

BSR/ASHRAE Addendum *j*
to ANSI/ASHRAE Standard 135-2004

Public Review Draft

ASHRAE® Standard

Proposed Addendum *j* to Standard 135-2004, *BACnet®—A Data Communication Protocol for Building Automation and Control Networks*

Second Public Review (September 2007)
(Draft Shows Proposed Changes to
Current Standard)

This draft has been recommended for public review by the responsible project committee. To submit a comment on this proposed addendum, use the comment form and instructions provided with this draft. The draft is subject to modification until it is approved for publication by the Board of Directors and ANSI. Until this time, the current edition of the standard (as modified by any published addenda on the ASHRAE web site) remains in effect. The current edition of any standard may be purchased from the ASHRAE Bookstore @ <http://www.ashrae.org> or by calling 404-636-8400 or 1-800-527-4723 (for orders in the U.S. or Canada).

This standard is under continuous maintenance. To propose a change to the current standard, use the change submittal form available on the ASHRAE web site @ <http://www.ashrae.org>.

The appearance of any technical data or editorial material in this public review document does not constitute endorsement, warranty, or guaranty by ASHRAE of any product, service, process, procedure, or design, and ASHRAE expressly disclaims such.

© **March 15, 2007**. This draft is covered under ASHRAE copyright. Permission to reproduce or redistribute all or any part of this document must be obtained from the ASHRAE Manager of Standards, 1791 Tullie Circle, NE, Atlanta, GA 30329. Phone: 404-636-8400, Ext. 1125. Fax: 404-321-5478. E-mail: standards.section@ashrae.org.

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



[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

The purpose of this addendum is to present a proposed change for public review. 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 proposed changes are summarized below.

SSPC 135 wishes to recognize the efforts of the following people in developing this addendum: Corey Balfour, Howard Coleman, Sharon Dinges, Stuart Donaldson, David Fisher, John Hartman, Bernhard Isler, Simon Lemaire, Les Mather, Kornelia Mergner, Hans-Joachim Mundt, Masahiro Ogawa, David Ritter, Stephen Treado, David White, and Rob Zivney,

- 135-2004j-1. Add a new Access Point object type, p. 1.
- 135-2004j-2. Add a new Access Zone object type, p. 24.
- 135-2004j-3. Add a new Access User object type, p. 35.
- 135-2004j-4. Add a new Access Rights object type, p. 39.
- 135-2004j-5. Add a new Access Credential object type, p. 45.
- 135-2004j-6. Add a new Authentication Factor Input object type, p. 54.
- 135-2004j-7. Add a new ACCESS_EVENT event algorithm, p. 60.
- 135-2004j-8. Add a new Annex X BACnet encoding rules for authentication factor values, p. 65.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-2004 and Addenda is indicated through the use of *italics*, while deletions are indicated by ~~strike through~~. Where entirely new subclauses are proposed to be added, plain type is used throughout. Only this new and deleted text is open to comment as this time. All other material in this addendum is provided for context only and is not open for public review comment except as it relates to the proposed changes.

135-2004j-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 135-2004j-1

[Insert the following new clauses in 3.2, pp. 2-5, distributing them alphabetically and renumbering subsequent clauses]

3.2.x authentication factor: a piece of information 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 access the zone that they have requested to enter through an access control door.

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

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

[Insert new Clause 12.X and Table 12-X, p. 251]

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 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 multifactor authentication, 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 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 Authentication Factor 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 transaction, such as a request to enter or an operator action, can result in one or more access events.

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	O
Out_Of_Service	BOOLEAN	R
Active_Authentication_Policy	Unsigned	O ¹
Number_Of_Authentication_Policies	Unsigned	O ¹
Authentication_Policy_List	BACnetARRAY[N]of BACnetAuthenticationPolicy	O ²
Authentication_Policy_Names	BACnetARRAY[N] of CharacterString	O ²
Authentication_Factor_Input_List	BACnetARRAY[N] of BACnetDeviceObjectReference	O
Authorization_Mode	BACnetAuthorizationMode	O
Lockdown	BOOLEAN	O ³
Lockdown_Relinquish_Time	Unsigned	O
Failed_Attempts	Unsigned	O
Failed_Attempt_Events	List of BACnetAccessEvent	O
Max_Failed_Attempts	Unsigned	O ⁴
Failed_Attempts_Time	Unsigned	O ⁴
Threat_Level	BACnetAccessThreatLevel	O
Occupancy_Upper_Limit_Enforced	BOOLEAN	O
Occupancy_Lower_Limit_Enforced	BOOLEAN	O
Access_Event	BACnetAccessEvent	O ^{5,6}
Access_Event_Time	BACnetTimeStamp	O ^{5,6}
Access_Event_Credential	BACnetObjectIdentifier	O ^{5,6}
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 ⁵
Transaction_Notification_Class	Unsigned	O
Access_Alarm_Events	List of BACnetAccessEvent	O ⁵
Access_Transaction_Events	List of BACnetAccessEvent	O ⁵
Event_Enable	BACnetEventTransitionBits	O ⁵
Acked_Transitions	BACnetEventTransitionBits	O ⁵
Notify_Type	BACnetNotifyType	O ⁵
Event_Time_Stamps	BACnetARRAY[3] of BACnetTimeStamp	O ⁵
Profile_Name	CharacterString	O

¹ These properties are required to be present if this object supports authentication policies.
² The size of this array shall equal the value of the Number_Of_Authentication_Policies property.
³ This property is required to be present if Lockdown_Relinquish_Time is present.
⁴ If this property is present, then the Failed_Attempts property shall be present.
⁵ These properties are required to be present if the object supports intrinsic reporting.
⁶ These properties are required if the object supports COV 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 property is present and does not have a value of 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 property is not present, then the value of Event_State shall be NORMAL, or
- (b) if the Reliability property is present and Reliability is NO_FAULT_DETECTED, then Event_State shall be NORMAL, or
- (c) if the Reliability property is present and Reliability is not NO_FAULT_DETECTED, then Event_State shall be FAULT.

12.X.7 Reliability

The optional 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, 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.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.

12.X.9 Active_Authentication_Policy

This optional 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, the Reliability property shall have the value CONFIGURATION_ERROR.

If this property is present, the Number_Of_Authentication_Policies property shall be present.

12.X.10 Number_Of_Authentication_Policies

This optional 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 Authentication_Policy_Names array, if present, shall also be changed to the same value.

12.X.11 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 elements of type BACnetAuthenticationRule that specifies the authentication factor inputs which are used for this policy. Each element of the list has the following fields:
Authentication-Factor-Input	This field, of type BACnetDeviceObjectReference, contains a reference to an object of type Authentication Factor Input where the authentication factor value is read.
Index	This field, of type Unsigned, indicates the order in which the authentication factors will be evaluated. 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 an option between any of these authentication factors and the user shall 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 all the authentication factors are required to be presented again.

An Authentication_Policy_List array element shall be considered invalid if the Policy field is empty or if it is not well formed.

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.11.1 Initializing New Array Elements When the Array Size is Increased

If the size of the Authentication_Policy_List array is increased without entry values being provided, then the new array entries shall be initialized with an empty Policy list, Order-Enforced shall be FALSE, and Timeout shall have the value zero.

12.X.12 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.13 Authentication_Factor_Input_List

This optional property, of type `BACnetARRAY[N]` of `BACnetDeviceObjectReference`, is used to specify those Authentication Factor Input objects that are used by this Access Point object. It is a local matter as to how this array is used. The array may be empty or not present if the vendor does not wish to expose the individual Authentication Factor Input objects that are used by this Access Point.

12.X.14 Authorization_Mode

This optional property, of type `BACnetAuthorizationMode`, determines how authorization is performed at the Access Point. The enumeration `BACnetAuthorizationMode` has the values:

<code>AUTHORIZE</code>	The access rights of an active credential are evaluated, in addition to other possible authorization checks.
<code>GRANT_ACTIVE</code>	An active credential is granted access without evaluating the access rights assigned to the credential. Other authorization checks can still lead to denying access.
<code>DENY_ALL</code>	Any credential, whether active or not, is denied access. Other authorization checks may be performed, but shall not lead to granting access.
<code><Proprietary Enum Values></code>	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.15 Lockdown

This optional property, of type `BOOLEAN`, is an indication whether (`TRUE`) or not (`FALSE`) the access controlled point this object represents is in a lockdown state. When the Access Point is in a lockdown state, any access request shall always be denied, except for a credential that has master exemption. For each denied access request the `Access_Event`, if present, shall be set to `DENIED_LOCKDOWN`. An Access Point may be set to a lockdown 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 `Lockdown` becomes `TRUE` due to too many failed authentication attempts, the Access Point object shall set `Access_Event` to `LOCKDOWN_MAX_ATTEMPTS`. If `TRUE` is written to this property for any other reason, the Access Point object shall set `Access_Event` to `LOCKDOWN_OTHER`. When the `Lockdown` property becomes `FALSE`, the Access Point shall set `Access_Event` to `LOCKDOWN_RELINQUISHED`.

12.X.16 Lockdown_Relinquish_Time

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

If the `Lockdown_Relinquish_Time` is present, the `Lockdown` property shall be present.

12.X.17 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 `Lockdown` becomes `FALSE`.

12.X.18 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.19 Max_Failed_Attempts

This optional property, of type Unsigned, shall specify the maximum number of successive failed access attempts before the Lockdown 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 Lockdown property is set to TRUE. Zero indicates that the Lockdown property is not set to TRUE as the result of failed access attempts.

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

12.X.20 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 an intervening successful access occurs, the Failed_Attempts property is set to zero.

If the Failed_Attempts_Time is present the Failed_Attempts property shall be present.

12.X.21 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.

12.X.22 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.

12.X.23 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.

12.X.24 Access_Event

This optional property, of type BACnetAccessEvent, shall specify the last access event at the 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. This is not a reported event. It is required to enable algorithmic ACCESS_EVENT reporting.

MUSTER	If the Access Point is a muster point a muster event is generated when an Access Credential is presented.
PASSBACK_DETECTED	A passback violation for the presented Access Credential has been detected.
DURESS	A duress incident was detected at this Access Point.
TRACE	The Access Credential presented has the Trace_Flag set.
LOCKDOWN_MAX_ATTEMPTS	The Access Point is in a lockdown state due to maximum failed authentication attempts.
LOCKDOWN_OTHER	The Access Point is in a lockdown state due to any reason other than maximum failed authentication attempts.
LOCKDOWN_RELINQUISHED	The Access Point has relinquished the lockdown state.
LOCKED_BY_HIGHER_PRIORITY	The controlled Access Door is commanded at a higher priority.
GRANTED	Access granted to the presented Access Credential
DENIED_DENY_ALL	Access denied because the authorization mode of the Access Point is set to DENY_ALL.
DENIED_UNKNOWN_CREDENTIAL	Access denied due to unknown credential value
DENIED_MISSING_AUTHENTICATION_FACTOR	Access denied due to missing authentication factor for multi-factor authentication.
DENIED_ZONE_NO_ACCESS_RIGHTS	Access denied due to no matching Access Rights to the Access Zone found for the presented Access Credential.
DENIED_POINT_NO_ACCESS_RIGHTS	Access denied due to no matching Access Rights to the Access Point found for the presented Access Credential.
DENIED_OUT_OF_TIME_RANGE	Access denied due to the presented Access 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 Access Credential.
DENIED_PASSBACK	Access denied due to a passback violation for the Access Credential.
DENIED_UNEXPECTED_LOCATION_USAGE	Access denied due to the Access Credential used at a location which is not expected.
DENIED_MAX_ATTEMPTS	Access denied due to too many retries for the presented Access Credential.
DENIED_LOW_OCCUPANCY_LIMIT	Exit from a zone for which this Access Point is an Exit Access Point is denied due to zone occupancy below or at the minimum limit.
DENIED_HIGH_OCCUPANCY_LIMIT	Access to a zone for which this Access Point is an Entry Access Point is denied due to zone occupancy at or above the maximum limit.
DENIED_CREDENTIAL_UNASSIGNED	Access denied due to the Access Credential used has not yet been assigned to an Access User.

DENIED_CREDENTIAL_INVALID	Access denied due to the Access Credential used is invalid.
DENIED_CREDENTIAL_NOT_YET_ACTIVE	Access denied due to the Access Credential used is not yet active.
DENIED_CREDENTIAL_EXPIRED	Access denied due to the Access Credential used is expired.
DENIED_CREDENTIAL_MANUAL_DISABLE	Access denied due to the Access Credential used is manually disabled.
DENIED_CREDENTIAL_LOCKOUT	Access denied due to the Access Credential used is locked out.
DENIED_CREDENTIAL_EXCEEDED_USE	Access denied due to the number of allowed uses of the Access Credential used has been exceeded.
DENIED_CREDENTIAL_INACTIVITY	Access denied due to the Access Credential used being disabled after a period of inactivity.
DENIED_CREDENTIAL_LOST	Access denied due to the Access Credential used being reported as lost.
DENIED_CREDENTIAL_STOLEN	Access denied due to the Access Credential used being reported as stolen.
DENIED_CREDENTIAL_DAMAGED	Access denied due to the Access Credential used being reported as damaged.
DENIED_CREDENTIAL_DESTROYED	Access denied due to the Access Credential used being reported as destroyed.
DENIED_CREDENTIAL_DISABLED	Access denied due to the Access Credential used is disabled for unspecified or unknown reasons.
DENIED_CREDENTIAL_TIMEOUT	Access denied due to the proper Access Credential not being presented within a specified time.
DENIED_INCORRECT_CREDENTIAL	Access denied due to the Access Credential entered is incorrect or not the credential expected.
DENIED_NO_ACCOMPANIMENT	Access denied due to the expected accompanying Access Credential not being presented.
DENIED_INCORRECT_ACCOMPANIMENT	Access denied due to the accompanying Access Credential presented was incorrect
DENIED_LOCKDOWN	Access denied due to the Access Point being in lockdown state.
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.

This property is required if intrinsic reporting is supported by this object. This property is also required if the object supports COV reporting.

12.X.25 Access_Event_Time

This optional 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.

This property shall be present if the Access_Event property is present or if the object supports COV reporting.

12.X.26 Access_Event_Credential

This optional property, of type BACnetObjectIdentifier, shall specify the Access Credential object that corresponds to the access event specified in the Access_Event property, if applicable. Otherwise it shall contain 4194303 for the object instance.

If there was no credential recognized up to now, this property shall contain 4194303 for the object instance.

This property shall be present if the Access_Event property is present or if the object supports COV reporting.

12.X.27 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 UNDEFINED with NULL in the value field.

If there was no authentication factor read up to now, or the device chooses not to expose the last authentication factor, this property shall contain UNDEFINED with NULL in the value field.

12.X.28 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.29 Priority_For_Writing

This property, of type Unsigned (1..16), defines the priority at which the referenced Access Door objects 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.30 Muster_Point

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

12.X.31 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.

12.X.32 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.

12.X.33 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.34 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.35 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.36 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.37 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.38 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.39 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.40 Event_Time_Stamps

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

12.X.41 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.254]

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

Object Type	Criteria	Properties Reported
...
<i>Access Point</i>	<i>If Access_Event_Time changes at all</i>	<i>Access_Event, Status_Flags, Access_Event_Time, Access_Event_Credential, Access_Event_Authentication_Factor (if present)</i>
...		

[Add new clause **19.2.7**, p.365]

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. 408]

```

BACnetAccessEvent ::= ENUMERATED {
    none (0),
    muster (1),
    passback-detected (2),
    duress (3),
    trace (4),
    lockdown-max-attempts (5),
    lockdown-other (6),
    lockdown-relinquished (7),
    locked-by-higher-priority (8),
    granted (9),
    denied-deny-all (10),
    denied-unknown-credential (11),
    denied-missing-authentication-factor (12),
    denied-zone-no-access-rights (13),
    denied-point-no-access-rights (14),
    denied-out-of-time-range (15),
    denied-threat-level (16),
    denied-passback (17),
    denied-unexpected-location-usage (18),
    denied-max-attempts (19),
    denied-low-occupancy-limit (20),
    denied-high-occupancy-limit (21),
    denied-credential-unassigned (22),
    denied-credential-invalid (23),
    denied-credential-not-yet-active (24),
    denied-credential-expired (25),
    denied-credential-manual-disable (26),
    denied-credential-lockout (27),
    denied-credential-exceeded-use (28),
    denied-credential-inactivity (29),
    denied-credential-lost (30),
    denied-credential-stolen (31),

```

denied-credential-damaged (32),
denied-credential-destroyed (33),
denied-credential-disabled (34),
denied-credential-timeout (35),
denied-incorrect-credential (36),
denied-no-accompaniment (37),
denied-incorrect-accompaniment (38),
denied-lockdown (39),
denied-other (40),

...
}

-- 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. 408]

BACnetAccessThreatLevel ::= Unsigned(0..100)

[Note: **BACnetAuthenticationFactor** is defined in Addendum 135-2004j-6.]

[Note: **BACnetAuthenticationFactorType** is defined in Addendum 135-2004j-6.]

[Add new **BACnetAuthenticationPolicy** production to Clause 21, p. 408]

BACnetAuthenticationPolicy ::= SEQUENCE {
policy [0] SEQUENCE OF BACnetAuthenticationRule,
order-enforced [1] BOOLEAN,
timeout [2] Unsigned
}

[Add new **BACnetAuthenticationRule** production to Clause 21, p. 408]

BACnetAuthenticationRule ::= SEQUENCE {
authentication-factor-input [0] BACnetDeviceObjectReference,
authentication-factor-choice [1] Unsigned
}

[Add new **BACnetAuthorizationMode** production to Clause 21, p. 408]

BACnetAuthorizationMode ::= ENUMERATED {
authorize (0),
grant-active (1),
deny-all (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.

[Change **BACnetObjectType** production in Clause 21, pp. 421-422]

[Note: enumeration values 25-31 are defined in Addendum 135-2004b-1, d-1, e-1, f-1 and i-1.]

```
BACnetObjectType ::= ENUMERATED {  
    access-credential           (32),  
    access-point               (33),  
    access-rights              (34),  
    access-user                 (35),  
    access-zone                 (36),  
    accumulator                (23),  
    authentication-factor-input (37),  
    ...  
    -- see pulse-converter      (24),  
    -- see access-credential    (32),  
    -- see access-point        (33),  
    -- see access-rights       (34),  
    -- see access-user         (35),  
    -- see access-zone         (36),  
    -- see authentication-factor-input (37),  
    ...  
}
```

[Change **BACnetObjectTypesSupported** production in Clause 21, p. 422]

[Note: bit positions 25-31 are defined in Addendum 135-2004b-1, d-1, e-1, f-1 and i-1.]

```
BACnetObjectTypesSupported ::= BIT STRING {  
    -- access-credential       (32),  
    -- access-point           (33),  
    -- access-rights          (34),  
    -- access-user            (35),  
    -- access-zone            (36),  
    -- accumulator           (23),  
    -- authentication-factor-input (37),  
    ...  
    pulse-converter           (24) (24),  
    access-credential       (32),  
    access-point           (33),  
    access-rights          (34),  
    access-user            (35),  
    access-zone            (36),  
    authentication-factor-input (37)  
}
```

[Change **BACnetPropertyIdentifier** production, Clause 21, pp. 423-428]

[Note: the renaming of the "log-enable" property identifier to "enable" also occurs in Addendum 135-2004b-2.]

[Note: enumerations 312 through 316 are assigned in proposed Addenda 135-2004k and 135-2004n .]

```
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-time         (250),
```

<i>access-rules</i>	(251),
<i>access-rules-enable</i>	(252),
<i>access-transaction-events</i>	(253),
<i>accompanied</i>	(254),
<i>activation-time</i>	(255),
<i>active-authentication-policy</i>	(256),
acked-transitions	(0),
...	
archive	(13),
<i>assigned-access-rights</i>	(257),
attempted-samples	(124),
<i>authentication-factor-input-list</i>	(258),
<i>authentication-factors</i>	(259),
<i>authentication-policy-list</i>	(260),
<i>authentication-policy-names</i>	(261),
<i>authorization-mode</i>	(262),
auto-slave-discovery	(169),
...	
backup-failure-timeout	(153),
<i>belongs-to</i>	(263),
bias	(14),
...	
cov-resubscription-interval	(128),
<i>credential-disable</i>	(264),
<i>credential-status</i>	(265),
<i>credentials</i>	(266),
<i>credentials-in-zone</i>	(267),
-- current-notify-time	(129), This property was deleted in version 1 revision 3.
...	
daylight-savings-status	(24),
<i>days-remaining</i>	(268),
deadband	(25),
...	
effective-period	(32),
elapsed-active-time	(33),
<i>enable</i>	(133), --renamed from previous version
<i>entry-points</i>	(269),
error-limit	(34),
...	
exception-schedule	(38),
<i>exit-points</i>	(270),
<i>expiry-time</i>	(271),
<i>extended-time-enable</i>	(272),
<i>failed-attempt-events</i>	(273),
<i>failed-attempts</i>	(274),
<i>failed-attempts-time</i>	(275),
fault-values	(39),
...	
firmware-revision	(44),
<i>format-class-supported</i>	(276),
<i>format-type</i>	(277),
high-limit	(45),
inactive-text	(46),
...	
in-process	(47),
input-reference	(181),

...	
-- issue-confirmed-notifications	(51), This property was deleted in version 1 revision 4.
<i>last-access-event</i>	(278),
<i>last-access-point</i>	(279),
<i>last-credential-added</i>	(280),
<i>last-credential-added-time</i>	(281),
<i>last-credential-removed</i>	(282),
<i>last-credential-removed-time</i>	(283),
last-notify-record	(173),
last-restore-time	(157),
<i>last-use-time</i>	(284),
life-safety-alarm-values	(166),
...	
location	(58),
<i>lockdown</i>	(285),
<i>lockdown-relinquish-time</i>	(286),
log-buffer	(131),
...	
log-enable	(133),
...	
manual-slave-address-binding	(170),
<i>master-exemption</i>	(287),
maximum-output	(61),
...	
max-apdu-length-accepted	(62),
<i>max-failed-attempts</i>	(288),
max-info-frames	(63),
...	
max-segments-accepted	(167),
<i>members</i>	(289),
member-of	(159),
...	
modification-date	(71),
<i>muster-point</i>	(290),
<i>negative-access-rules</i>	(291),
notification-class	(17), -- renamed from previous version
...	
number-of-APDU-retries	(73),
<i>number-of-authentication-policies</i>	(292),
number-of-states	(74),
...	
object-type	(79),
<i>occupancy-count</i>	(293),
<i>occupancy-count-enable</i>	(294),
<i>occupancy-count-exemption</i>	(295),
<i>occupancy-lower-limit</i>	(296),
<i>occupancy-lower-limit-enforced</i>	(297),
<i>occupancy-state</i>	(298),
<i>occupancy-upper-limit</i>	(299),
<i>occupancy-upper-limit-enforced</i>	(300),
operation-expected	(161),
...	
output-units	(82),
-- see event-parameters	(83),
<i>passback-exemption</i>	(301),
<i>passback-mode</i>	(302),

<i>passback-timeout</i>	(303),
<i>polarity</i>	(84),
<i>positive-access-rules</i>	(304),
<i>prescale</i>	(185),
...	
<i>read-only</i>	(99),
<i>read-status</i>	(305),
<i>reason-for-halt</i>	(100),
<i>reason-for-disable</i>	(306),
-- recipient	(101),
...	
<i>system-status</i>	(112),
<i>threat-authority</i>	(307),
<i>threat-level</i>	(308),
<i>time-delay</i>	(113),
...	
<i>total-record-count</i>	(145),
<i>trace-flag</i>	(309),
<i>tracking-value</i>	(164),
<i>transaction-notification-class</i>	(310),
<i>units</i>	(117),
...	
<i>update-time</i>	(189),
<i>user-external-identifier</i>	(311),
<i>user-information-reference</i>	(317),
<i>user-name</i>	(318),
<i>user-type</i>	(319),
<i>uses-remaining</i>	(320),
<i>utc-offset</i>	(119),
...	
<i>variance-value</i>	(151),
<i>vendor-format-identifier</i>	(321),
<i>vendor-identifier</i>	(120),
...	
<i>weekly-schedule</i>	(123),
<i>zone-from</i>	(322),
<i>zone-to</i>	(323),
-- see attempted-samples	(124),
...	
-- see value-change-time	(192),
-- see <i>absentee-limit</i>	(244),
-- see <i>access-alarm-events</i>	(245),
-- see <i>access-doors</i>	(246),
-- see <i>access-event</i>	(247),
-- see <i>access-event-authentication-factor</i>	(248),
-- see <i>access-event-credential</i>	(249),
-- see <i>access-event-time</i>	(250),
-- see <i>access-rules</i>	(251),
-- see <i>access-rules-enable</i>	(252),
-- see <i>access-transaction-events</i>	(253),
-- see <i>accompanied</i>	(254),
-- see <i>activation-time</i>	(255),
-- see <i>active-authentication-policy</i>	(256),
-- see <i>assigned-access-rights</i>	(257),
-- see <i>authentication-factor-input-list</i>	(258),
-- see <i>authentication-factors</i>	(259),

-- see <i>authentication-policy-list</i>	(260),
-- see <i>authentication-policy-names</i>	(261),
-- see <i>authorization-mode</i>	(262),
-- see <i>belongs-to</i>	(263),
-- see <i>credential-disable</i>	(264),
-- see <i>credential-status</i>	(265),
-- see <i>credentials</i>	(266),
-- see <i>credentials-in-zone</i>	(267),
-- see <i>days-remaining</i>	(268),
-- see <i>entry-points</i>	(269),
-- see <i>exit-points</i>	(270),
-- see <i>expiry-time</i>	(271),
-- see <i>extended-time-enable</i>	(272),
-- see <i>failed-attempt-events</i>	(273),
-- see <i>failed-attempts</i>	(274),
-- see <i>failed-attempts-time</i>	(275),
-- see <i>format-class-supported</i>	(276),
-- see <i>format-type</i>	(277),
-- see <i>last-access-event</i>	(278),
-- see <i>last-access-point</i>	(279),
-- see <i>last-credential-added</i>	(280),
-- see <i>last-credential-added-time</i>	(281),
-- see <i>last-credential-removed</i>	(282),
-- see <i>last-credential-removed-time</i>	(283),
-- see <i>last-use-time</i>	(284),
-- see <i>lockdown</i>	(285),
-- see <i>lockdown-relinquish-time</i>	(286),
-- see <i>master-exemption</i>	(287),
-- see <i>max-failed-attempts</i>	(288),
-- see <i>members</i>	(289),
-- see <i>muster-point</i>	(290),
-- see <i>negative-access-rules</i>	(291),
-- see <i>number-of-authentication-policies</i>	(292),
-- see <i>occupancy-count</i>	(293),
-- see <i>occupancy-count-enable</i>	(294),
-- see <i>occupancy-count-exemption</i>	(295),
-- see <i>occupancy-lower-limit</i>	(296),
-- see <i>occupancy-lower-limit-enforced</i>	(297),
-- see <i>occupancy-state</i>	(298),
-- see <i>occupancy-upper-limit</i>	(299),
-- see <i>occupancy-upper-limit-enforced</i>	(300),
-- see <i>passback-exemption</i>	(301),
-- see <i>passback-mode</i>	(302),
-- see <i>passback-timeout</i>	(303),
-- see <i>positive-access-rules</i>	(304),
-- see <i>read-status</i>	(305),
-- see <i>reason-for-disable</i>	(306),
-- see <i>threat-authority</i>	(307),
-- see <i>threat-level</i>	(308),
-- see <i>trace-flag</i>	(309),
-- see <i>transaction-notification-class</i>	(310),
-- see <i>user-external-identifier</i>	(311),
-- see <i>user-information-reference</i>	(317),
-- see <i>user-name</i>	(318),
-- see <i>user-type</i>	(319),
-- see <i>uses-remaining</i>	(320),

```
-- see vendor-format-identifier      (321),  
-- see zone-from                     (322),  
-- see zone-to                       (323),  
...  
}
```

[Change **Clause 21, BACnetPropertyStates** production, pp. 428-429]

[Note: other additions to this production appear in Addendum 135-2004b-5, f-1 and h-3.]

BACnetPropertyStates ::= CHOICE {

```
...  
access-event                [x] BACnetAccessEvent,  
zone-occupancy-state        [x+1] BACnetAccessZoneOccupancyState,  
access-credential-disable-reason [x+2] BACnetAccessCredentialDisableReason,  
access-credential-disable    [x+3] BACnetAccessCredentialDisable,  
authentication-factor-read-status [x+4] BACnetAuthenticationFactorReadStatus  
}
```

[Change **Table 23-1**, p.437]

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
<i>BACnetAccessCredentialDisable</i>	<i>0-255</i>	<i>65535</i>
<i>BACnetAccessCredentialDisableReason</i>	<i>0-255</i>	<i>65535</i>
<i>BACnetAccessEvent</i>	<i>0-511</i>	<i>65535</i>
<i>BACnetAccessUserType</i>	<i>0-63</i>	<i>65535</i>
<i>BACnetAccessZoneOccupancyState</i>	<i>0-63</i>	<i>65535</i>
<i>BACnetAuthorizationMode</i>	<i>0-63</i>	<i>65535</i>
BACnetDeviceStatus	0-63	65535
...
BACnetVTClass	0-63	65535

[Add to **Annex C**, p.453]

```

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 OPTIONAL,
    out-of-service [81] BOOLEAN,
    active-authentication-policy [256] Unsigned OPTIONAL,
    number-of-authentication-policies [292] Unsigned OPTIONAL,
    authentication-policy-list [260] SEQUENCE OF BACnetAuthenticationPolicy OPTIONAL,
    -- accessed as a BACnetARRAY
    authentication-policy-names [261] SEQUENCE OF CharacterString OPTIONAL,
    -- accessed as a BACnetARRAY
    authorization-mode [262] BACnetAuthorizationMode OPTIONAL,
    authentication-factor-input-list [258] SEQUENCE OF BACnetDeviceObjectReference OPTIONAL,
    lockdown [285] BOOLEAN OPTIONAL,
    lockdown-relinquish-time [286] Unsigned OPTIONAL,
    failed-attempts [274] Unsigned OPTIONAL,
    failed-attempt-events [273] SEQUENCE OF BACnetAccessEvent OPTIONAL,
    max-failed-attempts [288] Unsigned OPTIONAL,
    failed-attempts-time [275] Unsigned OPTIONAL,
    threat-level [308] BACnetAccessThreatLevel OPTIONAL,
    occupancy-upper-limit-enforced [300] BOOLEAN OPTIONAL,
    occupancy-lower-limit-enforced [297] BOOLEAN OPTIONAL,
    access-event [247] BACnetAccessEvent OPTIONAL,
    access-event-time [250] BACnetTimeStamp OPTIONAL,
    access-event-credential [249] BACnetObjectIdentifier OPTIONAL,
    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 [290] BOOLEAN OPTIONAL,
    zone-to [323] BACnetDeviceObjectReference OPTIONAL,
    zone-from [322] BACnetDeviceObjectReference OPTIONAL,
    notification-class [17] Unsigned OPTIONAL,

```

transaction-notification-class	[310]	Unsigned OPTIONAL,
access-alarm-events	[245]	SEQUENCE OF BACnetAccessEvent OPTIONAL,
access-transaction-events	[253]	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	[167]	CharacterString OPTIONAL
}		

[Add new ANNEX D.X, p.484]

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 Authentication Factor 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.

The last event at the Access Point happened 21-MAR-08, 18:50:21.2, and was a passback violation done with the Access Credential object instance 41445 using the simple number authentication factor 1334234499 of the credential class 2, and is not yet acknowledged.

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: Active_Authentication_Policy = 1
Property: Number_Of_Authentication_Policies = 3
Property: Authentication_Policy_List = {
    (((Device, Instance 12) (Authentication Factor Input, Instance 3)),1),
    (((Device, Instance 12) (Authentication Factor Input, Instance 4)),1),
    FALSE, 0),
    (((Device, Instance 12) (Authentication Factor Input, Instance 3)),1),
    (((Device, Instance 12) (Authentication Factor Input, Instance 4)),2),
    TRUE, 0),
    (((Device, Instance 12) (Authentication Factor Input, Instance 3)),1),
    (((Device, Instance 12) (Authentication Factor Input, Instance 5)),2),
    TRUE, 30)
}
Property: Authentication_Policy_Names = ("OFFICE_HOURS", "LATE_SHIFT", "NIGHT")
Property: Authentication_Factor_Input_List = (((Device, Instance 12) (Authentication Factor Input, Instance 3)),
    ((Device, Instance 12) (Authentication Factor Input, Instance 4)),
    ((Device, Instance 12) (Authentication Factor Input, Instance 5)))
Property: Authentication_Policy_Timeouts = (20, 20, 10)
Property: Active_Authentication_Policy = 1
Property: Authorization_Mode = AUTHORIZE
Property: Lockdown = FALSE
Property: Lockdown_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: Access_Event = PASSBACK_DETECTED
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 = (2, 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,
    DENIED_UNKNOWN_CREDENTIAL)
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))

```

135-2004j-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 135-2004j-2

[Insert new clause **12.X** and Table **12-X**, p. 251]

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 supports 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. However, it is a local matter as to how the 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
Occupancy_State	BACnetAccessZoneOccupancyState	R
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Reliability	BACnetReliability	O ¹
Out_Of_Service	BOOLEAN	R
Occupancy_Count	Unsigned	R ¹
Occupancy_Count_Enable	BOOLEAN	R
Adjust_Value	INTEGER	O
Occupancy_Upper_Limit	Unsigned	O
Occupancy_Lower_Limit	Unsigned	O
Credentials_In_Zone	List of BACnetObjectIdentifier	O
Last_Credential_Added	BACnetObjectIdentifier	O
Last_Credential_Added_Time	BACnetDateTime	O
Last_Credential_Removed	BACnetObjectIdentifier	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.

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 Occupancy_State

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

`BACnetAccessZoneOccupancyState` is an enumeration of possible occupancy states.

<code>NORMAL</code>	This is the occupancy state when occupancy counting is enabled and no other standard or proprietary states are applicable.
<code>BELOW_LOWER_LIMIT</code>	If <code>Occupancy_Lower_Limit</code> property is present and the <code>Occupancy_Count</code> property is lower than this value.
<code>AT_LOWER_LIMIT</code>	If <code>Occupancy_Lower_Limit</code> property is present and the <code>Occupancy_Count</code> property is equal to this value.
<code>AT_UPPER_LIMIT</code>	If <code>Occupancy_Upper_Limit</code> property is present and the <code>Occupancy_Count</code> property is equal to this value.
<code>ABOVE_UPPER_LIMIT</code>	If <code>Occupancy_Upper_Limit</code> property is present and the <code>Occupancy_Count</code> property is greater than this value.
<code>NOT_USED</code>	This is the occupancy state when occupancy counting is disabled.
<Proprietary Enum Values>	A vendor may use other proprietary enumeration values to indicate other states based on the <code>Occupancy_Count</code> property other than those defined by this standard. For proprietary extensions of this enumeration, see Clause 23.1 of this standard.

12.X.6 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:

<code>IN_ALARM</code>	Logical FALSE (0) if the <code>Event_State</code> property has a value of <code>NORMAL</code> , otherwise logical TRUE (1).
<code>FAULT</code>	Logical TRUE (1) if the <code>Reliability</code> property is present and does not have a value of <code>NO_FAULT_DETECTED</code> , otherwise logical FALSE (0).
<code>OVERRIDDEN</code>	The value of this flag shall be logical FALSE (0).
<code>OUT_OF_SERVICE</code>	Logical TRUE (1) if the <code>Out_Of_Service</code> property has a value of <code>TRUE</code> , otherwise logical FALSE (0).

12.X.7 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` property is not present, then the value of `Event_State` shall be `NORMAL`, or

- (b) if the Reliability property is present and Reliability is NO_FAULT_DETECTED, then Event_State shall be NORMAL, or
- (c) if the Reliability property is present and Reliability is not NO_FAULT_DETECTED, then Event_State shall be FAULT.

12.X.8 Reliability

The optional 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.8.1 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 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the Occupancy_Count property is decoupled from the processing of occupancy counting. In addition, the Reliability property and the corresponding state of the FAULT flag of the Status_Flags property shall be decoupled when Out_Of_Service is TRUE. While the Out_Of_Service property is TRUE, writing to the Adjust_Value property shall not modify the Occupancy_Count. The Occupancy_Count 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 Occupancy_State or Reliability properties shall respond to changes made to these properties while Out_Of_Service is TRUE.

12.X.10 Occupancy_Count

This 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 the occupancy count is set to.

12.X.11 Occupancy_Count_Enable

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

12.X.12 Adjust_Value

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

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 zero, then the value of the Occupancy_Count property shall be set to zero. If the resulting

value of the Occupancy_Count property would be less than zero, then 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, it shall have a value of zero.

12.X.13 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.14 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.15 Credentials_In_Zone

This optional property, of type List of BACnetObjectIdentifier, 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.16 Last_Credential_Added

This optional property, of type BACnetObjectIdentifier, 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, the instance part of the object identifier holds a value of 4194303. 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.17 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.18 Last_Credential_Removed

This optional property, of type BACnetObjectIdentifier, indicates the reference to the Access Credential object which has last been removed from the Credentials_In_Zone property. When no credential has been removed yet, the instance part of the object identifier holds a value of 4194303. 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.19 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.20 Passback_Mode

This optional property, of type `BACnetAccessPassbackMode`, specifies how all access controlled entry points to this zone shall handle passback violations. Passback modes are:

<code>PASSBACK_OFF</code>	Passback violations are not checked.
<code>HARD_PASSBACK</code>	Passback violations are checked and enforced (i.e., denied access) and violations are reported by the access controlled entry points.
<code>SOFT_PASSBACK</code>	Passback violations are checked but not enforced, but violations are reported by the access controlled entry points.

12.X.21 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.22 Entry_Points

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

12.X.23 Exit_Points

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

12.X.24 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.25 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.26 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.26.1 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 the same value 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.26.2 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.27 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.28 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.29 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.30 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.31 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 **13.2**, p. 255]

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**, p.256-257]

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

Object Type	Criteria	Event Type
...		
<i>Access Zone</i>	<i>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</i>	<i>CHANGE_OF_STATE</i>
...		

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

Object	Event Type	Notification Parameters	Referenced Object's Properties
...			
<i>Access Zone</i>	<i>CHANGE_OF_STATE</i>	<i>New_State</i> <i>Status_Flags</i>	<i>Occupancy_State</i> <i>Status_Flags</i>
....			

[Add new **BACnetAccessPassbackMode** production to Clause **21**, p. 408]

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

[Add new **BACnetAccessZoneOccupancyState** production to Clause **21**, p. 408]

```
BACnetAccessZoneOccupancyState ::= ENUMERATED {
    normal            (0),
    below-lower-limit (1),
    at-lower-limit    (2),
    at-upper-limit    (3),
    above-upper-limit (4),
    not-used          (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.

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetObjectTypesSupported** BITSTRING appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2004j-1.]

[Note: changes to **Clause 21, BACnetPropertyStates** production appear in Addendum 135-2004j-1]

[Note: changes to **Table 23-1 Extensible Enumerations** appear in Addendum 135-2004j-1.]

[Add new object type structure to **ANNEX C**, p.453]

```

ACCESS-ZONE ::= SEQUENCE {
    object-identifier      [75]    BACnetObjectIdentifier,
    object-name           [77]    CharacterString,
    object-type           [79]    BACnetObjectType,
    description           [28]    CharacterString OPTIONAL,
    occupancy-state       [298]   BACnetAccessZoneOccupancyState
    status-flags          [111]   BACnetStatusFlags,
    event-state           [36]    BACnetEventState,
    reliability           [103]   BACnetReliability OPTIONAL,
    out-of-service        [81]    BOOLEAN,
    occupancy-count       [293]   Unsigned,
    occupancy-count-enable [294]   BOOLEAN,
    adjust-value          [176]   INTEGER OPTIONAL,
    occupancy-upper-limit [299]   Unsigned OPTIONAL,
    occupancy-lower-limit [296]   Unsigned OPTIONAL,
    credentials-in-zone   [267]   SEQUENCE OF BACnetObjectIdentifier OPTIONAL,
    last-credential-added [280]   BACnetObjectIdentifier OPTIONAL,
    last-credential-added-time [281] BACnetDateTime OPTIONAL,
    last-credential-removed [282]  BACnetObjectIdentifier OPTIONAL,
    last-credential-removed-time [283] BACnetDateTime OPTIONAL,
    passback-mode         [302]   BACnetAccessPassbackMode OPTIONAL,
    passback-timeout      [303]   Unsigned OPTIONAL,
    entry-points          [269]   SEQUENCE OF BACnetDeviceObjectReference
    exit-points           [270]   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          [167]   CharacterString OPTIONAL,
}

```

[Add new **ANNEX D.X**, p.484]

D.X Example of an Access Zone object

In this example, a secured zone is represented as an Access Zone object. This Access Zone has an upper occupancy limit of 100 access credentials to be inside, and reports an alarm if the count is at or above the upper limit. `Occupancy_State` is `NORMAL` because two access credentials are inside. 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: 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)))

135-2004j-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 135-2004j-3

[Insert new clause **12.X** and Table **12-X**, p. 251]

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).

A specific user may be represented by a single Access User object in a single device or in multiple devices by one Access User object per device. However, it is a local matter as to how the 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
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 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.6 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.7 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.8 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.9 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.10 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.11 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.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 **BACnetAccessUserType** enumeration to Clause 21, p. 408]

```
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 135-2004j-1.]

[Note: changes to the **BACnetObjectTypesSupported** BITSTRING appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2004j-1.]

[Note: changes to **Table 23-1 Extensible Enumerations** appear in Addendum 135-2004j-1.]

[Add new object type structure to **ANNEX C**, p.453]

```
ACCESS-USER ::= SEQUENCE {  
    object-identifier    [75]  BACnetObjectIdentifier,  
    object-name          [77]  CharacterString,  
    object-type          [79]  BACnetObjectType,  
    description          [28]  CharacterString OPTIONAL,  
    user-type            [319] BACnetAccessUserType,  
    user-name            [318] CharacterString OPTIONAL,  
    user-external-identifier [311] CharacterString OPTIONAL,  
    user-information-reference [317] CharacterString OPTIONAL,  
    members              [289] SEQUENCE OF BACnetDeviceObjectReference OPTIONAL,  
    member-of            [159] SEQUENCE OF BACnetDeviceObjectReference OPTIONAL,  
    credentials          [266] SEQUENCE OF BACnetDeviceObjectReference,  
    profile-name         [167] CharacterString OPTIONAL  
}
```

[Add new **ANNEX D.X**, p.484]

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.

Property:	Object_Identifier =	(Access User, Instance 2)
Property:	Object_Name =	"Miller37"
Property:	Object_Type =	ACCESS_USER
Property:	Description =	"Dorothy H. Miller, CEO"
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))

135-2004j-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 135-2004j-4

[Insert new clause **12.X** and Table **12-X**, p. 251]

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.

A specific access rights collection may be represented by a single Access Rights object in a single device, or in multiple devices by one Access Rights object per device. However, it is a local matter as to how the 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
Enable	BOOLEAN	R
Negative_Access_Rules	BACnetARRAY[N] of BACnetAccessRule	R
Positive_Access_Rules	BACnetARRAY[N] of BACnetAccessRule	R
Accompanied	BACnetObjectIdentifier	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 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.6 Negative_Access_Rules

This property, of type BACnetARRAY[N] of BACnetAccessRule, specifies the negative access rules. Each element of the array is evaluated as per 12.X.6.1.

To determine how access rules are evaluated, see Clause 12.X.6.2.

12.X.6.1 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:
NEVER	The value of the Time-Range field is ignored and always evaluates to FALSE.
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 field, of type `BACnetObjectPropertyReference`, references a property that can be evaluated to `TRUE` or `FALSE`, which defines if the rule is valid (`TRUE`) or not (`FALSE`).

If `Time-Range-Specifier` is `NEVER` or `ALWAYS`, then this reference shall contain 4194303 in the instance part of the `Object-Identifier`.

If `Time-Range-Specifier` is `SPECIFIED`, then the following evaluation rules 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 its value cannot be retrieved, or the value 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:

<code>NONE</code>	The value of the <code>Location</code> field is ignored and does not match any access controlled point or zone.
<code>SPECIFIED</code>	Location references a specific <code>Access Point</code> or <code>Access Zone</code> object and is evaluated as specified for the <code>Location</code> field.
<code>ALL</code>	The value of the <code>Location</code> field is ignored and matches any access controlled point or zone.

Location This field, of type `BACnetObjectIdentifier`, refers to the `Access Point` or `Access Zone` that this access rule is valid for.

If `Location-Specifier` is `NONE` or `ALL`, then this reference shall contain 4194303 in the instance part of the `Object-Identifier`.

If `Location-Specifier` is `SPECIFIED`, then the following evaluation rules 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.

Enable This field, of type `BOOLEAN`, specifies whether this rule is enabled (`TRUE`) or not (`FALSE`).

An access rule evaluates to true when `Time_Range` evaluates to `TRUE` and `Location` matches the location of authentication.

12.X.6.2 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.

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

12.X.6.3 Initializing New Array Elements When the Array Size is Increased

If the size of the `Negative_Access_Rules` array is increased without entry values being provided, then the new array entries shall be initialized to contain `NEVER` for the `Time-Range-Specifier`, `NONE` for the `Location-Specifier` fields, and `TRUE` for the `Enable` field.

12.X.7 Positive_Access_Rules

This property, of type `BACnetARRAY[N]` of `BACnetAccessRule`, specifies the positive access rules. Each element of the array is evaluated as per 12.X.6.1.

To determine how access rules are evaluated, see Clause 12.X.6.2

12.X.7.1 Initializing New Array Elements When the Array Size is Increased

If the size of the `Positive_Access_Rules` array is increased without entry values being provided, then the new array entries shall be initialized to contain `NEVER` for the `Time-Range-Specifier`, `NONE` for the `Location-Specifier` fields, and `TRUE` for the `Enable` field.

12.X.8 Accompanied

This optional property, of type `BACnetObjectIdentifier`, specifies that the original credential is required to be accompanied by a second credential that meets the accompaniment criteria. It is a local matter as to whether the accompanying credential is required to be presented before or after the original credential. If this condition is not fulfilled, no access rules apply.

The accompaniment condition is 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 the instance part of this object reference has a value of 4194303, no accompaniment requirement is specified.

12.X.9 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. 408]

```

BACnetAccessRule ::= SEQUENCE {
    timeRangeSpecifier  [0] ENUMERATED {
        never            (0),
        specified        (1),
        always           (2)
    },
    timeRange           [1] BACnetObjectPropertyReference,
    locationSpecifier   [2] ENUMERATED {
        none            (0),
        specified        (1),
        all              (2)
    },
    location            [3] BACnetObjectIdentifier,
    enable              [4] BOOLEAN
}
    
```

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetObjectTypesSupported** BITSTRING appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2004j-1.]

[Add new object type structure to **ANNEX C**, p.453]

```

ACCESS-RIGHTS ::= SEQUENCE {
    object-identifier  [75] BACnetObjectIdentifier,
    object-name        [77] CharacterString,
    object-type        [79] BACnetObjectType,
    description        [28] CharacterString OPTIONAL,
    enable              [133] BOOLEAN,
}
    
```

```

negative-access-rules      [291] SEQUENCE OF BACnetAccessRule, -- accessed as a BACnetARRAY
positive-access-rules     [304] SEQUENCE OF BACnetAccessRule, -- accessed as a BACnetARRAY
accompanied                [254] BACnetObjectIdentifier OPTIONAL,
profile-name               [167] CharacterString OPTIONAL
}
    
```

[Add new ANNEX D.X, p.484]

D.X Examples of Access Rights objects

Assumed objects:	<u>Object_Identifier</u>	<u>Object_Name</u>	<u>Object_Type</u>
	(Access Point, Instance 1)	Main Production Area Door	ACCESS_POINT
	(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 (Access Point, Instance 7), for which access is never allowed. In addition, the main production area door (Access Point, Instance 1) may be accessed during times when the night shift hours schedule (Schedule, Instance 44) is active.

```

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: Enable = TRUE
Property: Negative_Access_Rules = (( ALWAYS, ((*, Instance 4194303), *),
                                     SPECIFIED, (Access Point, Instance 7), TRUE ))
Property: Positive_Access_Rules = (( SPECIFIED, ((Schedule, Instance 44), Present_Value),
                                     SPECIFIED, (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: Accompanied = (Access Credential, Instance 33)
    
```

135-2004j-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 associated with a credential used for authentication and its assigned access rights for 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 135-2004j-5

[Insert new Clause **12.X** and Table **12-X**, p. 251]

12.X Access Credential Object Type

The Access Credential object type defines a standardized object whose properties represent the externally visible characteristics associated with a credential used for authentication and for determining its assigned access rights for authorization when requesting access.

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

The Access Credential object is a container of related authentication factors. 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 will generate a corresponding TRACE access event.

A specific credential may be represented by a single Access Credential object in a single device or in multiple devices by one Access Credential object per device. However, it is a local matter as to how the 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
Credential_Status	BACnetBinaryPV	R
Reason_For_Disable	List of BACnetAccessCredentialDisableReason	R
Authentication_Factors	BACnetARRAY[N] of BACnetAuthenticationFactor	R
Activation_Time	BACnetDateTime	O
Expiry_Time	BACnetDateTime	O
Credential_Disable	BACnetAccessCredentialDisable	R
Days_Remaining	Unsigned	O
Uses_Remaining	Unsigned	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_Count_Exemption	BOOLEAN	O
Profile_Name	CharacterString	O

¹ If this property is present the property Last_Use_Time shall 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 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.

12.X.6 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 can 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 not valid any more at this time. 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 reached.
DISABLED_MAX_USES	The maximum number of uses for which this credential is valid for has been reached.
DISABLED_INACTIVITY	The credential has exceeded the allowed period of inactivity.
DISABLED_MANUAL	The credential is commanded to be disabled by a human operator.
DISABLED_LOST	The physical credential is reported to be lost.
DISABLED_STOLEN	The physical credential is reported to be stolen.
DISABLED_DAMAGED	The physical credential is reported to be damaged.
DISABLED_DESTROYED	The physical credential is reported to be destroyed.
<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.7 Authentication_Factors

This property, of type BACnetARRAY[N] of BACnetAuthenticationFactor, specifies the authentication factors which belong to this credential.

12.X.8 Activation_Time

This optional property, of type BACnetDateTime, specifies the date and time at or after which the credential is 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 contain a value of X'FF' or this property is not present, then the credential has an activation time of 'start of time'.

12.X.9 Expiry_Time

This optional property, of type BACnetDateTime, specifies the date and time after which the credential is expired. 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 contain a value of X'FF' or if this property is not present, then the credential has an expiry time of 'end-of-time'.

12.X.10 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_LOST	The credential is disabled because it has been reported to be lost. The disable-reason value DISABLED_LOST shall be added to the Reason_For_Disable property.
DISABLE_STOLEN	The credential is disabled because it has been reported to be stolen. The disable-reason value DISABLED_STOLEN shall be added to the Reason_For_Disable property.
DISABLE_DAMAGED	The credential is disabled because it has been damaged. The value DISABLED_DAMAGED shall be added to the Reason_For_Disable property.
DISABLE_DESTROYED	The credential is disabled because it has been destroyed. The value DISABLED_DESTROYED shall be added to the Reason_For_Disable property.
DISABLE_LOCKOUT	The credential is disabled because 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.11 Days_Remaining

This optional property, of type Unsigned, specifies the number of remaining days for which the credential can be used. The remaining days do not need to be contiguous days. If this property is present and has a value of zero, authentication is no longer possible using this credential. 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 list. The value DISABLED_MAX_DAYS shall be removed from the list when this condition no longer applies.

12.X.12 Uses_Remaining

This optional property, of type Unsigned, specifies the number of remaining uses which the credential can be used for authentication. Access granted at an access controlled point shall count as a single use. If this property is present and has a value of zero, the Access Credential shall be disabled and the value DISABLED_MAX_USES shall be added to the Reason_For_Disable list. The value DISABLED_MAX_USES shall be removed from the list when this condition no longer applies.

12.X.13 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 be present.

12.X.14 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 Object-Identifier field. 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.15 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 BACnetObjectIdentifier, refers to the Access Rights objects that define the access rights assigned to this credential. Each object referenced in this field shall be an Access Rights object. Any entry with an instance number of 4194303 or a reference to a non-existent Access Rights object shall be ignored.

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.15.1 Initializing New Array Elements When the Array Size is Increased

If the size of the Assigned_Access_Rights array is increased without entry values being provided, then the Assigned-Access-Rights field shall be initialized to contain an instance number of 4194303 and the Enable field shall be initialized to contain the value FALSE, for each new array entry.

12.X.16 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, this property shall contain 4194303 for the object instance.

12.X.17 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, this property shall have a value of NONE.

12.X.18 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, this property shall have the value X'FF' for all date and time octets.

12.X.19 Trace_Flag

This optional property, of type BOOLEAN, instructs, if TRUE, an Access Point object to generate a TRACE access event when this credential is used.

12.X.20 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.21 Extended_Time_Enable

This optional property, of type BOOLEAN, specifies which command of type BACnetDoorValue shall be used to unlock 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.22 Master_Exemption

This optional property, of type BOOLEAN, specifies a master exemption from authorization checks. Once authenticated, the credential 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.23 Passback_Exemption

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

12.X.24 Occupancy_Count_Exemption

This optional property, of type BOOLEAN, specifies an exemption from occupancy enforcement. If occupancy count exemption is enabled (TRUE), 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.25 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. 408]

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

[Add new **BACnetAccessCredentialDisable** enumeration to Clause 21, p. 408]

```
BACnetAccessCredentialDisable ::= ENUMERATED {  
    none                (0),  
    disable             (1),  
    disable-manual     (2),  
    disable-lost       (3),  
    disable-stolen     (4),  
    disable-damaged    (5),  
    disable-destroyed  (6),  
    disable-lockout    (7),  
    ...  
}  
-- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values  
-- 256-65535 may be used by others subject to the procedures and constraints described  
-- in Clause 23.
```

[Add new **BACnetAccessCredentialDisableReason** enumeration to Clause 21, p. 408]

```
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),
disabled-lost            (10),
disabled-stolen          (11),
disabled-damaged        (12),
disabled-destroyed      (13),
...
}
-- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
-- 256-65535 may be used by others subject to the procedures and constraints described
-- in Clause 23.

```

[Note: **BACnetAccessThreatLevel** is defined in Addendum 135-2004j-1.]

[Note: **BACnetAuthenticationFactor** is defined in Addendum 135-2004j-6.]

[Note: **BACnetAccessEvent** is defined in Addendum 135-2004j-1.]

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetObjectTypesSupported** BITSTRING appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2004j-1.]

[Note: changes to **Clause 21, BACnetPropertyStates** production appear in Addendum 135-2004j-1]

[Note: changes to **Table 23-1 Extensible Enumerations** appear in Addendum 135-2004j-1.]

[Add new object type structure to **ANNEX C**, p.453]

```

ACCESS-CREDENTIAL ::= SEQUENCE {
    object-identifier      [75]    BACnetObjectIdentifier,
    object-name            [77]    CharacterString,
    object-type            [79]    BACnetObjectType,
    description            [28]    CharacterString OPTIONAL,
    credential-status      [265]   BACnetBinaryPV,
    reason-for-disable     [306]   LIST OF BACnetAccessCredentialDisableReason
    authentication-factors [259]   SEQUENCE OF BACnetAuthenticationFactor,
                                -- accessed as a BACnetARRAY
    activation-time        [255]   BACnetDateTime OPTIONAL,
    expiry-time            [271]   BACnetDateTime OPTIONAL,
    credential-disable     [264]   BACnetAccessCredentialDisable,
    days-remaining         [268]   Unsigned OPTIONAL,
    uses-remaining         [320]   Unsigned OPTIONAL,
    absentee-limit        [244]   Unsigned OPTIONAL,
    belongs-to             [263]   BACnetDeviceObjectReference OPTIONAL,
    assigned-access-rights [257]   SEQUENCE OF BACnetAssignedAccessRights,
                                --accessed as a BACnetARRAY
    last-access-point      [279]   BACnetDeviceObjectReference OPTIONAL,
    last-access-event      [278]   BACnetAccessEvent OPTIONAL,
    last-use-time          [284]   BACnetDateTime OPTIONAL,
    trace-flag             [309]   BOOLEAN OPTIONAL,
    threat-authority       [307]   BACnetAccessThreatLevel OPTIONAL,
    extended-time-enable   [272]   BOOLEAN OPTIONAL,
    master-exemption       [287]   BOOLEAN OPTIONAL;
    passback-exemption     [301]   BOOLEAN OPTIONAL,
    occupancy-count-exemption [295]  BOOLEAN OPTIONAL,

```

```

profile-name          [167]  CharacterString OPTIONAL
}
    
```

[Add new ANNEX D.X, p.484]

D.X Example of an Access Credential object

This example of a 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')
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 Access Rights object instance 1 and 77 assigned to it. The credential also has the 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.

```

Property:  Object_Identifier =      (Access Credential, Instance 33)
Property:  Object_Name =           "Manager Credential 33"
Property:  Object_Type =           ACCESS_CREDENTIAL
Property:  Credential_Status =     ACTIVE
Property:  Reason_For_Disable =    ( )
Property:  Authentication_Factors = ( (WIEGAND26, 0, X'83004D'),
                                     (FASC_N, 0, X'254E04D20001E240') )
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:  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_Count_Exemption = FALSE
    
```

135-2004j-6. Add a new Authentication Factor Input object type.

Rationale

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

Addendum 135-2004j-6

[Insert new clause **12.X** and Table **12-X**, p. 251]

12.X Authentication Factor Input Object Type

The Authentication Factor Input object type defines a standardized object whose properties represent the externally visible characteristics of an authentication factor input. Examples of authentication factor input devices are card readers, keypads, biometric readers, etc. An authentication factor is a unique digital identifier that is used to verify the identity of a credential.

The Authentication Factor 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 Authentication Factor 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	O
Out_Of_Service	BOOLEAN	R
Format_Type	BACnetAuthenticationFactorType	R
Format_Class_Supported	Unsigned	O
Read_Status	BACnetAuthenticationFactorReadStatus	R
Update_Time	BACnetTimeStamp	R
Vendor_Identifier	Unsigned16	O ²
Vendor_Format_Identifier	Unsigned	O ²
Profile_Name	CharacterString	O

¹ This property is required to be writable when Out_Of_Service is TRUE.

² These properties are required if the choice PROPRIETARY is used for the 'Value' field of Present_Value.

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 AUTHENTICATION_FACTOR_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 internal representation of the authentication factor value in the Value field. The value of this field must be equal to the Format_Type property. If there is no current value read by this input, this field shall take on the value UNDEFINED. If an authentication factor is read that contains errors or that cannot be interpreted as the specified type, this field shall take on the value ERROR.
Format-Class	This field, of type Unsigned, is a site specific value that identifies the class of authentication factors that can be read by this input. If the optional property Format_Class_Supported exists, then this value shall correspond to the value of that property. This field is used in sites where different formats of authentication factors are used that have the same authentication factor format type. The format class value is used to differentiate between the different formats. A value of zero is used as the default where no differentiation of authentication factors is required.
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 X-1 of Annex X.

The Present_Value property shall be writable when Out_Of_Service is TRUE. When the object is not in service the allowable values that can be written to the Format-Class and Value fields shall still comply with the restrictions defined in the corresponding definitions.

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 Authentication Factor 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 property is present and does not have a value of 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

This optional 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}.

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 Authentication Factor Input object is prevented from being modified by software local to the BACnet device in which the object resides. When Out_Of_Service is TRUE, the Present_Value property may be written to.

12.X.9 Format_Type

This property, of type BACnetAuthenticationFactorType, is an enumeration that determines the type of authentication factor returned in the Value field of the Present_Value. When this value is written, it shall cause the Format-Type field of the Present_Value property to take the value UNDEFINED and Read_Status to take the value NOT_INITIALIZED.

12.X.10 Format_Class_Supported

This optional property, of type Unsigned, specifies the value used in the Format-Class field of the Present-Value property. This property is used in sites where different formats of authentication factors are used that have the same authentication factor format type. The format class value is used to differentiate between the different formats. If this property exists, a value of zero is used as default where no differentiation is required. Otherwise, the value is site specific and can be any non-zero value. When this value is changed, it shall cause the Format-Type field of the Present_Value property to take the value UNDEFINED and Read_Status to take on the value NOT_INITIALIZED.

12.X.11 Read_Status

This required property of type BACnetAuthenticationFactorReadStatus, indicates the status of the value of the Present_Value property. The values of this enumeration are as follows:

NOT_INITIALIZED	The Present_Value property does not yet contain a valid or erroneous reading. The Format-Type field of the Present_Value has a value of UNDEFINED.
VALID	The Present_Value contains a valid authentication factor.
ERROR	The Present_Value contains an erroneous reading. The Format-Type field of the Present_Value has a value of ERROR and may provide the erroneous raw authentication factor in the Value field.

12.X.12 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. 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.13 Vendor_Identifier

This optional property, of type Unsigned16, holds the vendor identifier of the vendor defining the custom authentication factor format. This property shall be present if a custom authentication format is supported. This value may differ from the Vendor_Identifier property value in the Device object, which indicates the device manufacturer.

12.X.14 Vendor_Format_Identifier

This optional property of type Unsigned holds the vendor's custom authentication format identifier. This property shall be present if a custom authentication format is supported.

12.X.15 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 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 Table 13-1, p. 254]

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

Object Type	Criteria	Properties Reported
...
<i>Authentication Factor Input</i>	<i>If Update_Time changes at all</i>	<i>Present_Value, Status_Flags, Update_Time, Read_Status</i>
...		

[Add new **BACnetAuthenticationFactor** production to Clause 21, p. 408]

```

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

[Add new **BACnetAuthenticationFactorReadStatus** enumeration to Clause 21, p. 408]

```

BACnetAuthenticationFactorReadStatus ::= ENUMERATED {
    notInitialized    (0),
    valid              (1),
    error              (2)
}
    
```

[Add new **BACnetAuthenticationFactorType** enumeration to Clause 21, p. 408]

```
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),  
    card32-facility16-number (9),  
    card32-facility32-number (10),  
    fasc-n            (11),  
    fasc-n-bcd        (12),  
    fasc-n-large      (13),  
    fasc-n-large-bcd (14),  
    gsa75             (15),  
    chuid             (16),  
    chuid-full        (17),  
    guid              (18),  
    cbeff-A           (19),  
    cbeff-B           (20),  
    cbeff-C           (21),  
    user-password     (22),  
}
```

[Note: changes to the **BACnetObjectType** enumeration appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetObjectTypesSupported** BITSTRING appear in Addendum 135-2004j-1.]

[Note: changes to the **BACnetPropertyIdentifier** enumeration appear in Addendum 135-2004j-1.]

[Note: changes to **Clause 21, BACnetPropertyStates** production appear in Addendum 135-2004j-1]

[Add new object type structure to **ANNEX C**, p.453]

```
AUTHENTICATION-FACTOR-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,  
    format-type       [277] BACnetAuthenticationFactorType,  
    format-class-supported [276] Unsigned OPTIONAL,  
    read-status       [305] BACnetAuthenticationFactorReadStatus,  
    update-time       [189] BACnetTimeStamp,  
    vendor-identifier [120] Unsigned16 OPTIONAL,  
    vendor-format-identifier [321] Unsigned OPTIONAL,  
    profile-name      [168] CharacterString OPTIONAL  
}
```

[Note: changes to the **BACnetReliability** enumeration for COMMUNICATION_FAILURE appear in Addendum 135-2004i-1.]

[Add new **ANNEX D.X**, p.484]

D.X Example of an Authentication Factor Input object

This example of an authentication factor input object represents a card reader on the unsecure side of the door. The format type for this reader is facility-and-card-number and the specific format read is 26 bit Wiegand. A valid card was recently read with a facility code of 121 (X'79') and a card number of 20926 (X'51BE'). The optional Vendor_Identifier and Vendor_Format_Identifier properties are not supported.

Property:	Object_Identifier =	(Authentication Factor Input, Instance 1)
Property:	Object_Name =	"Main Entrance Card Reader"
Property:	Object_Type =	AUTHENTICATION_FACTOR_INPUT
Property:	Present_Value =	(WIEGAND26, 0, X'7951BE')
Property:	Description =	"Main entrance south building card reader – 26 bit Wiegand"
Property:	Status_Flags =	(FALSE, FALSE, FALSE, FALSE)
Property:	Reliability =	NO_FAULT_DETECTED
Property:	Out_Of_Service =	FALSE
Property:	Format_Type =	WIEGAND26
Property:	Format_Class_Supported =	0
Property:	Read_Status =	VALID
Property:	Update_Time =	((7 JAN 2008, MONDAY), 15:30:51.70)

135-2004j-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 135-2004j-7

[Change 12.12.5 and Table 12-15, Event Enrollment object, pp. 185-186]

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, 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
...
CHANGE_OF_LIFE_SAFETY	NORMAL OFFNORMAL LIFE_SAFETY_ALARM	Time_Delay List_Of_Alarm_Values List_Of_Life_Safety_Alarm_Values Mode_Property_Reference
ACCESS_EVENT	NORMAL	List_Of_Access_Events Access_Event_Time_Reference

[Change 12.12.7, pp. 186-188]

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 that provide the last update time of the referenced property that is monitored for access events (normally the *Access_Event_Time* property). This parameter may only reference object properties that are of type *BACnetTimeStamp*.

[Change 13.2 and Table 13-2 through Table 3-4, pp.255-257]

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
...
Accumulator	If Pulse_Rate exceeds range from Low_Limit through High_Limit for longer than Time_Delay AND the new transition is enabled in Event_Enable and Limit_Enable, OR Pulse_Rate returns to range from Low_Limit through High_Limit for longer than Time_Delay AND the new transition is enabled in Event_Enable and Limit_Enable	UNSIGNED_RANGE
Access Point	<i>(Access Alarm Events) If Access_Event_Time changes and Access_Event is equal to one of the values in the Access_Alarm_Events list</i>	ACCESS_EVENT
Access Point	<i>(Access Transaction Events) If Access_Event_Time changes and Access_Event is equal to one of the values in the Access_Transaction_Events list</i>	ACCESS_EVENT

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

Object	Event Type	Notification Parameters	Referenced Object's Properties
...
Accumulator	UNSIGNED_RANGE	Exceeding_Value Status_Flags Exceeded_Limit	Pulse_Rate Status_Flags Low_Limit or High_Limit
Access Point	ACCESS_EVENT	Access_Event Status_Flags Access_Event_Time Access_Credential Authentication_Factor (if present)	Access_Event Status_Flags 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
...
UNSIGNED_RANGE	Exceeding_Value Status_Flags Exceeded_Limit	The value that exceeded a limit The Status_Flags of the referenced object The limit that was exceeded
ACCESS_EVENT	Access_Event Status_Flags Access_Event_Time Access_Credential Authentication_Factor (if present)	The new value of the referenced property The Status_Flags 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 13.3, p. 258]

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** and **Figure 13-X**, renumbering subsequent figures, p. 264]

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 only be applied to properties that are of type `BACnetAccessEvent`. For the purposes of event notification, `ACCESS_EVENT` events generate a `TO-NORMAL` transition.

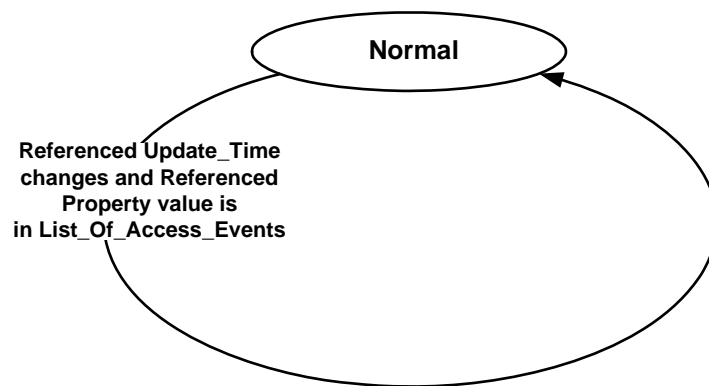


Figure 13-X. `ACCESS_EVENT` Algorithm

[Change `BACnetEventParameter` production, clause **21**, pp. 415-416]
 [Note: context tag 12 is used in Addendum 135-2004b-2.]

```

BACnetEventParameter ::= CHOICE {
  ...
  access-event      [13] SEQUENCE {
    list-of-access-events      [0] SEQUENCE OF BACnetAccessEvent,
    update-time-reference     [1] BACnetDeviceObjectPropertyReference
  }
}
  
```

[Change `BACnetEventType` enumeration, clause **21**, p. 417]
 [Note: enumeration 12 is used in Addendum 135-2004b-2.]

```

BACnetEventType ::= ENUMERATED {
  ...
  unsigned-range      (11),
  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. 419-420]

[Note: context tag 12 is used in Addendum 135-2004b-2.]

[Note: This could be due to an alphabetization issue in this table... is defined in Addendum 135-2004j-1.]

[Note: **BACnetAuthenticationFactor** is defined in Addendum 135-2004j-6.]

BACnetNotificationParameters ::= CHOICE {

- These choices have a one-to-one correspondence with the Event_Type enumeration with the exception of the
- complex-event-type, which is used for proprietary event types.

...

access-event [13] SEQUENCE {

access-event [0] *BACnetAccessEvent*,
status-flags [1] *BACnetStatusFlags*,
update-time [2] *BACnetTimeStamp*,
access-credential [3] *BACnetObjectIdentifier*,
authentication-factor [4] *BACnetAuthenticationFactor* OPTIONAL
}

}

135-2004j-8. Add a new Annex X 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 135-2004j-8

ANNEX X – 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 be emerging 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 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 value formats that have the same format type.

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

Table X-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
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

		<p>4 = Incomplete read</p> <p>128..255 = Any proprietary error reason</p> <p>Octet[2..3] = authentication factor format type expected</p> <p>Octet[4..n] = data array that holds erroneous data</p>
CUSTOM	<p>Custom (proprietary, or industry standard) format – each format specified is identified by the vendor ID and the proprietary format ID</p>	<p>Octet String Size = n</p> <p>Octet[1..2] = BACnet vendor-id (i.e., unsigned 16)</p> <p>Octet[3..4] = proprietary type id (i.e., unsigned 16)</p> <p>Octet[5..n] = data array that holds proprietary format</p>
SIMPLE_NUMBER16	<p>Simple unsigned number with range [0 .. 65535]</p>	<p>Octet String Size = 2,</p> <p>Octet[1..2] = number (i.e., unsigned 16 bit number)</p>
SIMPLE_NUMBER32	<p>Simple unsigned number with range [0 .. 4294967295]</p>	<p>Octet String Size = 4,</p> <p>Octet[1..4] = number (i.e., unsigned 32 bit number)</p>
SIMPLE_NUMBER56	<p>Simple unsigned number with range [0 .. 72057594037927935]</p> <p>Typically used for DESFire card Serial Numbers</p>	<p>Octet String Size = 7,</p> <p>Octet[1..7] = number (i.e., unsigned 56 bit number)</p>
SIMPLE_ALPHA_NUMERIC	<p>Simple alpha numeric string</p>	<p>Octet String Size = n,</p> <p>Octet[1] = length of character string in octets (max 255)</p> <p>Octet[2] = character set (as specified in 20.2.9)</p> <p>Octet[3..n] = string of characters (encoded as specified in 20.2.9)</p>
ABA_TRACK2	<p>Magnetic stripped card format (BCD² format) as developed by the banking industry (ABA).</p>	<p>Octet String Size = 15,</p> <p>Octet[1.. 10 (MS nibble)] = primary account number (19 digits)</p> <p>Octet[10 (LS nibble) – 12(MS nibble)] = 4 digit expiration date in form “MMYY”</p> <p>Octet[12 (LS nibble)..13] = 3 digit service code</p> <p>Octet[14.. 15] = discretionary data (4 digits)</p>
WIEGAND26	<p>Standard 26 bit Wiegand format as defined by SIA standard separated into facility code and card number.</p>	<p>Octet String Size = 3</p> <p>Octet[1] = facility-code (i.e., unsigned 8 bit number)</p>

		Octet[2..3] = card-number (i.e., unsigned 16 bit number)
CARD16_FACILITY32_NUMBER	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)
CARD32_FACILITY32_NUMBER	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)
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)

		<p>Octet[17..18] = organizational identifier (i.e., unsigned 16 bit number)</p> <p>Octet[19] = association category (i.e., unsigned 8 bit number)</p> <p>-- refer to NIST technical implementation Guidance document for more details</p>
FASC_N_LARGE_BCD	<p>Federal Agency Smart Credential – number. (BCD² format)</p> <p>Includes all FASC-N data fields excluding start sentinel, end sentinel, field separators and LRC.</p>	<p>Octet String Size = 16</p> <p>Octet[1..2] = agency-code (4 digit BCD number)</p> <p>Octet[3..4] = system-site code (4 digit BCD number)</p> <p>Octet[5..7] = credential number (6 digit BCD number)</p> <p>Octet[8 (MS nibble)] = series code (1 digit BCD number)</p> <p>Octet[8 (LS nibble)] = credential code (1 digit BCD number)</p> <p>Octet[9..13] = credential number (10 digit BCD number)</p> <p>Octet[14 (MS nibble)] = organizational category (1 digit BCD number)</p> <p>Octet[14 (LS nibble)..16(MS nibble)] = organizational identifier (4 digit BCD number)</p> <p>Octet[16 (LS nibble)] = association category (1 digit BCD number)</p> <p>-- refer to NIST technical implementation Guidance document for more details</p>
GSA75	GSA 75 bit (FASC-N plus expiry date)	<p>Octet String Size = 12</p> <p>Octet[1..2] = agency-code (i.e., unsigned 16 bit number)</p> <p>Octet[3..4] = system-site code (i.e., unsigned 16 bit number)</p> <p>Octet[5..8] = credential number (i.e., unsigned 32 bit number)</p> <p>Octet[9..12] = expiry date (4 octets encoded as specified in Clause 20.2.12)</p>
GUID	Global unique identifier represented as IPv6 address	<p>Octet String Size = 16</p> <p>-- Refer to RFC 2373 for format description and encoding</p>
CHUID	Card Holder Unique Identifier (CHUID), without Asymmetric Key and without Authentication	<p>Octet String Size = 45</p> <p>Octet[1..8] = FASC-N as specified in FASC_N</p>

	<p>Key MAP.</p> <p>See SP 800-73 Section 1.8.3 (Figure 1 & 2 pg 12 of the TIG 2.3)</p>	<p>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)</p> <p>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)</p> <p>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)</p> <p>Octet[26..41] = GUID (IPv6 address as defined in SP 800-73 (Figures 1 & 2 of the TIG 2.3)</p> <p>Octet[42..45] = Expiry Date expiry date (4 octets encoded as specified in Clause 20.2.12)</p>
CHUID_FULL	<p>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.</p> <p>See SP 800-73 Section 1.8.3 (Figure 1 & 2 pg 12 of the TIG 2.3)</p>	<p>Octet String Size = n (maximum size = 3397)</p> <p>Octet[1..n] = CHUID data string</p> <p>-- Octet encoding is defined in SP 800-73 (Figure 1 & 2 of the TIG 2.3) using CHUID Tags.</p>
CBEFF_A	<p>Common Biometric Exchange File Format (CBEFF) Patron format A</p>	<p>Octet String Size = n</p> <p>Octet[1..n] = CBEFF data</p> <p>-- NIST CBEFF Patron Format A (CBEFF) content formatted</p>
CBEFF_B	<p>Common Biometric Exchange File Format (CBEFF) Patron format B</p>	<p>Octet String Size = n</p> <p>Octet[1..n] = CBEFF data</p> <p>-- NIST CBEFF Patron Format B (BioAPI) content formatted</p>
CBEFF_C	<p>Common Biometric Exchange File Format (CBEFF) Patron format C</p>	<p>Octet String Size = n</p> <p>Octet[1..n] = CBEFF data</p> <p>-- NIST CBEFF Patron Format C (ANSI Standard X9.84) content formatted</p>
USER_PASSWORD	<p>User name and password</p>	<p>Octet String Size = n,</p> <p>Octet[1] = length of user name string in octets (max 255)</p> <p>Octet[2] = character set for user name string (as specified in 20.2.9)</p> <p>Octet[3..m] = string of characters for user name(encoded</p>

		<p>as specified in Clause 20.2.9)</p> <p>Octet[m+1] = length of password string in octets (max 255)</p> <p>Octet[m+2] = character set password string (as specified in 20.2.9)</p> <p>Octet[m+3..n] = string of characters for password (encoded as specified in Clause 20.2.9)</p>
--	--	---

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.