BSR/ASHRAE Addendum *a*
to ANSI/ASHRAE Standard 135-2001

This supplement will be submitted to the American National Standards Institute Board of Standards Review (BSR) for approval.

ASHRAE® STANDARD

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

SECOND PUBLIC REVIEW
(Independent Substantives Changes to First Public Review)

FEBRUARY 2003

This draft has been recommended for public review by the responsible project committee. Public review of this proposed addendum has been authorized by a subcommittee of the Standards Committee. Until final approval by the ASHRAE Board of Directors, this draft addendum is subject to modification and Standard 135-2001 remains in effect. Instructions and a form for commenting are provided with this draft. Although reproduction of drafts during the public review period is encouraged to promote additional comment, permission must be obtained to reproduce all or any part of this document from the ASHRAE Manager of Standards, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. Phone: 404-636-8400, Ext. 502. Fax: 404-321-5478. E-mail: cramspeck@ashrae.org

The parent standard, not including this proposed change, is under continuous maintenance. The change submittal form, instructions and deadlines may be obtained in electronic form from ASHRAE's Internet Home Page, *http://www/ashrae.org*, or in paper form from the Manager of Standards. The latest edition of an ASHRAE Standard may be purchased from ASHRAE Customer Service, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. E-mail: *orders@ashrae.org*. Fax: 404-321-5478. Telephone: 404-636-8400 (worldwide), or toll free 1-800-527-4723 (for orders in U.S. and Canada).

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

(This foreword is not part of the standard but is provided for information only.)

FOREWORD

Foreword to the Second Public Review Draft of Addendum 135*a* to ANSI/ASHRAE Standard 135-2001

The second public review draft of Addendum 135*a* to ANSI/ASHRAE Standard 135-2001 contains a number of independent substantive changes to the first public review draft. The clauses containing these changes are included in this document.

Comments on clauses outside the scope of this review may be submitted but will be considered solely at the discretion of the project committee. Comments pointing out typographical or editorial errors, regardless of location in the proposed addendum, are always welcome.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135-2001 and Addenda is indicated through the use of *italics* while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout.

135*a*-1. Add Partial Day Scheduling to the Schedule object.

Rationale
Events that are not day-long are not easily configured in the Schedule object; the list of actions normally scheduled for the day must be copied and then altered for that specific day to accommodate the event. More seriously, if the effective dates for two or more exception schedules only partially overlap, as is common with wild card dates, then correct full-day schedules cannot be created for these exceptions, and alternate, non-interoperable methods must be implemented to achieve the desired scheduling results. In addition, if a device containing a Schedule object is started on a day when there are no scheduled actions, or if there are no schedules at all in the object, there is no way to determine what value should be presented in Present_Value. Specifying new behavior for evaluation of schedules and adding the Schedule_Default property to the Schedule object resolve these situations.

There is also currently no way to override the internal calculations of the Schedule object, which may be desirable for manual operations. To remedy this, an Out_Of_Service property is added to the Schedule object, which behaves similarly to the Out_Of_Service property in other standard objects.

Additionally, there is currently no way for the Schedule object to indicate that it has been configured incorrectly. The Schedule object has several interrelated properties that all need to be configured consistently for proper operation of the object. New Status_Flags and Reliability properties are added to allow the Schedule object to indicate its “health” in a standard and network-visible way.

Addendum 135*a*-1

[Add to Clause 12 introduction, p. 129]

…

MULTI_STATE_FAULT	The Present_Value of the Multi-state object is equal to one of the states in the Fault_Values property and no other fault has been detected.
CONFIGURATION_ERROR	*The object’s properties are not in a consistent state.*

[Change **12.22**, p.224]

12.22 Schedule Object Type

The Schedule object type defines a standardized object used to describe a periodic schedule that may recur during a range of dates, with optional exceptions ~~on arbitrary dates~~ *at arbitrary times on arbitrary dates*. The Schedule object also serves as a binding between these scheduled times and the writing of specified "values" to specific properties of specific objects at those times. The Schedule object type and its properties are summarized in Table 12-26 and described in detail in this subclause.

Schedules are divided into days, of which there are two types: normal days within a week and exception days. *Both types of days can specify scheduling events for either the full day or portions of a day, and a priority mechanism defines which scheduled event is in control at any given time.*

The current state of the Schedule object is represented by the value of its Present_Value property, which is normally calculated using the time/value pairs from the Weekly_Schedule and Exception_Schedule properties, with a default value for use when no schedules are in effect. Details of this calculation are provided in 12.22.4.

~~It is assumed that the scheduler will exhibit restorative behavior in the event that the BACnet Device containing the schedule is restarted or the time is changed in the BACnet Device. The model for restoration assumes that each day's~~

~~schedule is circular in nature. Thus, if the BACnet Device is restarted after midnight but prior to the first time in the list of BACnetTimeValues for that day, then the last value on the list for that day is used as the restoration value. If some other value is desired, then an explicit time of 00:00 shall be the first entry in the list.~~

Versions of the Schedule object prior to Protocol_Revision <N> only support schedules that define an entire day, from midnight to midnight. For compatibility with these versions, this whole day behavior can be achieved by using a specific schedule format. Weekly_Schedule and Exception_Schedule values that begin at 00:00, and do not use any NULL values, will define schedules for the entire day. Property values in this format will produce the same results in all versions of the Schedule object.

Table 12-26. Properties of the Schedule Object Type

Property Identifier	Property Datatype	Conformance Code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	Any	R
Description	CharacterString	O
Effective_Period	BACnetDateRange	R
Weekly_Schedule	BACnetARRAY[7] of BACnetDailySchedule	O[1]
Exception_Schedule	BACnetARRAY[N] of BACnetSpecialEvent	O[1]
Schedule_Default	*Any*	*R*
List_Of_Object_Property_References	List of BACnetDeviceObjectPropertyReference	R
Priority_For_Writing	Unsigned (1..16)	R
Status_Flags	*BACnetStatusFlags*	*R*
Reliability	*BACnetReliability*	*R*
Out_Of_Service	*BOOLEAN*	*R*
Profile_Name	CharacterString	O

[1] At least one of these properties is required.

[Change **12.22.4**, p.224]

12.22.4 Present_Value

This property indicates the current value of the schedule, ~~i.e. the value most recently written to a referenced property of one member of List_Of_Object_Property_References.~~ ~~This~~ *which* may be any primitive datatype. As a result, *most* analog, binary and enumerated values may be scheduled. ~~If the List_Of_Object_Property_References is empty, then the value of this property will be that which would have been most recently written to the List_Of_Object_Property_References.~~ *This property shall be writable when Out_Of_Service is TRUE (see 12.22.11).*

Any change in the value of this property shall be written to all members of the List_Of_Object_Property_References property. An error writing to any member of the list shall not stop the Schedule object from writing to the remaining members.

The normal calculation of the value of the Present_Value property is illustrated as follows (the actual algorithm used is a local matter but must yield the same results as this one):

1. *Find the highest relative priority (as defined by 12.22.8) Exception_Schedule array element that is in effect for the current day and whose current value (see method below) is not NULL, and assign that value to the Present_Value property.*
2. *If the Present_Value was not assigned in the previous step, then evaluate the current value of the Weekly_Schedule array element for the current day and if that value is not NULL, assign it to the Present_Value property.*

3. If the Present_Value was not assigned in the previous steps, then assign the value of the Schedule_Default property to the Present_Value property.

The method for evaluating the current value of a schedule (either exception or weekly) is to find the latest element in the list of BACnetTimeValues that occurs on or before the current time, and then use that element's value as the current value for the schedule. If no such element is found, then the current value for the schedule shall be NULL.

These calculations are such that they can be performed at any time and the correct value of Present_Value property will result. These calculations must be performed at 00:00 each day, whenever the device resets, whenever properties that can affect the results are changed, whenever the time in the device changes by an amount that may have an effect on the calculation result, and at other times, as required, to maintain the correct value of the Present_Value property through the normal passage of time.

Note that the Present_Value property will be assigned the value of the Schedule_Default property at 00:00 of any given day, unless there is an entry for 00:00 in effect for that day. If a scheduled event logically begins on one day and ends on another, an entry at 00:00 shall be placed in the schedule that is in effect for the second day, and for any subsequent days of the event's duration, to ensure the correct result whenever Present_Value is calculated.

[Change clause **12.22.6**, p.224]

12.22.6 Effective_Period

This property specifies the range of dates within which the Schedule object is active. Seasonal scheduling may be achieved by defining several SCHEDULE objects with non-overlapping Effective_Periods to control the same property references. *Upon entering its effective period, the object shall calculate its Present_Value and write that value to all members of the List_Of_Object_Property_References property. An error writing to any member of the list shall not stop the Schedule object from writing to the remaining members.*

[Change **12.22.7**, p.225]

12.22.7 Weekly_Schedule

This property is a BACnetARRAY containing exactly seven elements. Each of the elements 1-7 contains a BACnetDailySchedule. A BACnetDailySchedule consists of a list of BACnetTimeValues that are (time, value) pairs, which describe the sequence of schedule actions on one day of the week when no Exception_Schedule is in effect. The array elements 1-7 correspond to the days Monday - Sunday, respectively. The Weekly_Schedule is an optional property, but either the Weekly_Schedule or a non-empty Exception_Schedule shall be supported in every instance of a Schedule object.

If the Weekly_Schedule property is written with a schedule item containing a datatype not supported by this instance of the Schedule object (e.g., the List_Of_Object_Property_References property cannot be configured to reference a property of the unsupported datatype), the device may return a Result(-) response, specifying an 'Error Class' of PROPERTY and an 'Error Code' of DATATYPE_NOT_SUPPORTED.

[Change **12.22.8**, p.225]

12.22.8 Exception_Schedule

This property is a BACnetARRAY of BACnetSpecialEvents. Each BACnetSpecialEvent describes a sequence of schedule actions that takes precedence over the normal day's behavior on a specific day or days.

BACnetSpecialEvent ::= (Period, List of BACnetTimeValue, EventPriority)

Period ::= Choice of {BACnetCalendarEntry | CalendarReference}

EventPriority ::= Unsigned (1..16)

The Period may be a BACnetCalendarEntry or it may refer to a Calendar object as described in 12.8. A BACnetCalendarEntry would be used if the Exception_Schedule is specific to this Schedule object, while calendars might be defined for common holidays to be referenced by multiple Schedule objects. Each BACnetCalendarEntry is either an individual date (Date), range of dates (BACnetDateRange), or month/week-of-month/day-of-week specification (BACnetWeekNDay). If the current date matches any of the calendar entry criteria, the Exception Schedule would be activated and the list of BACnetTimeValues would be enabled for use.

Individual fields of the various constructs of the BACnetCalendarEntry may also have a "wildcard" value used for determining whether the current date falls within the Period of the Exception Schedule. In a date range, for example, if the startDate is a wildcard, it means "any date up to and including the endDate." If the endDate is a wildcard, it means "any date from the startDate on." If the calendar entry were a BACnetWeekNDay with wildcard for month and week-of-month fields but with a specific day-of-week, it would mean that the Exception Schedule would apply on that day-of-week all year long.

Each BACnetSpecialEvent contains an EventPriority that determines its importance relative to other BACnetSpecialEvents within the same Exception_Schedule. Since SpecialEvents within the same Exception_Schedule may have overlapping periods, it is necessary to have a mechanism to determine ~~which SpecialEvent applies~~ *the relative priorities for the SpecialEvents that apply* on any given day. If more than one SpecialEvent applies to a given day, the *relative priority of the SpecialEvents shall be determined by their* ~~SpecialEvent with the highest~~ EventPriority *values* ~~shall have higher priority for evaluation take effect and all other applicable SpecialEvents shall be ignored for that day~~. If multiple overlapping SpecialEvents have the same ~~priority,~~ *EventPriority value*, then the SpecialEvent ~~occurring earliest~~ *with the lowest index number* in the array shall ~~take effect for that day~~ *have higher relative priority*. The highest EventPriority is 1 and the lowest is 16. The EventPriority is not related to the Priority_For_Writing property of the Schedule object.

If a BACnet Device supports writing to the Exception_Schedule property, all possible choices in the BACnetSpecialEvents shall be supported.

If the Exception_Schedule property is written with a schedule item containing a datatype not supported by this instance of the Schedule object (e.g., the List_Of_Object_Property_References property cannot be configured to reference a property of the unsupported datatype), the device may return a Result(-) response, specifying an 'Error Class' of PROPERTY and an 'Error Code' of DATATYPE_NOT_SUPPORTED.

[Renumber clause **12.22.9**, List_Of_Object_Property_References, to **12.22.10;**
Renumber clause **12.22.10**, Priority_For_Writing, to **12.22.11;**
Renumber clause **12.22.11**, Profile_Name, to **12.22.15**]

[Insert new clause **12.22.9**, p.225]

12.22.9 Schedule_Default

This property holds a default value to be used for the Present_Value property when no other scheduled value is in effect (see 12.22.4). This may be any primitive datatype.

If the Schedule_Default property is written with a value containing a datatype not supported by this instance of the Schedule object (e.g., the List_Of_Object_Property_References property cannot be configured to reference a property of the unsupported datatype), the device may return a Result(-) response, specifying an 'Error Class' of PROPERTY and an 'Error Code' of DATATYPE_NOT_SUPPORTED.

[Insert new clause **12.22.12**, p.226]

12.22.12 Status_Flags

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

{IN_ALARM, FAULT, OVERRIDDEN, OUT_OF_SERVICE}

where:

IN_ALARM	The value of this flag shall be logical FALSE (0).
FAULT	Logical TRUE (1) if the Reliability property (12.22.13) does not have a value of NO_FAULT_DETECTED, otherwise logical FALSE (0).
OVERRIDDEN	Logical TRUE (1) if the schedule object has been overridden by some mechanism local to the BACnet Device. In this context "overridden" is taken to mean that the Present_Value property is not changeable through BACnet services. Otherwise, the value is logical FALSE (0).
OUT_OF_SERVICE	Logical TRUE (1) if the Out_Of_Service property (12.22.14) has a value of TRUE, otherwise logical FALSE (0).

[Insert new clause **12.22.13**, p.226]

12.22.13 Reliability

The Reliability property, of type BACnetReliability, provides an indication that the properties of the schedule object are in a consistent state. All non-NULL values used in the Weekly_Schedule, the Exception_Schedule, and the Schedule_Default properties shall be of the same datatype, and all members of the List_Of_Object_Property_References shall be writable with that datatype. If these conditions are not met, then this property shall have the value CONFIGURATION_ERROR. The Reliability property for this object type may have any of the following values:

{NO_FAULT_DETECTED, CONFIGURATION_ERROR, UNRELIABLE_OTHER}.

If the List_Of_Object_Property_References contains a member that references a property in a remote device, the detection of a configuration error may be delayed until an attempt is made to write a scheduled value.

[Insert new clause **12.22.14**, p.226]

12.22.14 Out_Of_Service

The Out_Of_Service property, of type BOOLEAN, is an indication whether (TRUE) or not (FALSE) the internal calculations of the schedule object are used to determine the value of the Present_Value property. This means that the Present_Value property is decoupled from the internal calculations and will not track changes to other properties when Out_Of_Service is TRUE. Other functions that depend on the state of the Present_Value, such as writing to the members of the List_Of_Object_Property_References, shall respond to changes made to that property while Out_Of_Service is TRUE, as if those changes had occurred by internal calculations.

[Renumber **18.3.2** through **18.3.11** to **18.3.3** through **18.3.12**, and insert new **18.3.2**, p.334]

18.3.2 DATATYPE_NOT_SUPPORTED - The data is of, or contains, a datatype not supported by this instance of this property.

[Change Clause 21, p.385]

```
Error     ::= SEQUENCE {
…
     error-code     ENUMERATED {
                    other                                (0),
                    authentication-failed                (1),
                    character-set-not-supported          (41),
                    configuration-in-progress            (2),
                    datatype-not-supported               (47),
                    device-busy                          (3),
                    …
                    .
               -- see optional-functionality-not-supported, (45),
               -- see invalid-configuration-data           (46),
               -- see datatype-not-supported              (47),
                    ...
          }
          -- Enumerated values 0-255 are reserved for definition by ASHRAE. Enumerated values
          -- 256-65535 may be used by others subject to the procedures and constraints described
          -- in Clause 23. The last enumeration used in this version is 46 47.
   }
```

[Change Clause 21, p. 387]

```
Date ::= [APPLICATION 10] OCTET STRING (SIZE(4)) -- see 20.2.12
--   first octet    year minus 1900       X'FF' = unspecified
--   second octet   month (1..12 14)      1 = January
--                                        odd months=13
--                                        even months=14
--                                        X'FF' = unspecified
--   third octet    day of month (1..31 32),
--                                             32=last day of month
--                                        X'FF' = unspecified
--   fourth octet   day of week  (1..7)   1 = Monday
--                                        7 = Sunday
--                                        X'FF' = unspecified
```

[Change Clause 21, pp. 399-403]

```
BACnetPropertyIdentifier ::= ENUMERATED {
     …
     resolution                 (106),
     schedule-default           (174),
     segmentation-supported     (107),
     …
     -- see profile-name        (168),
     -- see schedule-default    (174),
     …
               }
```

[Change Clause 21, p. 405]

BACnetReliability ::= ENUMERATED {
…
multi-state-fault (9),
configuration-error (10),
...
}

[Change Clause 21, p. 407]

BACnetWeekNDay ::= OCTET STRING (SIZE (3))
-- first octet month (1..~~12~~*14*) January = 1,
-- *odd months=13*
-- *even months=14*
-- X'FF' = any month
-- second octet weekOfMonth where: 1 = days numbered 1-7
-- 2 = days numbered 8-14
-- 3 = days numbered 15-21
-- 4 = days numbered 22-28
-- 5 = days numbered 29-31
-- 6 = last 7 days of this month
-- X'FF' = any week of this month
-- third octet dayOfWeek (1..7) where 1 = Monday
-- 7 = Sunday
-- X'FF' = any day of week

[Change Annex C, p. 437]

SCHEDULE ::= SEQUENCE {
…

exception-schedule	[38]	SEQUENCE OF BACnetSpecialEvent OPTIONAL, -- accessed as a BACnetARRAY
schedule-default	*[174]*	*ABSTRACT-SYNTAX.&Type, -- Any primitive datatype,*
list-of-object-property-references	[54]	SEQUENCE OF BACnetDeviceObjectPropertyReference,
priority-for-writing	[88]	Unsigned (1..16),
status-flags	*[111]*	*BACnetStatusFlags,*
reliability	*[103]*	*BACnetReliability,*
out-of-service	*[81]*	*BOOLEAN,*
profile-name	[168]	CharacterString OPTIONAL

}

[Change **Annex D.22**, p. 455]

The following is an example of a Schedule object that is used to control a classroom during the school year. In this example, a different Schedule object is assumed to be defined for the remainder of the calendar year. The reference property of this schedule is the Present_Value of a Binary Output object that controls a rooftop unit providing conditioned air to room 208.

…
Property: Weekly_Schedule = {((8:00,ACTIVE),(17:00,INACTIVE)),
((8:00,ACTIVE)),
((8:00,ACTIVE),(17:00,INACTIVE)),
((8:00,ACTIVE),(17:00,INACTIVE),(19:00,ACTIVE),(23:30,INACTIVE)),
((8:00,ACTIVE),(17:00,INACTIVE)),
((00:00,INACTIVE)),
((10:00,ACTIVE),(17:00,INACTIVE))}

Property:	Exception_Schedule =	{((23-NOV-1995),(0:00,INACTIVE),10), ((HOLIDAYS,(0:00,INACTIVE),11), ((5-MAR-1996)-(7-MAR-1996),*((0:00,INACTIVE),*(9:00,ACTIVE),(14:00,INACTIVE)),6) *((8-MAR-1996),((10:00,INACTIVE),(11:00,NULL)),7)*}
Property:	List_Of_Object_Property_References =	((Binary Output, Instance 9), Present_Value)
Property:	Priority_For_Writing =	15
Property:	*Status_Flags =*	*{FALSE, FALSE, FALSE, FALSE}*
Property:	*Reliability =*	*NO_FAULT_DETECTED*
Property:	*Schedule_Default =*	*INACTIVE*
Property:	*Out_Of_Service =*	*FALSE*

On normal Mondays, Wednesdays, and Fridays the rooftop is made ACTIVE at 8 A.M. and INACTIVE at 5 P.M. On normal Tuesdays, the rooftop is made ACTIVE at 8 A.M. and is not otherwise affected by the schedule. On normal Thursdays, the rooftop is made ACTIVE at 8 A.M., INACTIVE at 5 P.M., ACTIVE again at 7 P.M., and INACTIVE at 11:30 P.M. *On Saturdays the rooftop is INACTIVE all day. On Sundays the rooftop is made ACTIVE at 10 A.M. and INACTIVE at 5 P.M.*

On ~~Friday~~ *Thursday*, November 23, 1995, the normal ~~Friday~~ *Thursday* schedule is overridden by the first SPECIALEVENT in the Exception_Schedule list, which sets the rooftop INACTIVE for the entire day.

On Monday, February 19, 1996, the normal Monday schedule is overridden by the second SPECIALEVENT in the Exception_Schedule list, which follows the HOLIDAYS calendar described in D.8. Since February 19, 1996, Presidents' Day, is in the HOLIDAYS object's DateList, the TIMEVALUEs associated with the second SPECIALEVENT are followed, which set the rooftop INACTIVE for the entire day.

The third SPECIALEVENT overrides the normal schedule between March 5, 1996, and March 7, 1996, for a teachers' conference. On these three days, the rooftop is made ACTIVE at 9 A.M. and INACTIVE at 2 P.M.

The fourth SPECIALEVENT overrides the normal Friday schedule on March 8, 1996, to create an hour of inactivity in the middle of the day for maintenance. On this day, the rooftop is made ACTIVE at 8 A.M. (by the normal schedule), INACTIVE at 10 A.M. (by the exception schedule), ACTIVE again at 11 A.M. (by the exception relinquishing, and the normal schedule resuming) and INACTIVE at 5 P.M. (by the normal schedule).