value expected, or could not be interpreted as expected.	Octet String Size = n Octet [1] = error reason, as follows: 0 = Unspecific error 1 = Parity failure 2 = Too few data 3 = Too much data 4 = Incomplete read 128..255 = Any proprietary error reason Octet[2..3] = authentication factor format type expected (if unknown or cannot be determined use UNDEFINED) Octet[4..n] = data array that holds erroneous data

CUSTOM	Custom (proprietary, or industry standard) format – each format specified is identified by the vendor ID and the proprietary format ID	Octet String Size = n
		Octet[1..2] = BACnet vendor-id (i.e., unsigned 16)
		Octet[3..4] = proprietary type id (i.e., unsigned 16)
		Octet[5..n] = data array that holds proprietary format

SIMPLE_NUMBER16	Simple unsigned number with range [0 .. 65535]	Octet String Size = 2, Octet[1..2] = number (i.e., unsigned 16-bit number)
SIMPLE_NUMBER32	Simple unsigned number with range [0 .. 4294967295]	Octet String Size = 4, Octet[1..4] = number (i.e., unsigned 32-bit number)
SIMPLE_NUMBER56	Simple unsigned number with range [0 .. 72057594037927935] Typically used for DESFire card Serial Numbers	Octet String Size = 7, Octet[1..7] = number (i.e., unsigned 56-bit number)
SIMPLE_ALPHA_ NUMERIC	Simple alpha numeric string	Octet String Size = n, Octet[1] = length of character string in octets including character set specifier (max 255) Octet[2] = character set specifier (as specified in 20.2.9 excluding DBCS, i.e. a value of X'01') Octet[3..n] = string of characters (encoded as specified in 20.2.9)

Format Type (BACnetAuthenticationFactorType)	Authentication Factor Format Description	Authentication Factor Value Encoding[1]
ABA_TRACK2	Magnetic stripe card format (BCD[2] format) as developed by the banking industry (ABA).	Octet String Size = 15, Octet[1..10 (MS nibble)] = primary account number (19 digits) Octet[10 (LS nibble) – 12(MS nibble)] = 4 digit expiration date in form "MMYY" Octet[12 (LS nibble)..13] = 3 digit service code Octet[14.. 15] = discretionary data (4 digits)

WIEGAND26	Standard 26-bit Wiegand format as defined by the SIA standard (SIA AC-01).It is separated into facility code and card number.	Octet String Size = 3
		Octet[1] = facility-code (i.e., unsigned 8-bit number)
		Octet[2..3] = card-number (i.e., unsigned 16-bit number)

WIEGAND37	37 bit Wiegand format with a 35 bit card number. (HID 37 bit format. – H10302)	Octet String Size = 5 Octet[1..5] = card-number (i.e., unsigned 40-bit number with range (0..34359738367))
WIEGAND37_ FACILITY	37 bit Wiegand format with a 16 bit facility code and 19 bit card number. (HID 37 bit format with facility code. – H10304)	Octet String Size = 5 Octet[1..2] = facility-code (i.e., unsigned 16-bit number) Octet[3..5] = card-number (i.e., unsigned 24-bit number with range (0..524287))

FACILITY16_CARD32	Non-standard Wiegand variants that have 32 bit card number and 16 bit facility code formats.	Octet String Size = 6
		Octet[1..2] = facility-code (i.e., unsigned 16-bit number)
		Octet[3..6] = card-number (i.e., unsigned 32-bit number)

FACILITY32_CARD32	Non-standard Wiegand variants that have 32 bit card number and 32 bit facility code formats.	Octet String Size = 8
		Octet[1..4] = facility-code (i.e., unsigned 32-bit number)
		Octet[5..8] = card-number (i.e., unsigned 32-bit number)

Format Type (BACnetAuthenticationFactorType)	Authentication Factor Format Description	Authentication Factor Value Encoding[1]

FASC_N	Federal Agency Smart Credential – Number.
	
	Includes only agency code, system code and credential number.
		Octet String Size = 8
		
		Octet[1..2] = agency-code (i.e., unsigned 16-bit number)
		
		Octet[3..4] = system-site code (i.e., unsigned 16-bit number)
		
		Octet[5..8] = credential number (i.e., unsigned 32-bit number)
		
		-- refer to NIST technical implementation Guidance document for more details

FASC_N_BCD	Federal Agency Smart Credential – Number (BCD[2] format)
	
	Includes only agency code, system code and credential number.
		Octet String Size = 7
		
		Octet[1..2] = agency-code (4-digit BCD number)
		
		Octet[3..4] = system-site code (4-digit BCD number)
		
		Octet[5..7] = credential number (6-digit BCD number)
		
		-- refer to NIST technical implementation Guidance document for more details

FASC_N_LARGE	Federal Agency Smart Credential – Number.
	
	Includes all FASC-N data fields excluding start sentinel, end sentinel, field separators and LRC.
		Octet String Size = 19
		
		Octet[1..2] = agency code (i.e., unsigned 16-bit number)
		
		Octet[3..4] = system/site code (i.e., unsigned 16-bit number)
		
		Octet[5..8] = credential number (i.e., unsigned 32-bit number)
		
		Octet[9] = series code (i.e., unsigned 8-bit number)
		
		Octet[10] = credential code (i.e., unsigned 8-bit number)
		
		Octet[11..15] = person identifier (i.e., Unsigned 40-bit number)
		
		Octet[16] = organizational category (i.e., unsigned 8-bit number)
		
		Octet[17..18] = organizational identifier (i.e., unsigned 16-bit number)
		
		Octet[19] = association category (i.e., unsigned 8-bit number)
		
		-- refer to NIST technical implementation Guidance document for more details

Format Type (BACnetAuthenticationFactorType)	Authentication Factor Format Description	Authentication Factor Value Encoding[1]
FASC_N_LARGE_BCD	Federal Agency Smart Credential – Number. (BCD[2] format) Includes all FASC-N data fields excluding start sentinel, end sentinel, field separators and LRC.	Octet String Size = 16 Octet[1..2] = agency-code (4-digit BCD number) Octet[3..4] = system-site code (4-digit BCD number) Octet[5..7] = credential number (6-digit BCD number) Octet[8 (MS nibble)] = series code (1-digit BCD number) Octet[8 (LS nibble)] = credential code (1-digit BCD number) Octet[9..13] = credential number (10-digit BCD number) Octet[14 (MS nibble)] = organizational category (1 digit BCD number) Octet[14 (LS nibble)..16(MS nibble)] = organizational identifier (4 digit BCD number) Octet[16 (LS nibble)] = association category (1 digit BCD number) -- refer to NIST technical implementation Guidance document for more details

GSA75	GSA 75 bit (FASC-N plus expiry date)	Octet String Size = 12
		
		Octet[1..2] = agency-code (i.e., unsigned 16-bit number)
		
		Octet[3..4] = system-site code (i.e., unsigned 16-bit number)
		
		Octet[5..8] = credential number (i.e., unsigned 32-bit number)
		
		Octet[9..12] = expiry date (4 octets encoded as specified in Clause 20.2.12)

Format Type (BACnetAuthenticationFactorType)	Authentication Factor Format Description	Authentication Factor Value Encoding[1]
CHUID	Card Holder Unique Identifier (CHUID), without Asymmetric Key and without Authentication Key MAP. See SP 800-73 Section 1.8.3 (Figure 1 & 2 pg 12 of the TIG 2.3)	Octet String Size = 45 Octet[1..8] = FASC-N as specified in FASC_N Octet[9..12] = agency code (4 ANSI.X3.4 characters as defined in SP 800-73 (Section 6.4, p. 34, of the TIG 2.3) Octet[13..16] = organization identifier (4 ANSI.X3.4 characters as defined in SP 800-73 (Section 6.4, p. 34, of the TIG 2.3) Octet[17..25] = DUNS number (9 ANSI.X3.4 numeric characters as defined in SP 800-73 (Figures 1 & 2 of the TIG 2.3) Octet[26..41] = GUID (IPv6 address as defined in SP 800-73 (Figures 1 & 2 of the TIG 2.3) Octet[42..45] = Expiry Date expiry date (4 octets encoded as specified in Clause 20.2.12)
CHUID_FULL	Complete Card Holder Unique Identifier stored as data string. The data elements are decoded using the CHUID tags which are embedded in the data string. See SP 800-73 Section 1.8.3 (Figure 1 & 2 pg 12 of the TIG 2.3)	Octet String Size = n (maximum size = 3397) Octet[1..n] = CHUID data string -- Octet encoding is defined in SP 800-73 (Figure 1 & 2 of the TIG 2.3) using CHUID Tags.
GUID	Global unique identifier represented as IPv6 address	Octet String Size = 16 -- Refer to RFC 2373 for format description and encoding
CBEFF_A	Common Biometric Exchange File Format (CBEFF) Patron format A	Octet String Size = n Octet[1..n] = CBEFF data -- NIST CBEFF Patron Format A (CBEFF) content formatted
CBEFF_B	Common Biometric Exchange File Format (CBEFF) Patron format B	Octet String Size = n Octet[1..n] = CBEFF data -- NIST CBEFF Patron Format B (BioAPI) content formatted

Format Type (BACnetAuthenticationFactorType)	Authentication Factor Format Description	Authentication Factor Value Encoding[1]
CBEFF_C	Common Biometric Exchange File Format (CBEFF) Patron format C	Octet String Size = n Octet[1..n] = CBEFF data -- NIST CBEFF Patron Format C (ANSI Standard X9.84) content formatted
USER_PASSWORD	User name and password	Octet String Size = n, Octet[1] = length of user name string in octets including character set specifier (max 255) Octet[2] = character set specifier for user name string (as specified in 20.2.9 excluding DBCS, i.e. a value of X'01') Octet[3..m] = string of characters for user name (encoded as specified in Clause 20.2.9) Octet[m+1] = length of password string in octets including character set specifier (max 255) Octet[m+2] = character set specifier for password string (as specified in 20.2.9 excluding DBCS, i.e. a value of X'01') Octet[m+3..n] = string of characters for password (encoded as specified in Clause 20.2.9)

[1] Multi-octet fields shall be conveyed with the most significant octet first.

[2] In BCD (binary coded decimal) format, each octet holds two 4-bit BCD encoded decimal digits. Bits 7 to 4 convey the most significant digit, while Bits 3 to 0 convey the least significant digit.

[Add a new entry to **History of Revisions**, p. 688]

(This History of Revisions is not part of this standard. It is merely informative and does not contain requirements necessary for conformance to the standard.)

HISTORY OF REVISIONS

Protocol		***Summary of Changes to the Standard***
Version	***Revision***	
…	…	**…**
1	9	**Addendum *j* to ANSI/ASHRAE 135-2008** Approved by the ASHRAE Standards Committee June 20, 2009; by the ASHRAE Board of Directors June 24, 2009; and by the American National Standards Institute June 25, 2009. 1. Add a new Access Point object type. 2. Add a new Access Zone object type. 3. Add a new Access User object type. 4. Add a new Access Rights object type. 5. Add a new Access Credential object type. 6. Add a new Credential Data Input object type. 7. Add a new ACCESS_EVENT event algorithm. 8. Add a new ANNEX P BACnet encoding rules for authentication factor values.

POLICY STATEMENT DEFINING ASHRAE'S CONCERN FOR THE ENVIRONMENTAL IMPACT OF ITS ACTIVITIES

ASHRAE is concerned with the impact of its members' activities on both the indoor and outdoor environment. ASHRAE's members will strive to minimize any possible deleterious effect on the indoor and outdoor environment of the systems and components in their responsibility while maximizing the beneficial effects these systems provide, consistent with accepted standards and the practical state of the art.

ASHRAE's short-range goal is to ensure that the systems and components within its scope do not impact the indoor and outdoor environment to a greater extent than specified by the standards and guidelines as established by itself and other responsible bodies.

As an ongoing goal, ASHRAE will, through its Standards Committee and extensive technical committee structure, continue to generate up-to-date standards and guidelines where appropriate and adopt, recommend, and promote those new and revised standards developed by other responsible organizations.

Through its *Handbook*, appropriate chapters will contain up-to-date standards and design considerations as the material is systematically revised.

ASHRAE will take the lead with respect to dissemination of environmental information of its primary interest and will seek out and disseminate information from other responsible organizations that is pertinent, as guides to updating standards and guidelines.

The effects of the design and selection of equipment and systems will be considered within the scope of the system's intended use and expected misuse. The disposal of hazardous materials, if any, will also be considered.

ASHRAE's primary concern for environmental impact will be at the site where equipment within ASHRAE's scope operates. However, energy source selection and the possible environmental impact due to the energy source and energy transportation will be considered where possible. Recommendations concerning energy source selection should be made by its members.