

BSR/ASHRAE Addendum *b*
to ANSI/ASHRAE Standard 135.1-2003

Public Review Draft

ASHRAE® Standard

Proposed Addendum *b* to Standard 135.1-2003, *Method of Test for Conformance to BACnet®*

First Public Review (March 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: **tbd**.

- 135.1-2003b-1. Omit certain tests when Averaging and Command properties are fixed or not present, p. 1.
- 135.1-2003b-2. Accommodate Group objects whose members list is not changeable, p. 3.
- 135.1-2003b-3. Revise Alarm Acknowledgement tests, p. 5.
- 135.1-2003b-4. Add new Alarm Acknowledgement "offnormal" tests, p. 8.
- 135.1-2003b-5. Label conditionally-writable properties in the EPICS, p. 10.
- 135.1-2003b-6. Add new object types, p. 11.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135.1-2003 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.

135.1-2003b-1. Omit certain tests when Averaging and Command properties are fixed or not present.

Rationale

Several tests for the Averaging and Command objects make no provision for being skipped if relevant optional properties are not present or if the object has a fixed configuration.

Addendum 135.1-2003b-1

[Change 7.3.2.4.2, p. 50.]

7.3.2.4.2 Managing the Sample Window

Purpose: To verify that an Averaging object correctly tracks the average, minimum, and maximum values attained in a sample. This includes monitoring before and after the sampling window is full.

Test Concept: An Averaging object is configured to monitor a property that can be controlled manually by the testing agent or by the TD. The TD initializes the sample and then monitors the `Minimum_Value`, `Average_Value`, `Maximum_Value`, `Attempted_Samples`, and `Valid_Samples` properties after each sampling interval to verify that their values are properly tracking the monitored value. This requires the ability to manipulate the values of the monitored property value and a slow enough sampling interval to permit the analysis. This continues until after the sample window is full. *If the IUT does not support Averaging object configuration, then this test shall be omitted.*

Configuration Requirements: The IUT shall be configured with an Averaging object used to monitor a property that can be controlled by the testing agent or by the TD. The sampling interval shall be configured to allow time to change the monitored property value and to determine if each of the properties `Minimum_Value`, `Average_Value`, `Maximum_Value`, `Attempted_Samples`, and `Valid_Samples` correctly changes after each sample interval.

[Change 7.3.2.9.1, pp. 55.]

7.3.2.9.1 All Writes Successful with Post_Delay Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clause: 12.9.8.

Purpose: To verify that a Command object can successfully execute an action list that includes post delays.

Test Concept: The IUT is configured with an action list that includes manipulating a sequence of externally visible outputs with a time delay between each output. The TD triggers this action list and the tester observes the external changes. *If the IUT does not support Post Delay, then this test shall be omitted. If the IUT does not support action list configuration, then this test shall be omitted.*

Configuration Requirements: The IUT shall be configured with a Command object having an action list, X, that includes writing to a sequence of externally visible outputs. There shall be a post delay between writes to the externally visible outputs that is long enough for the tester to observe the delay.

[Change 7.3.2.9.2, p. 56.]

7.3.2.9.2 Quit on Failure Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clause: 12.9.8.

Purpose: To verify that a Command object can successfully execute `Quit_On_Failure` procedures.

Test Concept: The IUT is configured with two action lists that include a sequence of externally visible outputs with a write somewhere in the sequence that will fail. The action lists are identical except that one has `Quit_On_Failure` set to `TRUE` and the other set to `FALSE`. The TD triggers both action lists. The external outputs are observed to verify that the failure procedures are properly implemented. *If the IUT does not support action list configuration, then this test shall be omitted.*

Configuration Requirements: The IUT shall be configured with a Command object having at least two action lists, X and Y, that includes writing to a sequence of externally visible outputs. Somewhere in the sequence there shall be a write command that will fail that is followed by write commands that will succeed. Both action lists shall be identical except that list X shall have `Quit_On_Failure` set to `TRUE` and Y shall have `Quit_On_Failure` set to `FALSE`.

[Change 7.3.2.9.4, p. 57.]

7.3.9.2.4 Empty Action List Test

Dependencies: WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clause: 12.9.8.

Purpose: To verify that a Command object takes no action when `Present_Value` is written to with a non-zero value that corresponds to an empty action list.

Test Concept: The IUT is configured with at least one empty action list. The TD triggers the action list. The external outputs are observed to verify that no changes occurred. If the IUT does not support action list configuration, then this test shall be omitted.

Configuration Requirements: The IUT shall be configured with a Command object that has an Action property with at least one empty action list.

[Change 7.3.2.9.7, p. 58.]

7.3.9.2.7 Write While In_Process is TRUE Test

Dependencies: WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clauses: 12.9.8 and 12.9.9.

Purpose: To verify that an action list continues to completion if a second action list is commanded while `In_Process` is `TRUE` and that the second action list is not executed.

Test Concept: The IUT is configured with two action lists that include a sequence of externally visible outputs with post delays for each action. The TD triggers the first action list. The external outputs are observed in order to trigger the second action list during the post delay of the first list. The TD triggers the second action list. The external outputs are observed to verify that the second action list is not executed. If the IUT does not support Post Delay, then this test shall be omitted. If the IUT does not support action list configuration, then this test shall be omitted.

Configuration Requirements: The IUT shall be configured with a Command object having two distinct action lists, X and Y, that include writing to a sequence of externally visible outputs. There shall be a post delay between writes to the externally visible outputs that is long enough for the tester to observe the delay (This ensures `In_Process` remains `TRUE` long enough to command the second action list).

135.1-2003b-2. Accommodate Group objects whose members list is not changeable.

Rationale

The Group Object (functionality) test should accommodate Group objects whose List_Of_Group_Members references one or more properties that are changeable.

Addendum 135.1-2003b-2

[Change 7.3.2.13, pp. 58-59.]

7.3.2.13 Group Object Test

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clause: 12.13.

Purpose: To verify that the Present_Value of a Group properly tracks the values of the properties of the objects that make up the group.

Test Concept: The Present_Value of a Group object is read. Each of the object, property combinations that make up the membership of the Group is also read. The values are compared to verify that they match. The value of one of the Group members is changed. The Present_Value of the Group is read again to verify that it correctly tracks the change.

Configuration Requirements: The IUT shall be configured with a Group object that has at least two members. One of the group members shall be changeable by the WriteProperty service or some other mechanism provided by the vendor. The value of the properties that make up the Group shall remain static for the duration of the test except for changes made as part of the test procedure.

Test Steps:

1. TRANSMIT ReadProperty-Request,
 'Object Identifier' = (the Group object being tested),
 'Property Identifier' = List_Of_Group_Members
2. RECEIVE ReadProperty-ACK,
 'Object Identifier' = (the Group object being tested),
 'Property Identifier' = List_Of_Group_Members,
 'Property Value' = (any valid list of group members)
3. TRANSMIT ReadProperty-Request,
 'Object Identifier' = (the Group object being tested),
 'Property Identifier' = Present_Value
4. RECEIVE ReadProperty-ACK,
 'Object Identifier' = (the Group object being tested),
 'Property Identifier' = Present_Value,
 'Property Value' = (any valid set of values consistent with the properties that make up the group)
5. REPEAT X = (each object, property combination returned in the List_Of_Group_Members in step 2) DO {
 VERIFY X = (the same value that was returned for this group member in step 4)}
6. IF (a property value of a group member is changeable) THEN
 IF (the changeable group member property value is writable) THEN
 WRITE (the writable property that is a member of the group) = (a value different from its current value)
 ELSE
 MAKE (the changeable group member property value different from its current value)
7. WAIT **Internal Processing Fail Time**
8. TRANSMIT ReadProperty-Request,
 'Object Identifier' = (the Group object being tested),
 'Property Identifier' = Present_Value
9. RECEIVE ReadProperty-ACK,

- 'Object Identifier' = (the Group object being tested),
 - 'Property Identifier' = Present_Value,
 - 'Property Value' = (the same set of values received in step 4 except for the value changed in step 6)
10. REPEAT X = (each object, property combination returned in the List_Of_Group_Members in step 2) DO {
VERIFY X = (the same value that was returned for this group member in step 9)}

135.1-2003b-3. Revise Alarm Acknowledgement tests.

Rationale

Several revisions to Alarm Acknowledgement tests are made to improve the handling of different test situations and new error codes.

Addendum 135.1-2003b-3

[Change 9.1.2.3, p. 184.]

9.1.2.3 Unsuccessful Alarm Acknowledgment of Confirmed Event Notifications Because the 'Event Object Identifier' is Invalid Referenced Object Does Not Exist

Purpose: This test case verifies that an alarm remains unacknowledged if the 'Event Object Identifier' represents an object that does not exist or is not consistent with the other parameters that define the alarm being acknowledged.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and at least one other device. The TD acknowledges the alarm using an improper 'Event Object Identifier' and verifies that the acknowledgment is not accepted by the IUT and that the IUT does not notify other devices that the alarm was acknowledged. The TD then acknowledges the alarm using the proper 'Event Object Identifier' and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm was acknowledged.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send confirmed notifications. The Acked_Transitions property shall have the value B'111" indicating that all transitions have been acknowledged. The TD and at least one other BACnet device shall be recipients of the alarm notification.

Test Steps: The test steps defined in 9.1.2.1 shall be followed except that in the first AcknowledgeAlarm request the 'Time Stamp' shall have the same value as the 'Time Stamp' from the event notification, and the 'Event Object Identifier' shall have a value that is different from the 'Event Object Identifier' in the event notification *and for which no object exists in the IUT.*

Passing Result: A passing result is the same message sequence described in 9.1.2.1 except that the *Error Class in step 7 shall be OBJECT and the Error Code in step 7 shall be INCONSISTENT_PARAMETERS. UNKNOWN_OBJECT. For devices that claim a Protocol_Revision of 5 or prior, an Error Class of SERVICES with an Error Code of INCONSISTENT_PARAMETERS shall also be accepted.*

[Change 9.1.2.4, p. 184.]

9.1.2.4 Unsuccessful Alarm Acknowledgment of Confirmed Event Notifications Because the 'Event State Acknowledged' is Invalid

Purpose: This test case verifies that an alarm remains unacknowledged if the 'Event State Acknowledged' is inconsistent with the other parameters that define the alarm being acknowledged.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and at least one other device. The TD acknowledges the alarm using an invalid event state and verifies that the acknowledgment is not accepted by the IUT and that the IUT does not notify other devices that the alarm was acknowledged. The TD then acknowledges the alarm using the proper event state and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm was acknowledged.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send confirmed notifications. The Acked_Transitions property shall have the value B'111" indicating that all transitions have been acknowledged. The TD and at least one other BACnet device shall be recipients of the alarm notification.

Test Steps: The test steps defined in 9.1.2.1 shall be followed except that in the first AcknowledgeAlarm request the 'Time Stamp' shall have the same value as the 'Time Stamp' from the event notification and the 'Event State

Acknowledged' shall have an *off-normal* value *other than OFFNORMAL* and *other than the value of that is different from the 'To State' parameter* in the event notification.

Passing Result: A passing result is the same message sequence described in 9.1.2.1 except that the Error Code in step 7 shall be ~~INCONSISTENT_PARAMETERS~~. *INVALID_EVENT_STATE*. For devices that claim a *Protocol_Revision* of 5 or prior, an Error Code of *INCONSISTENT_PARAMETERS* shall also be accepted.

[Change 9.1.2.6, p. 187.]

9.1.2.6 Unsuccessful Alarm Acknowledgment of Unconfirmed Event Notifications Because the 'Event Object Identifier' is Invalid Referenced Object Does Not Exist

Purpose: This test case verifies that an alarm remains unacknowledged if the 'Event Object Identifier' represents an object that does not exist ~~or is not consistent with the other parameters that define the alarm being acknowledged~~.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and at least one other device. The TD acknowledges the alarm using an invalid event object identifier and verifies that the acknowledgment is not accepted by the IUT and that the IUT does not notify other devices that the alarm was acknowledged. The TD then acknowledges the alarm using the proper event object identifier and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm was acknowledged.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send unconfirmed notifications. The *Acked_Transitions* property shall have the value B'111" indicating that all transitions have been acknowledged. The TD and at least one other BACnet device shall be recipients of the alarm notification.

Test Steps: The test steps defined in 9.1.2.5 shall be followed except that in the first *AcknowledgeAlarm* request the 'Time Stamp' shall have the same value as the 'Time Stamp' from the event notification, and the 'Event Object Identifier' shall have a value that is different from the 'Event Object Identifier' in the event notification *and for which no object exists in the IUT*.

Passing Result: A passing result is the same message sequence described as the passing result in 9.1.2.5 except that the *Error Class* in step 7 shall be *OBJECT* and the Error Code in step 7 shall be ~~INCONSISTENT_PARAMETERS~~. *UNKNOWN_OBJECT*. For devices that claim a *Protocol_Revision* of 5 or prior, an Error Code of *SERVICES* with an Error Code of *INCONSISTENT_PARAMETERS* shall also be accepted.

[Change 9.1.2.7, p. 187.]

9.1.2.7 Unsuccessful Alarm Acknowledgment of Unconfirmed Event Notifications Because the 'Event State Acknowledged' is Invalid

Purpose: This test case verifies that an alarm remains unacknowledged if the 'Event State Acknowledged' is inconsistent with the other parameters that define the alarm being acknowledged.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and at least one other device. The TD acknowledges the alarm using an invalid 'Event State Acknowledged' and verifies that the acknowledgment is not accepted by the IUT and that the IUT does not notify other devices that the alarm was acknowledged. The TD then acknowledges the alarm using the proper 'Event State Acknowledged' and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm was acknowledged.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send unconfirmed notifications. The *Acked_Transitions* property shall have the value B'111" indicating that all transitions have been acknowledged. The TD and at least one other BACnet device shall be recipients of the alarm notification.

Test Steps: The test steps defined in 9.1.2.1 shall be followed except that in the first AcknowledgeAlarm request the 'Time Stamp' shall have the same value as the 'Time Stamp' from the event notification and the 'Event State Acknowledged' shall have an *off-normal* value *other than OFFNORMAL and other than the value of that is different from the 'To State' parameter* in the event notification.

Passing Result: A passing result is the same message sequence described as the passing result in 9.1.2.5 except that the Error Code in step 7 shall be ~~INCONSISTENT_PARAMETERS~~ *INVALID_EVENT_STATE*. *For devices that claim a Protocol_Revision of 5 or prior, an Error Code of INCONSISTENT_PARAMETERS shall also be accepted.*

135.1-2003b-4. Add new Alarm Acknowledgement "offnormal" tests.

Rationale

Addendum 135-2004d-3 clarified the acknowledgement of "offnormal" states where there are multiple such states (e.g., high-limit and low-limit). Tests are added for acknowledgements in such situations.

Addendum 135.1-2003b-4

[Add new clause 9.1.1.7 p. 181.]

9.1.1.7 Successful Alarm Acknowledgment of any "Offnormal" Transitions Using an "Offnormal" 'To State'

Purpose: To verify the successful acknowledgment of an alarm that indicated an "offnormal" 'To State' other than offnormal.

Test Concept: An "offnormal" alarm is triggered in the IUT where the "offnormal" state is represented by an event-state other than offnormal (such as high-limit or low-limit). The TD acknowledges the alarm with an 'Event State Acknowledged' of offnormal and verifies that the acknowledgment is accepted by the IUT.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and can enter "offnormal" states other than offnormal. The TD shall be a recipient of the alarm notification and the IUT shall be configured to send it unconfirmed. If the IUT cannot be configured to generate such a notification, then this test shall be skipped.

Test Steps:

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. RECEIVE UnConfirmedEventNotification-Request,
 - 'Process Identifier' = (the process identifier configured for this event),
 - 'Initiating Device Identifier' = IUT,
 - 'Event Object Identifier' = (the object detecting the alarm),
 - 'Time Stamp' = (the current time or sequence number),
 - 'Notification Class' = (the notification class configured for this event),
 - 'Priority' = (the priority configured for this event),
 - 'Event Type' = (any valid event type),
 - 'Notify Type' = (any valid notify type),
 - 'AckRequired' = TRUE,
 - 'From State' = (any valid event-state),
 - 'To State' = (any "offnormal" event state other than offnormal itself),
 - 'Event Values' = (the values appropriate to the event type)
3. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = {0,?,?}
4. TRANSMIT AcknowledgeAlarm-Request,
 - 'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
 - 'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
 - 'Event State Acknowledged' = offnormal,
 - 'Time Stamp' = (the timestamp conveyed in the notification),
 - 'Time of Acknowledgment' = (any valid timestamp)
5. RECEIVE BACnet-Simple-ACK-PDU
6. IF (Protocol_Revision is present and Protocol_Revision \geq 1) THEN
 - RECEIVE UnconfirmedEventNotification-Request,
 - 'Process Identifier' = (the process identifier configured for this event),
 - 'Initiating Device Identifier' = IUT,
 - 'Event Object Identifier' = (the object detecting the alarm),
 - 'Time Stamp' = (the current time or sequence number),

'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step 2),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (offnormal or the 'To State' from step 2)

ELSE
RECEIVE UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step 2),
'Notify Type' = ACK_NOTIFICATION,

7. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = {1,?,?}

135.1-2003*b*-5. Label conditionally-writable properties in the EPICS.

Rationale

To allow the differentiation between properties that are always writable and those that are only writable under specific conditions, a new keyword is added to the EPICS language.

Addendum 135.1-2003*b*-5

[Change 4.5.10, pp. 8-9.]

4.5.10 Test Database

...

Properties in the test database that are writable shall have a "W" following the property value, as shown in the example below:

```
{
  object-identifier: (analog-value, 6)
  object-name: "□"
  object-type: analog-value
  present-value: 23.4 W
  other properties...
}
```

Properties in the test database that are conditionally writable shall have a "C" following the property value, as shown in the example below. It is recommended that the governing mechanism be identified in a comment:

```
{
  object-identifier: (analog-input, 6)
  object-name: "□"
  object-type: analog-input
  present-value: 12.3 C -- Writable when Out_Of_Service is TRUE
  other properties...
}
```

The following sections show templates for each of the standard object types. To improve readability the carriage return/linefeed pairs are not explicitly shown in the examples.

...

135.1-2003b-6. Add new object types.

Rationale
 Several object types have been added to ANSI/ASHRAE Standard 135 since the 2001 version. This revision adds object types that appear in the 2004 version, plus those published in Addenda 135-2004*d*, *e* and *f*. This adds references to those objects only, not the tests for those objects.

Addendum 135.1-2003b-6

[Change 4.5.4, p. 6.]

4.5.4 Object Types Supported

...

The standard objects may be any of:

<i>Access Door</i>	Command	Multi-state Output
<i>Accumulator</i>	Device	Multi-state Value
Analog Input	Event Enrollment	Notification Class
Analog Output	File	Program
Analog Value	Group	<i>Pulse Converter</i>
Averaging	Life Safety Point	Schedule
Binary Input	Life Safety Zone	<i>Structured View</i>
Binary Output	<i>Load Control</i>	Trend Log
Binary Value	Loop	
Calendar	Multi-state Input	

[Add new 4.5.10.X, p.19]

4.5.10.X Access Door

- {
- object-identifier: (access-door,)
- object-name: ""
- object-type: access-door
- present-value:
- description: ""
- status-flags:
- event-state:
- reliability:
- out-of-service:
- priority-array:
- relinquish-default:
- door-status:
- lock-status:
- secured-status:
- door-members: {, ...}
- door-pulse-time:
- door-extended-pulse-time:
- door-unlock-delay-time:
- door-open-too-long-time:
- door-alarm-state:
- masked-alarm-values: {, ...}
- maintenance-required:
- time-delay:
- notification-class:
- alarm-values: {, ...}
- fault-values: {, ...}
- }

```
event-enable: {□,□,□}  
acked-transitions: {□,□,□}  
notify-type: □  
event-time-stamps: {□,□,□}  
profile-name: "□"  
}
```

[Add new **4.5.10.X**, p.19]

4.5.10.X Accumulator

```
{  
  object-identifier: (accumulator, □)  
  object-name: "□"  
  object-type: accumulator  
  present-value: □  
  description: "□"  
  device-type: "□"  
  status-flags: □  
  event-state: □  
  reliability: □  
  out-of-service: □  
  scale: □  
  units: □  
  prescale: □  
  max-pres-value: □  
  value-change-time: □  
  value-before-change: □  
  value-set: □  
  logging-record: □  
  logging-object: □  
  pulse-rate: □  
  high-limit: □  
  low-limit: □  
  limit-monitoring-interval: □  
  notification-class: □  
  time-delay: □  
  limit-enable: □  
  event-enable: {□,□,□}  
  acked-transitions: {□,□,□}  
  notify-type: □  
  event-time-stamps: {□,□,□}  
  profile-name: "□"  
}
```

[Add new **4.5.10.X**, p. 19]

4.5.10.X Load Control

```
{  
  object-identifier: (load-control, □)  
  object-name: "□"  
  object-type: load-control  
  description: "□"  
  present-value: □  
  state-description: "□"  
  status-flags: □  
  event-state: □  
}
```

```
reliability: □  
requested-shed-level: □  
start-time: □  
shed-duration: □  
duty-window: □  
enabled: □  
full-duty-baseline: □  
expected-shed-level: □  
actual-shed-level: □  
shed-levels: {□, □...}  
shed-level-descriptions: {"□", "□"...}  
notification-class: □  
time-delay: □  
event-enable: {□,□,□}  
acked-transitions: {□,□,□}  
notify-type: □  
event-time-stamps: {□,□,□}  
profile-name: "□"  
}
```

[Add new **4.5.10.X**, p.19]

4.5.10.X Pulse Converter

```
{  
  object-identifier: (pulse-converter, □)  
  object-name: "□"  
  object-type: pulse-converter  
  description: "□"  
  present-value: □  
  input-reference: □  
  status-flags: □  
  event-state: □  
  reliability: □  
  out-of-service: □  
  units: □  
  scale-factor: □  
  adjust-value: □  
  count: □  
  update-time: □  
  count-change-time: □  
  count-before-change: □  
  cov-increment: □  
  cov-period: □  
  notification-class: □  
  time-delay: □  
  high-limit: □  
  low-limit: □  
  deadband: □  
  limit-enable: □  
  event-enable: {□,□,□}  
  acked-transitions: {□,□,□}  
  notify-type: □  
  event-time-stamps: {□,□,□}  
  profile-name: "□"  
}
```

[Add new **4.5.10.X**, p.19]

4.5.10.X Structured View

```
{  
  object-identifier: (structured-view, □)  
  object-name: "□"  
  object-type: structured-view  
  description: "□"  
  node-type: □  
  node-subtype: "□"  
  subordinate-list: {□, □...}  
  subordinate-annotations: {"□", "□"...}  
  profile-name: "□"  
}
```