A Joint Standard of AASHTO, ITE, and NEMA

NTCIP 1201:2005 **v02.32**

National Transportation Communications for ITS Protocol

Global Object (GO) Definitions – version 02

October 2005

A major revision of NTCIP 1201 v01.10,
which included NTCIP 1201 Amendment 1 v07

This Adobe® PDF copy of an NTCIP standard is available at no-cost for a limited time through support from the U.S. DOT / Federal Highway Administration, whose assistance is gratefully acknowledged.

Published by

American Association of State Highway and Transportation Officials (AASHTO)
444 North Capitol Street, N.W., Suite 249
Washington, D.C. 20001

Institute of Transportation Engineers (ITE)
1099 14th Street, N.W., Suite 300 West
Washington, D.C. 20005-3438

National Electrical Manufacturers Association (NEMA)
1300 North 17th Street, Suite 1752
Rosslyn, Virginia 22209-3806

NOTICES

Copyright Notice

PDF File License Agreement

Data Dictionary and MIB Distribution Permission

These materials are delivered "AS IS" without any warranties as to their use or performance.

AASHTO / ITE / NEMA and their suppliers do not warrant the performance or results you may obtain by using these materials. AASHTO / ITE / NEMA and their suppliers make no warranties, express or implied, as to noninfringement of third party rights, merchantability, or fitness for any particular purpose. In no event will AASHTO / ITE / NEMA or their suppliers be liable to you or any third party for any claim or for any consequential, incidental or special damages, including any lost profits or lost savings, arising from your reproduction or use of these materials, even if an AASHTO / ITE / NEMA representative has been advised of the possibility of such damages.

Some states or jurisdictions do not allow the exclusion or limitation of incidental, consequential or special damages, or the exclusion of implied warranties, so the above limitations may not apply to you.

Use of these materials does not constitute an endorsement or affiliation by or between AASHTO, ITE, or NEMA and you, your company, or your products and services.

If you are unwilling to accept the foregoing restrictions, you should immediately return these materials.

PRL and RTM Distribution Permission

A PICS is a Profile Requirements List which is completed to indicate the features that are supported in an implementation. Visit www.ntcip.org for information on electronic copies of the MIBs, PRLs, and RTMs.

TRF Distribution Permission [when standard contains a TRF]

To the extent that these materials are distributed by AASHTO / ITE / NEMA in the form of a Testing Requirements Form ("TRF"), AASHTO / ITE / NEMA extend the following permission:

(i) you may make or distribute unlimited electronic or hard copies, including derivative works of the TRF, provided that each copy you make or distribute contains the citation "Based on NTCIP 0000 [insert the document number] TRF. Used by permission. Original text © AASHTO / ITE / NEMA.";
(ii) you may not modify the logical flow of any test procedure, without clearly noting and marking any such modification; and
(iii) if the TRF excerpt is made from an unapproved draft, add to the citation "TRF excerpted from a draft document containing preliminary information that is subject to change."

Content and Liability Disclaimer

The information in this publication was considered technically sound by the consensus of persons engaged in the development and approval of the document at the time it was developed. Consensus does not necessarily mean that there is unanimous agreement among every person participating in the development of this document.

AASHTO, ITE, and NEMA standards and guideline publications, of which the document contained herein is one, are developed through a voluntary consensus standards development process. This process brings together volunteers or seeks out the views of persons who have an interest in the topic covered by this publication. While AASHTO, ITE, and NEMA administer the process and establish rules to promote fairness in the development of consensus, they do not write the document and they do not independently test, evaluate, or verify the accuracy or completeness of any information or the soundness of any judgments contained in their standards and guideline publications.

AASHTO, ITE, and NEMA disclaim liability for any personal injury, property, or other damages of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, application, or reliance on this document. AASHTO, ITE, and NEMA disclaim and make no guaranty or warranty, express or implied, as to the accuracy or completeness of any information published herein, and disclaims and makes no warranty that the information in this document will fulfill any of your particular purposes or needs. AASHTO, ITE, and NEMA do not undertake to guarantee the performance of any individual manufacturer or seller's products or services by virtue of this standard or guide.

In publishing and making this document available, AASHTO, ITE, and NEMA are not undertaking to render professional or other services for or on behalf of any person or entity, nor are AASHTO, ITE, and NEMA undertaking to perform any duty owed by any person or entity to someone else. Anyone using this document should rely on his or her own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstances. Information and other standards on the topic covered by this publication may be available from other sources, which the user may wish to consult for additional views or information not covered by this publication.

AASHTO, ITE, and NEMA have no power, nor do they undertake to police or enforce compliance with the contents of this document. AASHTO, ITE, and NEMA do not certify, test, or inspect products, designs, or installations for safety or health purposes. Any certification or other statement of compliance with any health or safety–related information in this document shall not be attributable to AASHTO, ITE, or NEMA and is solely the responsibility of the certifier or maker of the statement.

Trademark Notice

NTCIP is a trademark of AASHTO / ITE / NEMA. All other marks mentioned in this document are the trademarks of their respective owners.

< This page is intentionally left blank. >

ACKNOWLEDGEMENTS

This publication was prepared by the NTCIP Global Object Working Group, which is a subdivision of the Joint Committee on the NTCIP. The Joint Committee is organized under a Memorandum of Understanding among the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE), and the National Electrical Manufacturers Association (NEMA). The Joint Committee on the NTCIP consists of six representatives from each of the standards organizations, and provides guidance for NTCIP development.

At the time that this document was prepared, the following individuals were members of the NTCIP Global Object Working Group:

- Bob De Roche
- Curtis Herrick
- Paul Hing
- Bud Kent
- Larry Lindley
- Joerg “Nu” Rosenbohm
- Ken Vaughn (Chair)

Other individuals providing input to the document, include:

- Moon Chueng
- Blake Christie
- Mike Forbis
- Craig Gardner
- Robert Martindale
- Gary Meredith
- Jeris White

In addition to the many volunteer efforts, recognition is also given to those organizations who supported the efforts of the working groups by providing comments and funding for the standard, including:

- U.S. Department of Transportation, Federal Highway Administration
- Econolite Control Products, Inc.
- G. C. Herrick & Associates, Inc.
- Iteris, Inc.
- Ministry of Ontario
- Mitretek
- PB Farradyne, a Division of PBQD, Inc.
- Peek Transportation Systems
- Robert De Roche Consulting, Inc.
- Siemens ITS
- Trevilon
- Washington State Department of Transportation

This publication and version is dedicated to the memory of Curtis Herrick.

FOREWORD

This document uses only metric units.

The purpose of this publication is to identify and define the common object definitions that may be supported by devices that are NTCIP-compliant.

This document is an NTCIP Device Data Dictionary standard. Device Data Dictionary standards provide definitions of data elements for use within NTCIP systems. A Joint NTCIP data dictionary standards publication is equivalent to these document types at the standards organizations:

AASHTO – Standard Specification
ITE – Software Standard
NEMA – Standard

For more information about NTCIP standards, visit the NTCIP Web site at http://www.ntcip.org.

User Comment Instructions

The term “User Comment” includes any type of written inquiry, comment, question, or proposed revision, from an individual person or organization, about any part of this standard publication’s content. A “Request for Interpretation” of this standard publication is also classified as a User Comment. User Comments are solicited at any time. In preparation of this NTCIP standards publication, input of users and other interested parties was sought and evaluated.

All User Comments will be referred to the committee responsible for developing and/or maintaining this standards publication. The committee chairperson, or their designee, may contact the submitter for clarification of the User Comment. When the committee chairperson or designee reports the committee’s consensus opinion related to the User Comment, that opinion will be forwarded to the submitter. The committee chairperson may report that action on the User Comment may be deferred to a future committee meeting and/or a future revision of the standards publication. Previous User Comments and their disposition may be available for reference and information at www.ntcip.org.

A User Comment should be submitted to this address:

NTCIP Coordinator
National Electrical Manufacturers Association
1300 North 17th Street, Suite 1752
Rosslyn, Virginia 22209-3801
fax: (703) 841-3331
e-mail: ntcip@nema.org

A User Comment should be submitted in the following form:

Standard Publication number and version:
Page:
Paragraph or Clause:
Comment:

Please include your name, organization, and address in your correspondence.

Approvals

This document was separately balloted and approved by AASHTO, ITE, and NEMA after recommendation by the Joint Committee on the NTCIP. Each organization has approved this standard as the following standard type, as of the date:

AASHTO – Standard Specification; October 2004
ITE – Software Standard; March 2005
NEMA – Standard; November 2004

History

From 1996 to 1999, this document was referenced as NEMA TS 3.4. However, to provide an organized numbering scheme for the NTCIP documents, this document is now referenced as NTCIP 1201. The technical specifications of NTCIP 1201 are identical to the former reference, except as noted in the development history below:

NEMA TS 3.4-1996 v96.01.7, April 7, 1997. October 1996 – Version 1.5 approved by NEMA. April 1997 – Version 1.7 published by NEMA with editorial corrections. October 1996 – Accepted as a Recommended Standard by the Joint Committee on the NTCIP. Approved by AASHTO in 1997 and approved by ITE in December 1997.

NEMA TS 3.4 Amendment 1 v98.01.07. October 1998 – Version 98.01.05 accepted as a Recommended Amendment by the Joint Committee on the NTCIP, and edited v01.07 referred for balloting and approval by NTCIP Standards Bulletin B0032 in May 1999. Approved by AASHTO in October 1999, approved by ITE in January 2001, and approved by NEMA in December 1999. Amendment 1 clarified ambiguities discovered during real-world implementations of this standard.

NTCIP 1201:1996 [assigned version 01.08]. August 1999 – Assigned NTCIP 1201 document number in NTCIP Standards Bulletin B0038. August 2000 – Joint NTCIP Standards Publication cover used over TS 3.4 contents.

NTCIP 1201:1996 v01.10, December 2001. January 2002 – Formatted for printing: incorporated Amendment 1 v07 into text; updated title page date and version number; modified and reorganized front matter to conform to NTCIP 8002. Most references to TS 3 standard designations were changed to equivalent NTCIP standard numbers.

NTCIP 1201 v02. Developed to reflect additional lessons learned, to incorporate better documentation (in the Annex) of some of the logic required to implement the standards, and to add new features requested by the ITS community.

NTCIP 1201 v02.14. September 2001 – Accepted by the NTCIP Joint Committee as a User Comment Draft. February 2002 – NTCIP Standards Bulletin B0071 distributed NTCIP 1201 v02.16 for review and comment.

NTCIP 1201 v02.24. October 2002 – Accepted by the NTCIP Joint Committee as a Recommended Standard. April 2004 – NTCIP Standards Bulletin B0092 referred NTCIP 1201 v02.26 for balloting. Approved by AASHTO in October 2004, approved by ITE in March 2005, and approved by NEMA in November 2004.

NTCIP 1201 v02.31. February 2005 – Disposed of ballot period comments on backward compatibility, object deprecation, and others. In clause 1.3 Terms, added Deprecated and Obsolete definitions.

Specific changes made from NTCIP 1201 v01.10 (1996) to v02.31 include:

1. Added Unified Modeling Language (UML) diagrams to explain interrelationships between objects.
2. Adjusted the ISO Tree Structure diagram to conform to this updated document.
3. Modified the DESCRIPTION field of the object definitions to conform with ISO 14817.
4. Changed MIB name to reflect new version.
5. Added Default Value statements to several Configuration and Control object definitions. This was not necessary for Status-type object definitions.
6. Added wording to the moduleVersion object definition to clarify the value content of this object.
7. Added an object definition that allows to identify on which Information Profile MIB an implementation is based.
8. Added additional clarifications to the DESCRIPTION field of the dbCreateTransation object definition.
9. Deprecated the dbErrorType, the dbErrorID, the dbTransactionID, and the dbMakeID object definitions.
10. Expanded on the possible parameter values for the globalDaylightSavings object definition.
11. Added a timeBaseScheduleTable-status object definition to identify which schedule is currently selected by the schedule.
12. Added additional clarifying wording to the DESCRIPTION field of several objects within the dayPlanTable, especially the dayPlanActionNumberID object definition.
13. Deprecated the globalLocalTimeDifferential object definition.
14. Added the standardTimeZone and localTime object definitions.
15. Added clarifying wording to several object definitions within the Report node tables.
16. Changed the sequence of the three tables within the Report node to show the logical sequence of populating and querying those tables.
17. Added another value (andedWithValue) to the eventConfigMode object definition and clarified the wording explaining the various possible values.
18. Added another value (error) to the eventConfigStatus object definition.
19. Modified the SYNTAX of the maxEventLogSize object.
20. Deprecated the hdlcGroupAddress object definition.
21. Moved the object definitions previously defined under the Security node to NTCIP 1103.
22. Moved the object definitions for the Auxiliary I/O objects previously defined in NTCIP 1203 to a node under the Global object definitions. Some clarifications were added to those object definitions. Additionally, an object definition (auxIO-lastCommandedState) was added.
23. Deleted Section 3 (Group Definitions) and Section 4 (Conformance Statements) because conformance to definitions within this document should be made within the referencing Information Profiles such as NTCIP 1202 (ASC) or NTCIP 1203 (DMS).
24. Added Annex A to provide a Concept of Operations explaining the interrelationships between objects to achieve a particular function.
25. Added Annex B to show all deprecated object definitions.
26. Added Annex C to show UML class diagrams.

27. NOTE on dbVerifyStatus. To align NTCIP 1201 v02 with other NTCIP and several Internet standards, the object definitions in NTCIP 1201 v01.10 that had enumerated values starting with a value of (0) have been changed in NTCIP 1201 v02 to start with a value of (1). This change affected clause 2.3.6, dbVerifyStatus. The changed definition in NTCIP 1201 v02 makes this object incompatible with NTCIP 1201:1996 v01.10.

NTCIP 1201:2005 v02.32. October 2005 – Edited document for publication with modified and reorganized front matter.

Compatibility of Versions

All NTCIP Standards Publications have a major and minor version number for configuration management. The version number syntax is "v00.00a," with the major version number before the period, and the minor version number and edition letter (if any) after the period.

Anyone using this document should seek information about the version number that is of interest to them in any given circumstance. The MIB, the PRL, and the PICS should all reference the version number of the standards publication that was the source of the excerpted material.

Compliant systems based on later, or higher, version numbers MAY NOT be compatible with compliant systems based on earlier, or lower, version numbers. Anyone using this document should also consult NTCIP 8004 for specific guidelines on compatibility.

INTRODUCTION

This publication defines data elements for use with various transportation devices. The data is defined using the Simple Network Management Protocol (SNMP) object-type format as defined in RFC 1212 and the NTCIP format defined in NTCIP 8004. This data would typically be exchanged using one of the NTCIP 1103 recognized Application Layers (e.g., SNMP).

This standard defines requirements that are applicable to all NTCIP environments and it also contains optional and conditional clauses that are applicable to specific environments for which they are intended.

The following keywords apply to this document: AASHTO, ITE, NEMA, NTCIP, global, data, data dictionary, object.

In 1992, the NEMA 3-TS Transportation Management Systems and Associated Control Devices Section began the effort to develop the NTCIP. Under the guidance of the Federal Highway Administration's NTCIP Steering Group, the NEMA effort was expanded to include the development of communications standards for all transportation field devices that could be used in an ITS network.

In September 1996, an agreement was executed among AASHTO, ITE, and NEMA to jointly develop, approve, and maintain the NTCIP standards. In late 1998, the Global Object Working Group was tasked with updating the Global Object Definitions standard. The first meeting of the GO WG was held in January 1999.

CONTENTS

Page

Section 1
GENERAL

1.1 SCOPE

The messaging between Transportation Management Center and field devices is accomplished by using the NTCIP Application Layer services to convey requests to access or modify values stored in a given device; these values are referred to as objects. The purpose of this publication is to identify and define these objects definitions that may be supported by multiple device types (e.g. actuated signal controllers and variable message signs). The grouping of objects for a given device type is performed in the device-type-specific object definition standard.

1.2 REFERENCES

For approved revisions, contact:

NTCIP Coordinator
National Electrical Manufacturers Association
1300 North 17th Street, Suite 1752
Rosslyn, VA 22209-3806
fax: (703) 841-3331
e-mail: ntcip@nema.org

For draft revisions, which are under discussion by the relevant NTCIP Working Group, and recommended revisions of the NTCIP Joint Committee, visit http://www.ntcip.org.

The following standards (normative references) contain provisions which, through reference in this text, constitute provisions of this Standard. Other documents and standards (other references) are referenced in these documents, which might provide a complete understanding of the entire protocol and the relations between all parts of the protocol. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standard listed below.

1.2.1 Normative References

National Electrical Manufacturers Association
1300 North 17th Street, Suite 1752
Rosslyn, VA 22209

NTCIP 1103 *Transportation Management Protocols*

NTCIP 8004 *Structure and Identification of Management Information*

ANSI
11 West 42nd Street, 13th Floor
New York, NY 10036

ISO/IEC 8824-1:1998 *Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation*

Obtain Request for Comment (RFC) electronic documents from several repositories on the World Wide Web, or by “anonymous” File Transfer Protocol (FTP) with several hosts. Browse or FTP to:

http://www.rfc-editor.org/
http://www.rfc-editor.org/repositories.html
for FTP sites, read ftp://ftp.isi.edu/in-notes/rfc-retrieval.txt

1.2.2 Other References

National Electrical Manufacturers Association
1300 North 17th Street, Suite 1752
Rosslyn, VA 22209

TS 2-2003 *Traffic Controller Assemblies with NTCIP Requirements*

NTCIP 1102 *Octet Encoding Rules*

NTCIP 1104 *Naming Conventions*

NTCIP 9001 *NTCIP Guide*

ANSI
11 West 42nd Street, 13th Floor
New York, NY 10036
(212) 642-4900

ISO/IEC 8825-1:1998 *Information Technology—ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

Obtain Request for Comment (RFC) electronic documents from several repositories on the World Wide Web, or by “anonymous” File Transfer Protocol (FTP) with several hosts. Browse or FTP to:

http://www.rfc-editor.org/
http://www.rfc-editor.org/repositories.html
for FTP sites, read ftp://ftp.isi.edu/in-notes/rfc-retrieval.txt

IAB STD 15 RFC 1157 *A Simple Network Management Protocol (SNMP)*. M. Schoffstall; M. Feder; J. Davin; J. Case; 05/10/1990

IAB STD 16 RFC 1155 *Structure and Identification of Management Information for TCP/IP-based Internets*. K. McCloghrie; M. Rose; 05/10/1990

RFC 1212 *Concise MIB Definitions*. K. McCloghrie; M. Rose; 03/26/1991

IAB STD 17 RFC 1213 *Management Information Base for Network Management of TCP/IP-based Internets:* MIB-II. K. McCloghrie; M. Rose; CP/IP-base

1.3 TERMS

For the purposes of this standard, the following terms, definitions, acronyms, and abbreviations apply. Terms not defined in this clause are in accordance with their definitions in the NTCIP 8004. Electrical and electronic terms not defined in this clause are used in accordance with their definitions in IEEE Std 100-2000. English words not defined in this clause or in IEEE Std 100-2000 are used in accordance with their definitions in *Webster’s New Collegiate Dictionary*.

class An abstraction of any kind of object that may be described; equivalent to a ISO 14817 Object Class.

component A central system, field device, etc. that supports NTCIP.

conformance level Each of the defined Profiles have one or more layers specifying the protocols that must be implemented in a device to correspond to a particular level of NTCIP support.

data value The value of a data element.

database object Any object identified as a ‘database object’ by the relevant device-specific standard. For example, in NTCIP 1202 (version 2), objects that are identified as ‘P’ or ‘P2’ are database objects.

deprecated The 'deprecated' value in the STATUS field of an 'OBJECT-TYPE' macro indicates that the subject object was included in a previous version of the standard but no longer represents the preferred design. An implementer implementing this version of the standard is not required to implement a deprecated object, but may wish to support it to foster interoperability with older implementations. The STATUS of a deprecated object will likely change to obsolete in some future version of the standard.

Feature A capability of an component

obsolete The 'obsolete' value in the STATUS field of an 'OBJECT-TYPE' macro indicates that the subject object was included in a previous version of the standard but no longer in significant use within the industry. While an implementer is allowed to implement an obsolete object, the benefits of doing so may be minimal due to the limited use of the object.

Profile Refers to a set of protocols, each of which operates independently on one of the seven (7) OSI Layers, if this layer is utilized. Different protocols are utilized at the same layer within different profiles.

Static Database Object A parameter that does not change other than by a user command. For example, the controllerTimeZone object is a static database object since it only changes value through some sort of user command, however, globalTime is not a static database object since it is constantly incrementing.

1.4 ABBREVIATIONS AND ACRONYMS

The abbreviations and acronyms used in this Standard Publication are defined as follows:

ASC Actuated Signal Controller

CRC Cyclic Redundancy Check, polynominal algorithm performed on a specified range of data resulting in a 16 or 32 bit value.

MIB Management Information Base, a collection of objects defined using Abstract Syntax Notation One (ASN.1) that can be accessed via a network management protocol.

NVT-ASCII Network Virtual Terminal, American Standard Code for Information Interchange as defined in RFC 854

PMPP Point-to-MultiPoint Protocol, a transporation specific subnetwork layer protocol that enables communication between multiple devices on the same communications line/channel.

STMP Simple Transportation Management Protocol, part of the Transportation Management Protocols of the NTCIP effort (see NTCIP 1103). STMP provides a simple and bandwidth efficient mechanism to communicate with field devices.

1.5 OBJECT TREE

The following figure provides an overview of the organization of the data defined by this document.

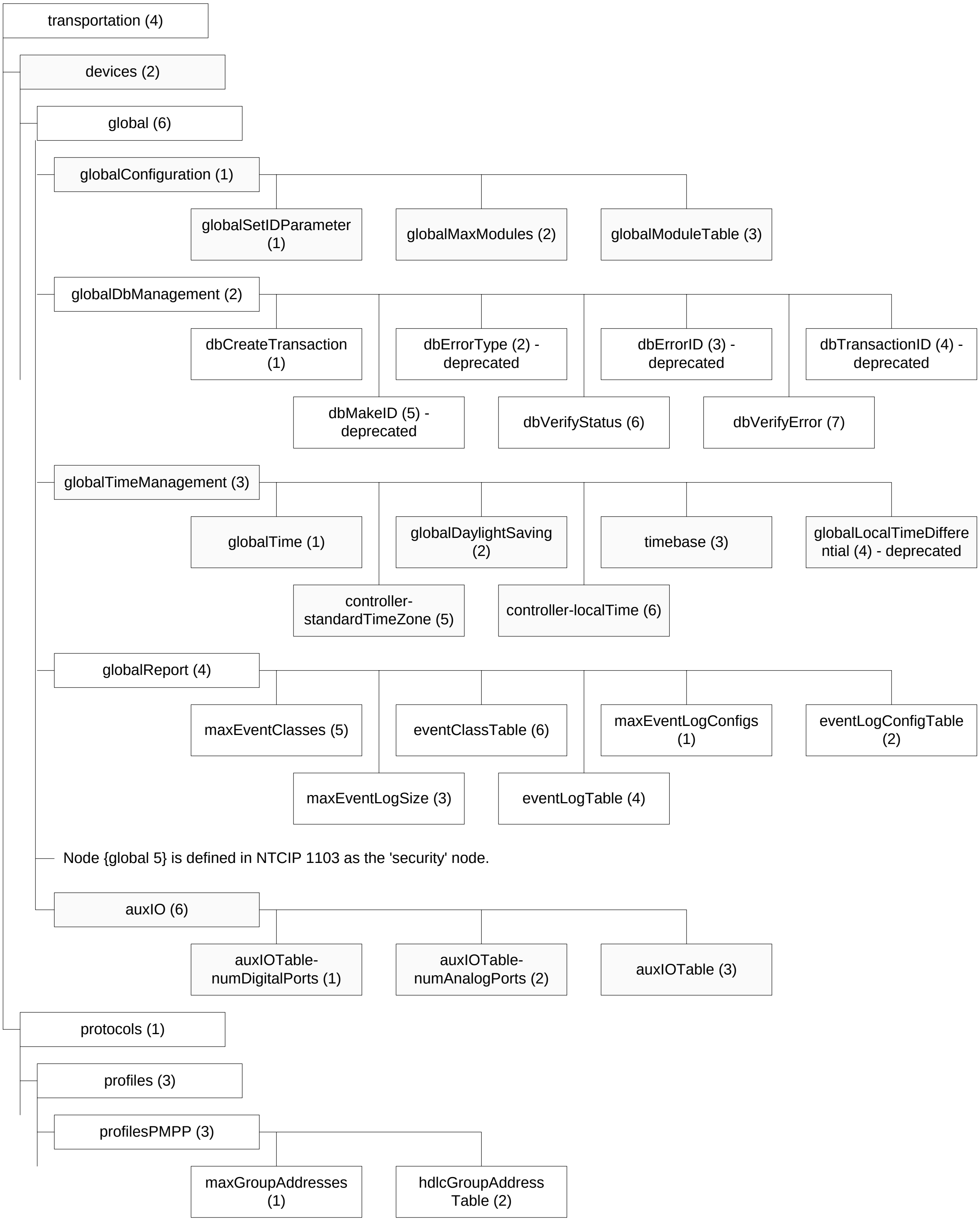

Figure 1-1
ISO Tree Structure

1.6 CONTROLLER CLASS DIAGRAM

The following figure depicts the components of data stored within a controller in Unified Modeling Language (UML) notation.

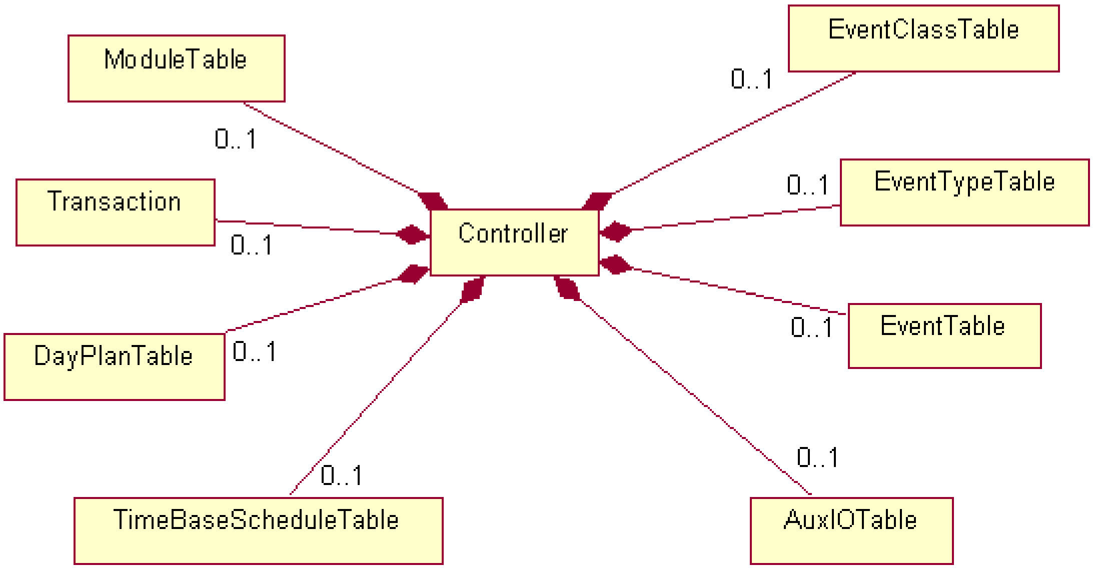

Figure 1-2
Controller Class Diagram

The diagram indicates that a Controller may or may not contain the following major components:

- A module table
- A transaction service
- A day plan table
- A timebase schedule table
- An event class table
- An event type table
- An event table
- An auxiliary input/output table

The details of each class are defined through the Management Information Base provided in Section 2. More detailed class diagrams for each feature are provided in Annex C.

Section 2
OBJECT DEFINITIONS

This section defines those objects which are expected to be used by different device types such as actuated signal controllers, variable message signs, ramp meter controllers. The objects are defined in OBJECT-TYPE macro defined in RFC 1212 per the rules defined in NTCIP 8004. The text provided from Clause 2.1 through the end of the section (except the clause headings) constitutes the standard NTCIP1201-2004 MIB.

In order to convert these object definitions into data concepts, e.g. for the exchange in center-to-center communications, the rules defined in NTCIP 8005 shall apply.

All of the objects defined in this document reside under the "global" node of the global naming tree. To aid in object management, the "global" node has been subdivided into logical categories, each defined by a node under the "global" node. The individual objects are then located under the appropriate node.

Nodes should not be confused with conformance requirements, which are defined in device specific object standards. Conformance requirements are based on logical groupings of objects that provide specific features that may be desired in a device. While the conformance requirements will frequently correspond to the nodal structure, a conformance group may contain objects that are not lexigraphically ordered. For example, a schedule conformance group may contain both "global" and "asc" specific objects.

NOTE—This version of the standard uses the NTCIP 8004 version v01.36+ conventions. It specifies all (non-deprecated/non-obsolete) objects to have a STATUS of "mandatory" according to the conventions stated in NTCIP 8004; it is the responsibility of any document referring to this standard to specify exactly which objects should be supported under what conditions through a Protocol Requirements List (PRL). Documents referring to the 1201:1996 v01.10 version of this standard shall use the STATUS settings as defined in the published (and amended) 1201:1996 v01.10 version.

Text preceded by a double hyphen in the MIB definitions represent normative text for this standard. The class diagrams contained within this standard are supplemental normative requirements to the text.

2.1 NTCIP OBJECTS

```
--NTCIP OBJECTS
NTCIP1201-2004 DEFINITIONS ::= BEGIN

-- NTCIP 8004 Header
-- <DataConceptType>Entity Type
-- <DescriptiveName>Controller
-- <DescriptiveNameContext>ITS
-- <Definition>A microprocessor, typically located in the field, that controls
and/or monitors a wayside
-- device of interest to an ITS management system.


--For the purpose of this section, the following OBJECT IDENTIFIERS are used:
IMPORTS
   OBJECT-TYPE
   FROM RFC-1212
   DisplayString
   FROM RFC1213-MIB
   devices, protocols, profiles, global
   FROM NTCIP8004-A-2004
   Opaque, Counter, Gauge, null
   FROM RFC1155-SMI;

-- global OBJECT IDENTIFIER ::= { devices 6 }
```

2.2 GLOBAL CONFIGURATION NODE

```
globalConfiguration OBJECT IDENTIFIER
::= { global 1 }
--This node is an identifier used to group all objects for support of
configuration functions
-- that are common to most device types.
```

2.2.1 Global Set ID Parameter

```
globalSetIDParameter OBJECT-TYPE
   SYNTAX   INTEGER (0..65535)
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>Specifies a relatively unique ID (e.g., this could be
      a counter, a check-sum, etc.) for current values stored in all
      user-changeable static database objects of the particular device-
      type currently implemented in the device.  This value shall be
      calculated on the change of any static database object.  The value
      reported by this object shall not change unless there has been a
      change in the static data since the last request; however a genErr
      shall be returned if the unique ID value has not yet been updated.
      A management station will be able to detect any change in the
      static database objects by monitoring this value after it has
      established a baseline. Often this ID is calculated using a CRC
      algorithm.

      <DescriptiveName>Controller.databaseID:number

      <DataConceptType>Data Element"
   ::= { globalConfiguration 1}
```

2.2.2 Maximum Modules Parameter

```
globalMaxModules   OBJECT-TYPE
   SYNTAX   INTEGER (1..255)
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>The number of rows that are listed in the
      globalModuleTable.

      <DescriptiveName>ModuleTable.maxModules:quantity

      <DataConceptType>Data Element

      <Unit>module"
   ::= { globalConfiguration 2}
```

2.2.3 Module Table

```
globalModuleTable   OBJECT-TYPE
   SYNTAX   SEQUENCE OF ModuleTableEntry
   ACCESS   not-accessible
   STATUS   mandatory
   DESCRIPTION
      "<Definition>A table containing information regarding manufacturer
      of software and hardware and the associated module models and
      version numbers as well as an indicator if the module is hardware
      or software related. The number of rows in this table shall equal
      the value of the globalMaxModules object.

      <DescriptiveName>ModuleTable

      <DataConceptType>Entity Type

      <TableType> static"
   ::= { globalConfiguration 3 }

moduleTableEntry OBJECT-TYPE
   SYNTAX   ModuleTableEntry
   ACCESS   not-accessible
   STATUS   mandatory
   DESCRIPTION
      "<Definition>This object defines an entry in the module table.

      <DescriptiveName>Module

      <DataConceptType>Entity Type"
   INDEX { moduleNumber }
   ::= { globalModuleTable 1 }

ModuleTableEntry ::= SEQUENCE {
   moduleNumber          INTEGER,
   moduleDeviceNode      OBJECT IDENTIFIER,
   moduleMake            OCTET STRING,
   moduleModel           OCTET STRING,
   moduleVersion         OCTET STRING,
   moduleType            INTEGER }
```

2.2.3.1 Module Number Parameter

```
moduleNumber OBJECT-TYPE
   SYNTAX   INTEGER (1..255)
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>This object contains the row number (1..255) within
      this table for the associated module.

      <DescriptiveName>Module.number:identifier

      <DataConceptType>Data Element"
   ::= { moduleTableEntry 1 }
```

2.2.3.2 Module Device Node Parameter

```
moduleDeviceNode   OBJECT-TYPE
   SYNTAX   OBJECT IDENTIFIER
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>This object contains the device node number of the
      device-type, e.g., an ASC signal controller would have an OID of
      1.3.6.1.4.1.1206.4.2.1.
      <DescriptiveName>Module.deviceNode:identifier
      <DataConceptType>Data Element"
   ::= { moduleTableEntry 2 }
```

2.2.3.3 Module Make Parameter

```
moduleMake   OBJECT-TYPE
   SYNTAX   OCTET STRING
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>This object specifies the manufacturer of the
      associated module.  A null-string shall be transmitted if this
      object has no entry.
      <DescriptiveName>Module.make:text
      <DataConceptType>Data Element"
   ::= { moduleTableEntry 3 }
```

2.2.3.4 Module Model Parameter

```
moduleModel  OBJECT-TYPE
   SYNTAX   OCTET STRING
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>This object specifies the model number (hardware) or
      firmware reference (software) of the associated module.  A null-
      string shall be transmitted if this object has no entry.
      <DescriptiveName>Module.model:text
      <DataConceptType>Data Element"
   ::= { moduleTableEntry 4 }
```

2.2.3.5 Module Version Parameter

```
moduleVersion OBJECT-TYPE
   SYNTAX   OCTET STRING
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>This object specifies the version of the associated
      module.  If the moduleType has a value of software, the value of
      this object shall include the date on which the software was
      released as a string in the form of YYYYMMDD, it shall be followed
      by a space, a hyphen, another space, the lower-case letter ‘v’,
      followed by a version or configuration number.  Preceding zeros
      shall be required for the date.  For example, version 7.03.02 of
      the software released on July 5, 2002 would be presented as
      20020705 – v7.03.02
      A null-string shall be transmitted if this object has no entry.
      <DescriptiveName>Module.version:text

      <DataConceptType>Data Element"
   ::= { moduleTableEntry 5 }
```

2.2.3.6 Module Type Parameter

```
moduleType  OBJECT-TYPE
   SYNTAX   INTEGER {
                  other (1),
                  hardware (2),
                  software (3) }
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>This object specifies if the associated module is a
      hardware or software module.

      <DescriptiveName>Module.type:code

      <DataConceptType>Data Element"
   ::= { moduleTableEntry 6 }
```

2.2.4 Base Standards Parameter

```
controllerBaseStandards   OBJECT-TYPE
   SYNTAX   OCTET STRING (SIZE (0..256))
   ACCESS   read-only
   STATUS   mandatory
   DESCRIPTION
      "<Definition>An ASCII string that shall identify all of the
      standard document numbers that define or reference MIBs upon which
      the device is based.  Where applicable, profiles shall be
      referenced rather than the base standards.  The string shall be
      constructed as follows: The acronym of the standards development
      organization (or other body) that developed and approved the
      standard; a space; the standards document number; a colon; and the
      documents version number as designated by the standards
      development organization (or other body).  Separate entries in the
      list of standards shall be separated by a carriage return (0x0d)
      and line feed (0x0a).

      In the case of NTCIP documents prior to formal approval, the
      version number shall be the version number in the form of lower
      case ‘v’ followed by the major version followed by a period
      followed by the minor revision.  In the case of approved NTCIP
      standards, the version number shall be the four digit year of
      publication followed by a space and the version string indicated
      above. In the case of amended NTCIP standards, it shall consist of
      the four digit year of publication of the published standard
      followed by the upper case letter ‘A’, followed by the amendment
      number.

      For example, a message sign may have the following value for this
      object:

            NTCIP 1201:v02.19
            NTCIP 1203:1997A1
            NTCIP 2101:2001 v01.19
            NTCIP 2103:v01.13
            NTCIP 2201:v01.14
            NTCIP 2301:2001 v01.08

      <DescriptiveName>Controller.baseStandards:text

      <DataConceptType>Data Element"
   ::= { globalConfiguration 4 }
```

2.3 GLOBAL DATABASE MANAGEMENT NODE

```
globalDBManagement OBJECT IDENTIFIER
::= { global 2 }

-- This node is an identifier used to group those objects used to manage a
transaction.
-- A transaction is a SET of one or more database parameters that have
interrelationships with other
-- database parameters, as such a SET for any one of these objects must be
validated against a set of
-- consistency checks and may potentially require the setting of a large
number of objects
-- simultaneously.  Thus, the mode described by these objects allow for such a
large database download.  --   Any device standard that allows this feature
shall define which objects are
-- database parameters versus status or control objects.
```

2.3.1 Database Creation Transaction

```
dbCreateTransaction   OBJECT-TYPE
SYNTAX      INTEGER {   normal (1),
                        transaction (2),
                        verify (3),
                        done (6)
                        }
ACCESS      read-write
STATUS      mandatory

DESCRIPTION
      "<Definition>This object provides transaction control for device
      configuration.  The transaction mode changes the behavior of the
      agent to force buffering of database objects until all related
      database objects have been modified.  In the normal mode, SET
      operations to any database object shall either be stored in a
      device’s database immediately with no regard to whether other
      changes will be made or be rejected (as defined in the device-
      specific Information Profile).  In the transaction mode, SET
      operations to any database object shall be buffered until a verify
      state performs a consistency check.  When the consistency check
      completes, the device automatically transitions to the done state
      where a normal or transaction command may be issued.

      A database object is a user provided piece of setup information
      (or it may be defined in an information profile) that is necessary
      for the proper operation of a device.  It is static in nature in
      that the agent would never change it without direction from the
      management station.  For example, a parameter that defines a
      default mode of operation would be a database object.  A parameter
      that indicates the current state of the device would not be a
      database object.
```

The states and commands are defined as:

NORMAL: SET operations behave as normal SETs and shall have an immediate effect on the value of any database objects used by the device if none of the objects contained in the operation require the use of the transaction mode (as defined in the device-specific Information Profile). A SET operation containing any database object that requires the use of transaction mode shall result in a genErr. This is the default state of this object.

The only command that may be written to dbCreateTransaction while in this state is TRANSACTION. Any other values written to this object in this state shall result in an error response of 'badValue'.

TRANSACTION: A SET operation of one or more database objects that use the same community name as used in the request for the TRANSACTION state are buffered by the agent device for later consistency checks and a normal response is returned. A SET operation of one or more database objects using different community names shall result in a genErr with the index set to zero. A SET operation without a community name field (e.g., an STMP operation) shall be buffered by the agent device for later consistency checks and a normal response is returned. Standard SYNTAX checking shall take place at the time of the SET operation. A transaction may consist of multiple SET operations over multiple frames.

A SET operation for one or more non-database objects shall be processed as normal even if it uses another community name, except for this (i.e., the dbCreateTransaction) object.

A SET operation containing both database and non-database objects shall be processed in full according to these two rules. Thus, if it contains the same community name as used in the request for the TRANSACTION state, the non-database objects shall be stored immediately while the database objects shall be buffered. If it uses a different community name, the entire request will be rejected and a genErr with an index of zero shall be returned.

GET operations on any object shall return the values of the data stored in the controller and shall ignore any values contained in the buffer.

Any valid community name may read this (dbCreateTransaction) object when in this state, but only the community name used to command the object to the transaction mode and the administrator community name can set this object. A set from any other community name shall result in a genErr with an index of zero. The only commands that can be written to dbCreateTransaction while in this state are VERIFY and NORMAL. A VERIFY command will change the state to VERIFY. If a NORMAL command is received, all buffered data is discarded and the state is returned to NORMAL. Any other values written to this object when in this state shall result in an error response of 'badValue'.

VERIFY: Specific database objects are checked for consistency. When consistency checks are complete the device will automatically advance to the DONE state.

The state of dbCreateTransaction cannot be changed when in the VERIFY state. Any values written to this object in this state shall result in an error response of ‘badValue’.

The consistency check analyzes certain critical objects 'in context' and treats them as an interrelated whole rather than separate non-related data items. The consistency check rules are not defined in this standard. They are device and implementation specific. Where applicable, the consistency check rules are defined in application specific object definition standards. A specific implementation may add additional checks beyond those defined in the standards.

A SET operation containing any database objects while in the VERIFY state shall result in a genErr with the index set to zero.

DONE: This state is entered automatically once consistency checks have completed in the VERIFY mode. The value of dbVerifyStatus and dbVerifyError indicate whether the consistency check found any errors.

A SET operation containing any database objects while in the DONE state shall result in a genErr with the index set to zero.

Any valid community name may read this (dbCreateTransaction) object when in this state, but only the community name used to command the object to the transaction mode and the administrator community name can set this object. A set from any other community name shall result in a genErr with an index of zero. The only commands that can be written to dbCreateTransaction while in this state are NORMAL and TRANSACTION. Any other values written to this object in this state shall result in an error response of ‘badValue’.

If a NORMAL command is issued and dbVerifyStatus indicates doneWithNoError, the buffered data is transferred to the device memory and the state is returned to NORMAL. If a NORMAL command is issued and dbVerifyStatus indicates something other than doneWithNoError then the buffered data is discarded and the state is returned to NORMAL.

If a TRANSACTION command is issued, regardless of dbVerifyStatus, no action takes place (the buffered data is not changed) and the TRANSACTION state is re-entered.

		COMMANDED STATE (9)			
		transaction	*verify*	*normal*	*done*
CURRENT STATE	normal	transaction (1)	normal (2)	normal (2)	normal (2)
	transaction	transaction (2)	verify (3)	normal (4)	transaction (2)
	verify (7)	verify (2)	verify(2)	verify (2)	verify (2)
	done (8)	transaction (5)	done(2)	normal (6)	done (2)

```
      Operational procedures and error responses:
            (1)  Once a copy of all database objects is placed in a
            buffer the state is changed to transaction and error
            response indicates noError.  If the operation fails, the
            state remains the same and error response indicates genErr.
            (2)  No action takes place, the state remains the same, but
            response indicates badValue.
            (3)  The state is changed to verify, a consistency check is
            started, and response indicates noError.  Once the
            consistency check is completed, the state automatically
            changes to done.
            (4).  The buffered copy of all database objects is
            discarded, the state is changed to normal, and response
            indicates noError.
            (5)  The buffered copy of all database objects is not
            changed or reloaded, the state is changed to transaction,
            and response indicates noError.
            (6)  If dbVerifyStatus indicates doneWithNoError, then the
            copy of all database objects is transferred to memory, the
            state is changed to normal and response indicates noError.
            If dbVerifyStatus indicates doneWithError then the buffered
            data is discarded, the state is changed to NORMAL, and
            response indicates noError.
            (7)  The state will automatically change to done when the
            consistency check completes.
            (8)  dbVerifyStatus and dbVerifyError are only valid in this
            state.

            (9) All SET operations on this (dbCreateTransaction) parameter
            shall be made using a protocol that uses a community name, or
            equivalent field (e.g., SNMP).
      <DescriptiveName>Transaction.mode:code
      <DataConceptType>Data Element"
DEFVAL      {normal}
::= { globalDBManagement 1 }
```

2.3.2 Database Error Type Parameter

-- This object has been deprecated. See Clause B.1 for more information.

```
dbErrorType OBJECT-TYPE
SYNTAX      INTEGER { tooBig (1),
              noSuchName (2),
              badValue (3),
              readOnly (4),
              genError (5),
              updateError (6),
              noError (7)}
ACCESS      read-only
STATUS      deprecated
DESCRIPTION
"This object returns the current error status of the transaction.  The value
of this
object is only valid when the dbCreateTransaction object is in the Done or
Error state."
::= { globalDBManagement 2 }
```

2.3.3 Database Error ID Parameter

-- This object has been deprecated. See Clause B.1 for more information.

```
dbErrorID  OBJECT-TYPE
SYNTAX       OBJECT IDENTIFIER
ACCESS       read-only
STATUS       deprecated
DESCRIPTION
"This object contains the object identifier of the first object in the
transaction buffer that caused an error while dbCreateTransaction object was
in the Verifying or Updating state.  The value of this object is only valid
when the dbCreateTransaction object is in the Error state.  It is undefined
when the dbCreateTransaction object is in other states."
::= { globalDBManagement 3 }
```

2.3.4 Database Transaction ID Parameter

-- This object has been deprecated. See Clause B.1 for more information.

```
dbTransactionID  OBJECT-TYPE
SYNTAX      INTEGER (0..255)
ACCESS      read-write
STATUS      deprecated
DESCRIPTION
"This object contains the transaction ID value that is to be contained in all
SET operation writes while the dbCreateTransaction object is not in the Normal
state.  During transaction operations every SET command shall begin with a
write to this object with the current value of this object.  If a SET
operation is performed without writing to this object, or with a value that
does not match the current value, then an error response of ‘genError’ shall
be returned.  This mechanism is used to determine that the same management
station that started the transaction is performing the SET operations that are
being buffered or modifying the state of dbCreateTransaction."
::= { globalDBManagement 4 }
```

2.3.5 Database Make ID Parameter

-- This object has been deprecated. See Clause B.1 for more information.

```
dbMakeID OBJECT-TYPE
SYNTAX      INTEGER (0..255)
ACCESS      read-only
STATUS      deprecated
DESCRIPTION
"This object is used to create unique transaction ID’s for management stations
to use when starting transactions using the dbCreateTransaction object.  This
object will be incremented by one every time it is read, so that different
values will be returned for each read.  Management stations wishing to start a
transaction should first read the dbCreateTransaction object to verify that it
is in the Normal state.  If so then the management shall GET dbMakeID to
obtain a transaction ID to use, then SET dbCreateTransaction to startCmd and
dbTransactionID to the value just received.  If the response to the SET
operation is ‘noError’ then the management station has started a transaction.
If the response to the SET operation is ‘genError’ then the management station
should read the dbCreateTransaction and dbTransactionID objects to ensure that
the error was not due to a communications retry.  If the dbCreateTransaction
is in the Transaction state, and the dbTransactionID is the same value
returned by the read of this object, then the management station is the owner
of the transaction.  If the dbTransactionID does not match the value
originally returned by this object, then the management station is not the
owner of the transaction and must wait until the dbCreateTransaction object
returns to the Normal state before attempting to start the transaction."
::= { globalDBManagement  5 }
```

2.3.6 Database Verify Status Parameter

```
dbVerifyStatus   OBJECT-TYPE
SYNTAX      INTEGER {   notDone (1),
                        doneWithError (2),
                        doneWithNoError  (3) }
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>This object indicates the current status of verify
      (consistency checking) processing. The value of this object is
      only meaningful when the dbCreateTransaction object is in the
      Verify or Done state.

      <DescriptiveName>Transaction.verifyStatus:code

      <DataConceptType>Data Element"
::= { globalDBManagement 6 }
```

2.3.7 Database Verify Error Parameter

```
dbVerifyError   OBJECT-TYPE
SYNTAX      OCTET STRING (SIZE (0..255))
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>This object contains a textual description of or a
      reference to an error that was found by the verify (consistency
      checking) processing.  The value of this object is only meaningful
      when the dbCreateTransaction object is in the Done state and the
      dbVerifyStatus object is in the doneWithError state.
      <DescriptiveName>Transaction.errorMessage:text

      <DataConceptType>Data Element"
::= { globalDBManagement 7 }
```

2.4 GLOBAL TIME MANAGEMENT NODE

```
globalTimeManagement   OBJECT IDENTIFIER
::= { global 3 }

-- This node is an identifier used to organize all objects for support of
time-related
-- functions that are common to most device types.
```

2.4.1 Global Time Parameter

```
globalTime  OBJECT-TYPE
SYNTAX      Counter
ACCESS      read-write
STATUS      mandatory
DESCRIPTION

      "<Definition>The number of seconds since the epoch of 00:00:00
      (midnight) January 1, 1970 UTC (a.k.a. Zulu or GMT).

      <DescriptiveName>Controller.globalTime:quantity

      <DataConceptType>Data Element

      <Unit>second"
DEFVAL      { 0 }
::= { globalTimeManagement 1}
```

2.4.2 Global Daylight Savings Parameter

```
globalDaylightSaving   OBJECT-TYPE
SYNTAX   INTEGER {    other (1),
                      disableDST (2),
                      enableUSDST (3),
                      enableEuropeDST (4),
                      enableAustraliaDST (5),
                      enableTasmaniaDST (6),
                      enableEgyptDST (7),
                      enableNamibiaDST (8),
                      enableIraqDST (9),
                      enableMangoliaDST (10),
                      enableIranDST (11),
                      enableFijiDST (12),
                      enableNewZealandDST (13),
                      enableTongaDST (14),
                      enableCubaDST (15),
                      enableBrazilDST (16),
                      enableChileDST (17),
                      enableFalklandsDST (18),
                      enableParaguayDST (19) }
ACCESS      read-write
STATUS      mandatory
DESCRIPTION

      "<Definition>This object specifies if the Daylight Savings Time
      (DST) is enabled, disabled or some other form of daylight savings
      time is active.
```

```
          other -  DST adjustments by a mechanism not defined within this
             standard.
          disableDST - DST clock adjustments shall NOT occur.
          enableUSDST - DST shall begin the first Sunday in April and shall end
             the last Sunday of October.  All changes of time occur at 2:00 AM.
          enableEuropeDST -    DST shall start the last Sunday of March at 2:00
             AM and ends the last Sunday of October at 3:00 AM.
          enableAustraliaDST - DST shall start the last Sunday in October at
             2:00 AM and ends the last Sunday in March at 2:00 AM.
          enableTasmaniaDST - DST shall start the first Sunday in October at
             2:00 AM and ends the last Sunday in March at 3:00 AM.
          enableEgyptDST – DST shall start the last Friday in April and end the
             last Thursday in September.
          enableNamibiaDST – DST shall start the first Sunday in September and
             end the first Sunday in April.
          enableIraqDST – DST shall start on April 1 and end on October 1.
          enableMongoliaDST – DST shall start the last Sunday in March and end
             the last Sunday in September.
          enableIranDST – DST shall start the first day of Farvardin and end
             the first day of Mehr
          enableFijiDST – DST shall start the first Sunday in November and end
             the last Sunday in February.
          enableNewZealandDST – DST shall start the first Sunday in October and
             end the first Sunday on or after March 5th.
          enableTongaDST – DST shall start the first Saturday in October and
             end the first Saturday on or after April 15th.
          enableCubaDST – DST shall start April 1st and end last Sunday in
             October.
          enableBrazilDST – DST shall start the first Sunday in October and end
             the last Sunday in February.
          enableChileDST – DST shall start the first Sunday on or after October
             9th and end the first Sunday on or after March 9th.
          enableFalklandsDST – DST shall start the first Sunday on or after
             September 8th and end the first Sunday on or after April 8th.
          enableParaguayDST – DST shall start the first Sunday in October and
             end the last Saturday in February.

      <DescriptiveName>Controller.daylightSavingsMode:code

      <DataConceptType>Data Element"
REFERENCE
"NEMA TS 2  Clause 3.8.2; http://fatty.law.cornell.edu/uscode/15/260a.html;
http://www.timing.se/Daylight.htm;
http://www.dstc.qut.edu.au/DST/marg/daylight.html#cutoffs;
http://www.dstc.qut.edu.au/DST/marg/daylight.html#cutoffs;
http://webexhibits.org/daylightsaving/g.html "
DEFVAL      { disableDST }
::= { globalTimeManagement 2 }
```

2.4.3 TimeBase Event Scheduler Node

```
timebase OBJECT IDENTIFIER
::= { globalTimeManagement 3 }

-- This node is an identifier used to organize the main objects for event
scheduling.
-- Device type-specific objects (tables) pointed to are defined within the
appropriate MIB.
```

2.4.3.1 Maximum Number of Time Base Schedule Entries Parameter

```
maxTimeBaseScheduleEntries   OBJECT-TYPE
SYNTAX      INTEGER (1..65535)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The value of this object specifies the maximum number
      of different entries supported by the device as shown by the
      number of rows in the timeBaseScheduleTable.

      <DescriptiveName>TimeBaseScheduleTable.maxEntries:quantity

      <DataConceptType>Data Element

      <Unit>TimeBaseScheduleEntry"
::= { timebase 1 }
```

2.4.3.2 Time Base Schedule Table

```
timeBaseScheduleTable   OBJECT-TYPE
SYNTAX      SEQUENCE OF TimeBaseScheduleEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>A table containing the time base schedule parameters
      for the device.  The number of rows in this table shall be equal
      to the maxTimeBaseScheduleEntries object.  The table references
      the appropriate day plan for the device.  The plan is determined
      by comparing the current month (MONTH), day of week (DOW) and date
      of month (DOM) to the appropriate fields. The settings for MONTH,
      DOW and DOM are connected with a logical AND.  In order to
      determine which timebased event to select, determine the event
      which has the most specific date specified.  Select the more
      specific event based on their MONTH settings; if the same, select
      the most specific DOM; if that is still the same, select the most
      specific DOW;  if that’s still the same, the first occurrence
      within the time base event table shall be selected. ’More
      specific’ means the least number of bits set within an object.
      All entries in Time Base Schedule Table are expressed in local
      time and date.  A row in the table may be deactivated by setting
      the Month, Day, Date, or DayPlan parameters to zero (0)

      <DescriptiveName>TimeBaseScheduleTable

      <DataConceptType>Entity Type

      <TableType> static"
::= { timebase 2 }

timeBaseScheduleEntry   OBJECT-TYPE
SYNTAX      TimeBaseScheduleEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>Event Parameters for the time based schedule
      programming of the device.

      <DescriptiveName>TimeBaseSchedule

      <DataConceptType>Entity Type"
```

```
INDEX   { timeBaseScheduleNumber }
::= { timeBaseScheduleTable 1 }

TimeBaseScheduleEntry ::= SEQUENCE {
            timeBaseScheduleNumber     INTEGER,
            timeBaseScheduleMonth      INTEGER,
            timeBaseScheduleDay        INTEGER,
            timeBaseScheduleDate       INTEGER,
            timeBaseScheduleDayPlan    INTEGER }
```

2.4.3.2.1 Time Base Schedule Number Parameter

```
timeBaseScheduleNumber   OBJECT-TYPE
SYNTAX      INTEGER (1..65535 )
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The time base schedule number for objects in this
      row.  The value of this object shall not exceed the value of the
      maxTimeBaseScheduleEntries object.  The activation of a scheduled
      entry shall occur whenever allowed by all other objects within
      this table.

      <DescriptiveName>TimeBaseSchedule.number:identifier

      <DataConceptType>Data Element"
::= { timeBaseScheduleEntry 1 }
```

2.4.3.2.2 Time Base Schedule Month of Year Parameter

```
timeBaseScheduleMonth   OBJECT-TYPE
SYNTAX      INTEGER (0..65535)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>The Month(s) Of the Year that the schedule entry
      shall be allowed.  Each bit represents a specific month.  If the
      bit is set to one (1), then the scheduled entry shall be allowed
      during the associated month.  If the bit is zero (0), then the
      scheduled entry shall not be allowed during the associated month.
      The bits are defined as:
       Bit      Month of Year
       0        Reserved
       1        January
       2        February
       3        March
       4        April
       5        May
       6        June
       7        July
       8        August
       9        September
       10       October
       11       November
       12       December
       13 - 15  Reserved

      Thus, a value of six (6) would indicate that the entry would only
      be allowed during the months of January and February.  A value of
      zero (0) shall indicate that this row has been disabled.
      <DescriptiveName>TimeBaseSchedule.monthMask:code

      <DataConceptType>Data Element"
::= { timeBaseScheduleEntry 2 }
```

2.4.3.2.3 Time Base Schedule Day of Week Parameter

```
timeBaseScheduleDay   OBJECT-TYPE
SYNTAX      INTEGER (0..255)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION

      "<Definition>The Day(s) Of Week that the schedule entry shall be
      allowed. Each bit represents a specific day of the week.  If the
      bit is set to one (1), then the scheduled entry shall be allowed
      during the associated DOW.  If the bit is set to zero (0), then
      the scheduled entry shall not be allowed during the associated
      DOW.  The bits are defined as:
        Bit     Day of Week
        0       Reserved ('Holiday', not defined by this standard)
        1       Sunday
        2       Monday
        3       Tuesday
        4       Wednesday
        5       Thursday
        6       Friday
        7       Saturday

      Thus, a value of six (6) would indicate that the entry would only
      be allowed on Sundays and Mondays.  A value of zero (0) shall
      indicate that this row has been disabled.

      <DescriptiveName>TimeBaseSchedule.dayMask:code

      <DataConceptType>Data Element"
::= { timeBaseScheduleEntry 3 }
```

2.4.3.2.4 Time Base Schedule Date Parameter

```
timeBaseScheduleDate   OBJECT-TYPE
SYNTAX      INTEGER (0..4294967295)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION

      "<Definition>The Day(s) Of a Month that the schedule entry shall
      be allowed.  Each bit represents a specific date of the month.  If
      the bit is set to one (1), then the scheduled entry shall be
      allowed during the associated date.  If the bit is set to zero
      (0), then the scheduled entry shall not be allowed during the
      associated date.  The bits are defined as:
        Bit     Day Number
        0       Reserved
        1       Day 1
        2       Day 2
        ||
        31      Day 31

      Thus, a value of six (6) would indicate that the entry would only
      be allowed on the first and second of the allowed months.  A value
      of zero (0) shall indicate that this row has been disabled.
      <DescriptiveName>TimeBaseSchedule.dateMask:code

      <DataConceptType>Data Element"
::= { timeBaseScheduleEntry 4 }
```

2.4.3.2.5 Time Base Schedule Day Plan Parameter

```
timeBaseScheduleDayPlan   OBJECT-TYPE
SYNTAX      INTEGER (0..255)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION

      "<Definition>This object specifies what Plan number shall be
      associated with this timeBaseScheduleDayPlan -object.  A value of
      zero (0) shall indicate that this row has been disabled.

      <DescriptiveName>TimeBaseSchedule.dayPlan:identifier

      <DataConceptType>Data Element"
::= { timeBaseScheduleEntry 5 }
```

2.4.3.3 Schedule Status Parameter

```
    timeBaseScheduleTableStatus OBJECT-TYPE
       SYNTAX   INTEGER (0..65535)
       ACCESS   read-only
       STATUS   mandatory
       DESCRIPTION
      "<Definition>This object indicates the number of the TimeBaseSchedule
      which is currently selected by the scheduling logic; the device may or
      may not be using the selected schedule.  The value of zero (0) indicates
      that there is no timeBaseScheduleNumber that is currently selected.

      <DescriptiveName>TimeBaseScheduleTable.status:identifier

      <DataConceptType> Data Element"
    ::={timebase 7}
```

2.4.4 Day Plan Parameters

2.4.4.1 Maximum Number of Day Plans - Parameter

```
maxDayPlans   OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION

      "<Definition>The value of this object specifies the maximum, fixed
      number of different timebased Day Plans supported by the device.
      The value of this object represents the number of day plans
      (primary key into the table) available in the
      timeBaseDayPlanTable.

      <DescriptiveName>DayPlanTable.maxDayPlans:quantity

      <DataConceptType>Data Element

      <Unit>DayPlan"
::= { timebase 3 }
```

2.4.4.2 Maximum Number of Day Plan Events - Parameter

```
maxDayPlanEvents   OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The value of this object specifies the fixed number
      of different timebased Day Plan Events within each Day Plan
      supported by the device.  The value of this object represents the
      number of rows (secondary key into the table) available within
      each of the day plans that are available in the
      timeBaseDayPlanTable.  All day plans shall have the same number of
      day plan events available for use.

      <DescriptiveName>DayPlanTable.maxDayPlanEvents:quantity

      <DataConceptType>Data Element

      <Unit>DayPlanEvent"
::= { timebase 4 }
```

2.4.4.3 Day Plan Table

```
timeBaseDayPlanTable OBJECT-TYPE
SYNTAX      SEQUENCE OF TimeBaseDayPlanEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>A table containing day plan numbers, the times when
      to implement them and the associated actions. The number of rows
      in this table shall be equal to the product of the maxDayPlans
      object and the maxDayPlanEvents object.  The dayPlanNumbers within
      this table shall begin with day plan number 1 and increment by one
      to the maxDayPlans.  The dayPlanEventNumbers within this table
      shall begin with day plan event number 1 and increment by one to
      the maxDayPlanEvents.


      This table is always used in association with device-type specific
      objects specifying device-type specific actions such as activating
      a message on a VMS sign or initiating a pattern for a signal
      controller.  A device MIB that defines an action table should
      define the relative priority of the action table as compared to
      the priority of system and other commands.  The device-type
      specific action will only be initiated when (1) the specific
      DayPlan has been activated, (2) the scheduler has sufficient
      priority to override the current operation of the device, and (3)
      at the indicated time.


      After a power recovery or after a change to globalTime, the
      operational mode called for by the scheduler shall be per the last
      event that would have been called by the currently defined
      schedule; the logic will search for all events that may have
      occurred for at least the previous 24 hours.

      <DescriptiveName>DayPlanTable

      <DataConceptType>Entity Type

      <TableType> static"
```

```
::= { timebase 5 }

timeBaseDayPlanEntry   OBJECT-TYPE
SYNTAX      TimeBaseDayPlanEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>A table containing the timebased day plan parameters
      of a device.

      <DescriptiveName>DayPlan

      <DataConceptType>Entity Type"
INDEX { dayPlanNumber, dayPlanEventNumber}
::= { timeBaseDayPlanTable 1 }

TimeBaseDayPlanEntry ::= SEQUENCE {
             dayPlanNumber          INTEGER,
             dayPlanEventNumber     INTEGER,
             dayPlanHour            INTEGER,
             dayPlanMinute          INTEGER,
             dayPlanActionNumberOID OBJECT IDENTIFIER }
```

2.4.4.3.1 Day Plan Number

```
dayPlanNumber   OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>This object specifies the day plan number for objects
      in this row. The value shall not exceed the value of the
      maxDayPlans object.  Day plan numbers are used in the TimeBase
      Event Table to specify day plan numbers to be implemented on
      specific days of the year or as part of the week plans.

      <DescriptiveName>DayPlan.number:identifier

      <DataConceptType>Data Element"
::= { timeBaseDayPlanEntry 1 }
```

2.4.4.3.2 Day Plan Event Number

```
dayPlanEventNumber   OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>This object identifies day plan event number(s) to be
      scheduled on a specific day plan number.  Several different events
      can be scheduled to take place during a day, and each of these
      events is one entry or row within a specified day plan number.
      The total number of events for one day plan shall not exceed the
      value of the maxDayPlanEvents object. If multiple non-conflicting
      events occur at the same time, they shall be logically executed in
      order of their dayPlanEventNumber with the lowest number occurring
      first.  An implementation shall omit lower number actions that are
      in conflict with higher number actions at the same time.
      <DescriptiveName>DayPlanEvent.number:identifier

      <DataConceptType>Data Element"
::= { timeBaseDayPlanEntry 2 }
```

2.4.4.3.3 Day Plan Hour Parameter

```
dayPlanHour   OBJECT-TYPE
SYNTAX      INTEGER (0..23)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>The Hour of day, as measured by the
      controllerLocalTime object, that the associated event shall become
      active.

      <DescriptiveName>DayPlanEvent.hour:number

      <DataConceptType>Data Element"
DEFVAL      {0}
::= { timeBaseDayPlanEntry 3 }
```

2.4.4.3.4 Day Plan Minute Parameter

```
dayPlanMinute   OBJECT-TYPE
SYNTAX      INTEGER (0..59)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>The Minute of the hour (defined in the dayPlanHour),
      as measured by the controllerLocalTime object, that the associated
      event shall become active.

      <DescriptiveName>DayPlanEvent.minute:number

      <DataConceptType>Data Element"
DEFVAL      {0}
::= { timeBaseDayPlanEntry 4 }
```

2.4.4.3.5 Day Plan Action Number OID Parameter

```
dayPlanActionNumberOID   OBJECT-TYPE
SYNTAX      OBJECT IDENTIFIER
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>This object provides a reference to the device-type
      specific action that shall be executed.  The object shall
      reference the action by its associated object identifier,
      including its instance (i.e., the full OID of the scalar or
      columnar object).  Only objects whose description field explicitly
      states that they may be called by the action table may be
      referenced.  If a management system attempts to set this value to
      any other object identifier, the device shall respond with a
      genErr.

      Any object allowing the action table to reference it shall define
      precisely what action will take place when it is activated and
      whether the action is transitionary or continuous until
      deactivated.  The object shall also define what, if any,
      restrictions may be placed on other operations the device may be
      able to perform.

      If the action to be performed is defined by a row of a table, one
      of the index columns should be identified as the explicit object
      that is referenced.

      <DescriptiveName>DayPlanEvent.action:identifier

      <DataConceptType>Data Element"
DEFVAL      {null}
::= { timeBaseDayPlanEntry 5 }
```

2.4.4.4 Day Plan Status Parameter

```
dayPlanStatus   OBJECT-TYPE
SYNTAX      INTEGER (0..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>This object indicates the current value of the active
      day PlanNumber-object.  A value of zero (0) indicates that there
      is no dayPlanNumber that is currently active.

      <DescriptiveName>DayPlanTable.activeDayPlan:identifier

      <DataConceptType>Data Element"
::= { timebase 6 }
```

2.4.5 Global Local Time Differential Parameter

-- This object has been deprecated. See Clause B.2 for more information.

```
globalLocalTimeDifferential   OBJECT-TYPE
SYNTAX      INTEGER (-43200..43200)
ACCESS      read-write
STATUS      deprecated
DESCRIPTION
      "Indicates the number of seconds offset between local time and
      GMT.  Positive values indicate local times in the Eastern
      Hemisphere up to the International Date Line and negative values
      indicate local times in the Western Hemisphere back to the
      International Date Line.  If one of the daylight savings times is
      activated, this value will change automatically at the referenced
      time. For example, Central Standard Time (CST) is -21600 and
      Central Daylight Time (CDT) is -18000."
::= { globalTimeManagement 4 }
```

2.4.6 Standard Time Zone Parameter

```
controllerStandardTimeZone   OBJECT-TYPE
SYNTAX      INTEGER (-43200..43200)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>Indicates the number of seconds offset between local
      Standard Time and GMT.  Positive values indicate local times in
      the Eastern Hemisphere up to the International Date Line and
      negative values indicate local times in the Western Hemisphere
      back to the International Date Line.  This value does not change
      in response to a daylight savings time event.

      <DescriptiveName>Controller.standardTimeZone:quantity

      <DataConceptType>Data Element

      <Unit>second"
DEFVAL      {0}
::= { globalTimeManagement 5 }
```

2.4.7 Local Time Parameter

```
controllerLocalTime   OBJECT-TYPE
SYNTAX      Counter
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition> The current local time expressed in seconds since
      00:00:00 (midnight) January 1, 1970 of the same time offset.  This
      value changes by 3600 seconds in response to a daylight savings
      time event.

      <DescriptiveName>Controller.localTime:quantity

      <Unit>second

      <DataConceptType>Data Element"
::= { globalTimeManagement 6 }
```

2.5 REPORT PARAMETER NODE

NOTE—These objects will be moved to a future version of NTCIP 1103.

```
globalReport OBJECT IDENTIFIER
::= { global 4 }

-- This node is an identifier used to organize all objects for support of
report functions
-- that are common to most device types.

-- NOTE—The event class table is presented first in order to ease the
-- readability of the standard; however, the node numbers assigned to this
-- table reflect the original node numbering used in the original 1996
-- specification in order to preserve backwards compatibility with existing
-- systems.
```

2.5.1 Maximum Event Classes Parameter

```
maxEventClasses OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition> The object defines the number of rows in the
      eventClassTable that this device supports.  This is a static
      table.

      <DescriptiveName>EventClassTable.maxEventClasses:quantity

      <DataConceptType>Data Element
      <Unit>EventClass"
::= { globalReport 5 }
```

2.5.2 Event Class Table

```
eventClassTable OBJECT-TYPE
SYNTAX      SEQUENCE OF EventClassEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>This table is used to configure event logging limits
      and log table maintenance.

      <DescriptiveName>EventClassTable

      <DataConceptType>Entity Type

      <TableType> static"
::= { globalReport 6 }

eventClassEntry  OBJECT-TYPE
SYNTAX      EventClassEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>This defines a row in the Event Class Table

      <DescriptiveName>EventClass

      <DataConceptType>Entity Type"
INDEX { eventClassNumber }
::= { eventClassTable 1 }

EventClassEntry ::= SEQUENCE {
      eventClassNumber        INTEGER,
      eventClassLimit         INTEGER,
      eventClassClearTime           Counter,
      eventClassDescription         OCTET STRING,
      eventClassNumRowsInLog  INTEGER,
      eventClassNumEvents           INTEGER }
```

2.5.2.1 Event Class Number Parameter

```
eventClassNumber OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>This is a class value that is to be configured.

      <DescriptiveName>EventClass.number:identifier

      <DataConceptType>Data Element"
::= { eventClassEntry 1 }
```

2.5.2.2 Event Class Limit Parameter

```
eventClassLimit OBJECT-TYPE
SYNTAX      INTEGER (0..255)
ACCESS      read-write
STATUS      mandatory
```

```
DESCRIPTION
      "<Definition>This object specifies the maximum number of events of
      the associated class to store in the log.  Once the limit is
      reached, the oldest entry of the matching class will be
      overwritten by any new entry of the same class.  If the value of
      this object is set to a number smaller than the current number of
      rows within this class in the eventLogTable, then the oldest
      entries shall be lost/deleted. The sum of all event class limits
      shall not exceed the maxEventLogSize object; if a SET operation to
      this object causes the sum of eventClassLimit objects to exceed
      maxEventLogSize, then the agent shall respond with a genErr.

      <DescriptiveName>EventClass.eventLimit:quantity

      <DataConceptType>Data Element

      <Unit>Event"
::= { eventClassEntry 2 }
```

2.5.2.3 Event Class Clear Time Parameter

```
eventClassClearTime OBJECT-TYPE
SYNTAX      Counter
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>This object is used to clear multiple event log
      entries from the eventLogTable.  All events of this class that
      have an eventLogTime equal to or less than this object shall be
      cleared from the eventLogTable.  If this object has a value
      greater than the current value of globalTime, it shall prevent the
      logging of any events of this class.

      <DescriptiveName>EventClass.clearTime:quantity

      <DataConceptType>Data Element

      <Unit>second"
DEFVAL      {0}
::= { eventClassEntry 3 }
```

2.5.2.4 Event Class Description Parameter

```
eventClassDescription OBJECT-TYPE
SYNTAX      OCTET STRING
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>This object specifies a description of the class in
      ASCII characters.

      <DescriptiveName>EventClass.description:text

      <DataConceptType>Data Element"
::= { eventClassEntry 4 }
```

2.5.2.5 Event Class Number of Rows in Event Log Table Parameter

```
eventClassNumRowsInLog OBJECT-TYPE
SYNTAX      INTEGER (0..255)
ACCESS      read-only
```

```
STATUS      mandatory
DESCRIPTION

      "<Definition>The number of rows for this class that currently
      exist in the eventLogTable.

      <DescriptiveName>EventClass.currentEntries:quantity

      <DataConceptType>Data Element

      <Unit>Event"
::= { eventClassEntry 5 }
```

2.5.2.6 Class Event Log Counter Parameter

```
eventClassNumEvents OBJECT-TYPE
SYNTAX      INTEGER (0..65535)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION

      "<Definition> This object is a counter that gets incremented every
      time an event occurs for this class; it shall initialize to zero
      at power up.  The value shall roll over each time it exceeds the
      maximum of 65535.  This value shall not be affected by logic
      related to the eventClassLimit or eventClassClearTime objects.

      <DescriptiveName>EventClass.eventCounter:quantity

      <DataConceptType>Data Element

      <Unit>Events"
::= { eventClassEntry 6 }
```

2.5.3 Maximum Event Log Configurations Parameter

```
maxEventLogConfigs   OBJECT-TYPE
SYNTAX      INTEGER (1..65535)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION

      "<Definition>The number of rows that exist in the static
      eventLogConfig table for this device.

      <DescriptiveName>EventTypeTable.maxEventTypes:quantity

      <DataConceptType>Data Element

      <Unit>EventType"
::= { globalReport 1}
```

2.5.4 Event Log Configuration Table

```
eventLogConfigTable   OBJECT-TYPE
SYNTAX      SEQUENCE OF EventLogConfigEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION

      "<Definition>A table containing Event Log Configuration
      information. The number of rows in this table is equal to the
      maxEventLogConfigs object.  This table defines the parameters that
      the device will monitor to create an event.

      <DescriptiveName>EventTypeTable
      <DataConceptType>Entity Type

      <TableType> static"
::= { globalReport 2 }

eventLogConfigEntry OBJECT-TYPE
SYNTAX      EventLogConfigEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>This object defines an entry in the event log
      configuration table.

      <DescriptiveName>EventType

      <DataConceptType>Entity Type"
INDEX { eventConfigID }
::= { eventLogConfigTable 1 }

EventLogConfigEntry ::= SEQUENCE {
      eventConfigID           INTEGER,
      eventConfigClass        INTEGER,
      eventConfigMode         INTEGER,
      eventConfigCompareValue INTEGER,
      eventConfigCompareValue2      INTEGER,
      eventConfigCompareOID   OBJECT IDENTIFIER,
      eventConfigLogOID       OBJECT IDENTIFIER,
      eventConfigAction       INTEGER,
      eventConfigStatus       INTEGER }
```

2.5.4.1 Event Log Configuration ID Parameter

```
eventConfigID  OBJECT-TYPE
SYNTAX      INTEGER (1..65535)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>This object contains the row number which is used to
      identify the event associated with this row in the
      eventLogConfigTable.  The number of event IDs shall not exceed the
      value indicated in the maxEventLogConfigs object.

      <DescriptiveName>EventType.identifier:identifier

      <DataConceptType>Data Element"
::= { eventLogConfigEntry 1 }
```

2.5.4.2 Event Log Configuration Class Parameter

```
eventConfigClass OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>This object contains the class value to assign to the
      event associated with this row in the event configuration table.
      This value is used in the event log table to organize various
      events defined in this table into logical groupings.  This value
      shall not exceed the maxEventClasses object value.
      NOTE1—See NTCIP 1103 for additional requirements related to traps.

      NOTE2—The event cannot be logged if the EventClass has an
      eventClassLimit of zero (0),

      <DescriptiveName>EventType.class:identifier

      <DataConceptType>Data Element "
DEFVAL      {1}
::= { eventLogConfigEntry 2 }
```

2.5.4.3 Event Log Configuration Mode Parameter

```
eventConfigMode OBJECT-TYPE
SYNTAX      INTEGER { other (1),
                     onChange (2),
                     greaterThanValue (3),
                     smallerThanValue (4),
                     hysteresisBound (5),
                     periodic (6),
                     andedWithValue (7) }
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>This object specifies the mode of operation for this
      event. The modes are defined as follows:

      Value             Description
      other             the event mode of operation is not described in this
                        standard, refer to the device manual.
      onChange          create a log entry when the object value referenced by
                        eventConfigCompareOID changes.  The values of
                        eventConfigCompareValue and eventConfigCompareValue2
                        are ignored in this mode.
      greaterThanValue  create a log entry when the object value referenced by
                        eventConfigCompareOID becomes greater than the value
                        of eventConfigCompareValue for the time (tenth
                        seconds) defined by eventConfigCompareValue2 (zero
                        means immediate logging).
      smallerThanValue  create a log entry when the object value referenced by
                        eventConfigCompareOID becomes less than the value of
                        eventConfigCompareValue for the time (tenth seconds)
                        defined by eventConfigCompareValue2 (zero means
                        immediate logging).
      hysteresisBound   create a log entry when the object value referenced by
                        eventConfigCompareOID becomes less than or greater
                        than the bound values.  The lowerbound value is the
                        lower value of eventConfigCompareValue and
                        eventConfigCompareValue2; the upperbound value is the
                        higher value of the two values.

                        When the object value becomes greater than the
                        upper bound value, subsequent logging of
                        upperbound conditions shall not occur until the
                        object value becomes less than the lower bound
                        value.

                        When the object value becomes less than the
                        lower bound value, subsequent logging of
                        lowerbound conditions shall not occur until the
                        object value becomes greater than the upper
                        bound value.
      periodic          create a log entry every x seconds, where x is defined
                        by the value stored in eventConfigCompareValue.  The
                        values stored in eventConfigCompareValue2 and
                        eventConfigCompareOID are ignored in this mode.
      andedWithValue    create a log entry when the object value
                        referenced by eventConfigCompareOID ANDED with
                        the value of eventConfigCompareValue is NOT
                        equal to zero for the time (tenth seconds)
                        defined by eventConfigCompareValue2 (zero means
                        immediate logging).  This allows monitoring of a
                        specific bit; the condition becomes true anytime
                        that any one of the selected bits become true.


      <DescriptiveName>EventType.mode:code

      <DataConceptType>Data Element"
DEFVAL      {onChange}
::= { eventLogConfigEntry 3 }
```

2.5.4.4 Event Log Configuration Compare Value Parameter

```
eventConfigCompareValue OBJECT-TYPE
SYNTAX      INTEGER
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>This object contains the comparision value to use
      with eventConfigMode values (greaterThanValue, smallerThanValue,
      hysteresisBound ).  No value within this object is necessary when
      the eventConfigMode-object has the value onChange (2).

      <DescriptiveName>EventType.compareValue:number

      <DataConceptType>Data Element"
DEFVAL      {0}
::= { eventLogConfigEntry 4 }
```

2.5.4.5 Event Log Configuration Compare Value 2 Parameter

```
eventConfigCompareValue2 OBJECT-TYPE
SYNTAX      INTEGER
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>If the eventConfigMode is set to hysteresisBound,
      this object specifies the second comparison value for the
      hysteresis.  If the eventConfigMode is set to greaterThanValue or
      smallerThanValue, this object specifies the time (in tenth of
      seconds) for which the comparison must be true prior to the event
      condition becoming true (the value shall be reported in tenths of
      seconds, per the original specification, but the accurracy shall
      be plus/minus one second due to implementation experience).  If
      the eventConfigMode is set to onChange or periodic, the value of
      this object shall be ignored.

      <DescriptiveName>EventType.compareValue2:number
      <DataConceptType>Data Element"
DEFVAL      {0}
::= { eventLogConfigEntry 5 }
```

2.5.4.6 Event Log Configuration Compare Object Identifier Parameter

```
eventConfigCompareOID OBJECT-TYPE
SYNTAX      OBJECT IDENTIFIER
ACCESS      read-write
STATUS      mandatory
DESCRIPTION

      "<Definition>This object contains the object identifier which
      references the value against which the comparison is made.  If the
      eventConfigMode is set to periodic, the value of this object shall
      be ignored.  If the eventConfigMode is set to greaterThanValue,
      smallerThanValue or hysteresisBound, this object must reference an
      object whose SYNTAX resolves to a ranged or unranged INTEGER.  As
      with all other objects that are sub-ranged by a given
      implementation, an agent should return a badValue error if it
      receives a set command indicating a OID which is not supported by
      the implementation or which is not null.

      <DescriptiveName>EventType.compareObject:identifier

      <DataConceptType>Data Element"
DEFVAL      {null}
::= { eventLogConfigEntry 6 }
```

2.5.4.7 Event Log Configuration Log Object Identifier Parameter

```
eventConfigLogOID OBJECT-TYPE
SYNTAX      OBJECT IDENTIFIER
ACCESS      read-write
STATUS      mandatory
DESCRIPTION

      "<Definition>This object contains the object identifier which
      indicates what value to log when a condition or event occurs
      (e.g., log the phase display when the watchdog alarm status
      changes). As with all other objects that are sub-ranged by a given
      implementation, an agent should return a badValue error if it
      receives a set command indicating a value which is not supported
      by the implementation.  The valid value range of this object shall
      not include any values, other than null, that do not correspond to
      objects that may exist within the agent, although it may be
      further restricted.

      The valid value range of this object shall not include objects under the
      following nodes:
         Security - { nema transportation devices global security }
         CHAP - { nema transportation protocols layers chap }


      <DescriptiveName>EventType.logObject:identifier

      <DataConceptType>Data Element"
DEFVAL      {null}
::= { eventLogConfigEntry 7 }
```

2.5.4.8 Event Log Configuration Action Parameter

```
eventConfigAction  OBJECT-TYPE
SYNTAX      INTEGER { other (1),
                     disabled (2),
                     log (3)
                     }
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>The value of this object indicates what action shall
      take place when this event occurs.

      disabled  - no entry will be recorded due to this event.
      log - an entry will be recorded in the event log table when this event
      occurs.

      NOTE—See NTCIP 1103 for additional requirements related to traps.


      <DescriptiveName>EventType.action:code

      <DataConceptType>Data Element"
DEFVAL      {disabled}
::= { eventLogConfigEntry 8 }
```

2.5.4.9 Event Log Configuration Status Parameter

```
eventConfigStatus  OBJECT-TYPE
SYNTAX      INTEGER { other (1),
                     disabled (2),
                     log (3),
                     error (4)
                     }
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The value of this object indicates the current status
      of the configured event. Upon setting any object in this row of
      the eventLogConfigTable, the agent will determine if the setting
      is valid and will set this object to one of the following states:

      other indicates that the action is successfully set to a mode
               other than that defined in this standard

      disabled indicates that the action is set to disabled

      log   indicates that the action is successfully set to the log state
               after passing consistency checks.

      error indicates that the requested action could not be implemented due
               to a consistency check

      <DescriptiveName>EventType.status:code

      <DataConceptType>Data Element"
::= { eventLogConfigEntry 9 }
```

2.5.5 Maximum Event Log Size Parameter

```
maxEventLogSize   OBJECT-TYPE
SYNTAX      INTEGER (1..65535)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The maximum, fixed number of rows that can be
      utilized within the eventLogTable.

      <DescriptiveName>EventTable.maxEventLogSize:quantity

      <DataConceptType>Data Element

      <Unit>Event"
::= { globalReport 3}
```

2.5.6 Event Log Table

```
eventLogTable   OBJECT-TYPE
SYNTAX      SEQUENCE OF EventLogEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>A table containing Event History data collected. A
      request for an object from a row that has not been instantiated or
      has been cleared shall return a noSuchName error.

      <DescriptiveName>EventTable

      <DataConceptType>Entity Type

      <TableType> dynamic status"
::= { globalReport 4 }

eventLogEntry OBJECT-TYPE
SYNTAX      EventLogEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>This object defines an entry in the event log table

      <DescriptiveName>Event

      <DataConceptType>Entity Type"
INDEX { eventLogClass, eventLogNumber }
::= { eventLogTable 1 }

EventLogEntry ::= SEQUENCE {
      eventLogClass     INTEGER,
      eventLogNumber    INTEGER,
      eventLogID     INTEGER,
      eventLogTime      Counter,
      eventLogValue     Opaque }
```

2.5.6.1 Event Log Class Parameter

```
eventLogClass OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
```

```
      "<Definition>This object contains the class of the associated
      event as defined in the eventLogConfig Table.

      <DescriptiveName>ClassLog.class:identifier

      <DataConceptType>Data Element"
::= { eventLogEntry 1 }
```

2.5.6.2 Event Log Number Parameter

```
eventLogNumber OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The event number within this class for this event.
      Event numbers shall be assigned starting at 1 and shall increase
      to the value specified by the associated eventClassLimit for the
      class associated with the rows.  Events shall maintain a
      chronological ordering in the table with the oldest event of a
      class occupying the row with eventNumber = 1, and subsequent
      events filling subsequent rows. This ordering shall be maintained
      for those rows still remaining when events are cleared.

      <DescriptiveName>Event.number:identifier

      <DataConceptType>Data Element"
::= { eventLogEntry 2 }
```

2.5.6.3 Event Log ID Parameter

```
eventLogID  OBJECT-TYPE
SYNTAX      INTEGER (1..65535)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>This object contains the event configuration ID (from
      the eventLogConfigTable) that caused this table entry.  It
      indicates the row in the eventLogConfig table reponsible for this
      event entry.

      <DescriptiveName>Event.type:identifier

      <DataConceptType>Data Element"
::= { eventLogEntry 3 }
```

2.5.6.4 Event Log Time Parameter

```
eventLogTime OBJECT-TYPE
SYNTAX      Counter
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The time that the event was detected.  If the device
      supports the globalTime object, the value shall reflect the value
      of globalTime when the event occurred, otherwise this shall be the
      time in seconds since the device powered up. The event shall be
      detected and timestamped within one second from the event becoming
      true.  The event shall be logged in the table within five seconds
      of the event being detected. These timing resolutions may be
      modified by a device profile.
      <DescriptiveName>Event.logTime:quantity

      <DataConceptType>Data Element

      <Unit>second"
::= { eventLogEntry 4 }
```

2.5.6.5 Event Log Value Parameter

```
eventLogValue  OBJECT-TYPE
SYNTAX      Opaque
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The value of this object is set to the BER encoding
      of the value referenced by the eventConfigLogOID of the associated
      eventLogID when the event was logged.  Its length is variable.
      The value shall not contain any padding characters either before
      or after the values.

      NOTE—Opaque objects are doubly wrapped.  For SNMP operations,
      which use BER, this would be {type, length, {type, length,
      value}}.  For example, a zero-length octet string, would be
      encoded in BER as 0x44 02 04 00.  For STMP or SFMP operations,
      which use OER, this would be { length, {type, length, value}}.
      For example, the same example would be encoded in OER as 0x02 04
      00.

      <DescriptiveName>Event.logValue:frame

      <DataConceptType>Data Element"
::= { eventLogEntry 5 }
```

2.5.7 Total Event Log Counter Parameter

```
numEvents OBJECT-TYPE
SYNTAX      INTEGER (0..65535)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition> This object is a counter that gets incremented every
      time an event occurs and shall initialize to zero at power up.
      The value shall roll over each time it exceeds the maximum of
      65535.  This value shall not be affected by logic related to the
      eventClassLimit or eventClassClearTime objects.

      <DescriptiveName>EventTable.numEvents:quantity

      <DataConceptType>Data Element

      <Unit>Events"
::= { globalReport 7 }
```

2.6 PMPP OBJECT NODE

```
-- NOTE: In the future, these objects will be moved to NTCIP 2101.

profilesPMPP OBJECT IDENTIFIER
::= { profiles 3 }
```

```
-- This node is an identifier used to group all objects for support of the
PMPP function that
-- are common to all device types.  The objects under this node are placed
under the
-- Protocols\Profiles\PMPP subtree within the NEMA node, but they have been
listed here due to the lack
-- of a separate document that lists these objects.
```

2.6.1 Maximum HDLC Group Address Parameter

```
maxGroupAddresses OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The maximum number of group addresses this device
      supports. This object indicates the maximum number of rows in the
      hdlcGroupAddressTable.

      <DescriptiveName>Secondary.maxGroupAddresses:quantity

      <DataConceptType>Data Element

      <Unit>address"
::= {profilesPMPP 1 }
```

2.6.2 HDLC Group Address Table

```
hdlcGroupAddressTable OBJECT-TYPE
SYNTAX      SEQUENCE OF HdlcGroupAddressEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition> A table containing group addresses at which a device
      may receive frames.

      <DescriptiveName> HdlcGroupAddressTable

      <DataConceptType> Entity Type

      <TableType> static"
::= { profilesPMPP 2 }

hdlcGroupAddressEntry   OBJECT-TYPE
SYNTAX      HdlcGroupAddressEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition> An entry in the group address table that contains a
      device’s data link layer group address at which it will accept
      frames.

      <DescriptiveName> HdlcGroupAddress

      <DataConceptType> Entity Type"
INDEX { hdlcGroupAddressIndex }
::= { hdlcGroupAddressTable 1 }

HdlcGroupAddressEntry ::= SEQUENCE {
   hdlcGroupAddressIndex            INTEGER,
   hdlcGroupAddress                 INTEGER, -- deprecated
   hdlcGroupAddressNumber           INTEGER }
```

2.6.2.1 HDLC Group Address Index Parameter

```
hdlcGroupAddressIndex   OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>The index number for the group address in this row.

      <DescriptiveName>GroupAddress.index:identifier

      <DataConceptType>Data Element"
::= { hdlcGroupAddressEntry 1}
```

2.6.2.2 HDLC Group Address Parameter

-- This object has been deprecated. See Clause B.3 for more information.

```
hdlcGroupAddress  OBJECT-TYPE
SYNTAX      INTEGER
ACCESS      read-write
STATUS       deprecated
DESCRIPTION
      "A group address for the data link layer. For PMPP, the syntax is
      an 8 or 16 bit entry with the second low order bit set to a one
      indicating that this is a group address."
REFERENCE
"NEMA TS 3.3 Clause 3.3.3.1"
::= { hdlcGroupAddressEntry 2}
```

2.6.2.3 HDLC Group Address Number Parameter

```
hdlcGroupAddressNumber  OBJECT-TYPE
SYNTAX      INTEGER (0..62)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>A group address number prior to any encoding for the
      data link layer. The address of 63 is reserved for the all
      stations address. The value of zero (0) shall disable this row of
      the table.

      NOTE—In PMPP all group addresses are encoded in one byte.

      <DescriptiveName>GroupAddress.address:number

      <DataConceptType>Data Element"
REFERENCE
"NTCIP 2101"
DEFVAL  { 0 }
::= { hdlcGroupAddressEntry 3}
```

2.7 SECURITY NODE

-- NOTE: In the future, these objects will be moved to NTCIP 1103.
-- however, these objects are expected to remain under the global 5 node of the
-- ISO tree.

```
security OBJECT IDENTIFIER ::= {global 5}
-- This node is an identifier used to group all objects related to the
-- assignment of community names and the access rights they provide.
```

2.7.1 Community Name Administrator Parameter

```
communityNameAdmin OBJECT-TYPE
     SYNTAX    OCTET STRING (SIZE(8..16))
     ACCESS    read-write
     STATUS    mandatory
     DESCRIPTION
      "<Definition>This object is the community name that must be used to
      specifically gain access to information under the security node.  A
      message with this value in the community name field of an SNMP message
      has user read-write access to the security node objects and all other
      objects implemented in the device.  The syntax is defined as an OCTET
      STRING and therefore any character can have a value of 0..255.

      <DescriptiveName>CommunityNames.admin:text

      <DataConceptType>Data Element"
     DEFVAL { "administrator" }
     ::= { security 1 }
```

2.7.2 Maximum Community Names Parameter

```
communityNamesMax     OBJECT-TYPE
     SYNTAX    INTEGER (1..255)
     ACCESS    read-only
     STATUS    mandatory
     DESCRIPTION
      "<Definition>This object specifies the maximum number of rows that are
      implemented in the community name table.

      <DescriptiveName>CommunityNames.maximumNames:quantity

      <DataConceptType>Data Element"
     ::= { security 2 }
```

2.7.3 Community Names Table

```
communityNameTable  OBJECT-TYPE
     SYNTAX    SEQUENCE OF CommunityNameTableEntry
     ACCESS    not-accessible
     STATUS    mandatory
     DESCRIPTION
      "<Definition>This table defines the community names that can appear in
      the community name field of the SNMP message and access privileges
      associated with that community name.

      <DescriptiveName>CommunityNameTable

      <DataConceptType>Entity Type

      <TabelType>Static"
     ::= { security 3 }
```

```
communityNameTableEntry  OBJECT-TYPE
     SYNTAX    CommunityNameTableEntry
     ACCESS    not-accessible
     STATUS    mandatory
     DESCRIPTION
      "<Definition>This is the row index of information in the community name
      table.

      <DescriptiveName>CommunityName

      <DataConceptType>Entity Type"
     INDEX     { communityNameIndex }
     ::= { communityNameTable 1}

CommunityNameTableEntry::=SEQUENCE
     {  communityNameIndex       INTEGER,
        communityNameUser        OCTET STRING,
        communityNameAccessMask  Gauge
     }
```

2.7.3.1 Community Name Index Parameter

```
communityNameIndex  OBJECT-TYPE
     SYNTAX  INTEGER (1..255)
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
      "<Definition>This object defines the row index into the
      communityNameTable. This value shall not exceed the communityNamesMax
      object value.

      <DescriptiveName>CommunityName.index:identifier

      <DataConceptType>Data Element"
     ::=  { communityNameTableEntry 1 }
```

2.7.3.2 User Community Name Parameter

```
communityNameUser  OBJECT-TYPE
     SYNTAX  OCTET STRING (SIZE(6..16))
     ACCESS  read-write
     STATUS  mandatory
     DESCRIPTION
      "<Definition>This object defines a community name value that a security
      administrator can assign user read-write access to information (other
      than security) in a device. A message with this value in the community
      name field of an SNMP/SFMP message has user access rights as defined in
      the communityNameAccessMask.  The syntax is defined as an OCTET STRING
      and therefore any character can have a value of 0..255.

      <DescriptiveName>CommunityName.user:text

      <DataConceptType>Data Element"
     DEFVAL  { "public" }
     ::=  { communityNameTableEntry 2 }
```

2.7.3.3 User Community Name Mask Parameter

```
communityNameAccessMask OBJECT-TYPE
     SYNTAX  Gauge
     ACCESS  read-write
     STATUS  mandatory
     DESCRIPTION
      "<Definition>This object defines a 32 bit mask that can be used to
      associate 'write access' with a community name.  A value of 0x00 00 00
      00 grants the community name user read-only access and overrides any
      individual object's read-write access clause.  A value of 0xFF FF FF FF
      grants the community name user read-write access and an individual
      object's read-write access clause applies.  Values other than 0x00 00 00
      00 and 0xFF FF FF FF are implementation specific and may limit viewing
      and/or accessing the information in a device.

      <DescriptiveName>CommunityName.accessMask:code

      <DataConceptType>Data Element"
     DEFVAL  { 4294967295 }
     ::= { communityNameTableEntry 3 }
```

2.8 AUXILIARY I/O OBJECTS

```
auxIO  OBJECT IDENTIFIER ::= { global 7}
-- This node is an identifier used to group all objects supporting auxiliary
I/O functions
-- NOTE: These objects were formerly located under the experimental node.
```

2.8.1 Maximum Number of Digital Auxiliary IOs Parameter

```
auxIOTableNumDigitalPorts OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION

      "<Definition>The number of rows contained in the ‘auxIOTable’ with
      the auxPortType set to ‘digital’.

      <DescriptiveName>AuxIOTable.maxDigitalPorts:quantity

      <DataConceptType>Data Element

      <Unit>port"
::= {auxIO 1}
```

2.8.2 Maximum Number of Analog Auxiliary IOs Parameter

```
auxIOTableNumAnalogPorts OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION

      "<Definition>The number of rows contained in the ‘auxIOTable’ with
      the auxPortType set to ‘analog’.

      <DescriptiveName>AuxIOTable.maxAnalogPorts:quantity
      <DataConceptType>Data Element

      <Unit>port"
::= {auxIO 2}
```

2.8.3 Auxiliary IO Table Parameter

```
auxIOTable   OBJECT-TYPE
SYNTAX      SEQUENCE OF AuxIOEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>A table providing the means to access any non-
      mission-critical or safety-related auxiliary I/O of the
      Controller, this includes reading inputs and setting outputs.  A
      maximum of 255 auxiliary IOs can be defined for all, digital,
      analog or other types of ports.  This table shall not be used to
      control or monitor any safety related equipment.  The user should
      be aware that the electrical levels used by the ports are not
      standardized by these objects; such information should be
      contained in the hardware manual.

      <DescriptiveName>AuxIOTable

      <DataConceptType>Entity Type

      <TableType> static"
::= { auxIO 3}

auxIOEntry OBJECT-TYPE
SYNTAX      AuxIOEntry
ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
      "<Definition>Parameters of the auxiliary I/O table.

      <DescriptiveName>AuxIOPort

      <DataConceptType>Entity Type"
INDEX {auxIOPortType, auxIOPortNumber}
::={auxIOTable 1}

AuxIOEntry ::= SEQUENCE {
   auxIOPortType                    INTEGER,
   auxIOPortNumber                  INTEGER,
   auxIOPortDescription             DisplayString,
   auxIOPortResolution              INTEGER,
   auxIOPortValue                   INTEGER,
   auxIOPortDirection               INTEGER,
   auxIOPortLastCommandedState      INTEGER
   }
```

2.8.3.1 Auxiliary Port Type Parameter

```
auxIOPortType OBJECT-TYPE
SYNTAX      INTEGER{
                   other (1),
                   analog (2),
                   digital (3)
```

```
                     }
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>Indicates the type of auxiliary I/O, which can be
      analog, digital or other.

      <DescriptiveName>AuxIOPort.type:code

      <DataConceptType>Data Element"
::= {auxIOEntry 1}
```

2.8.3.2 Auxiliary Port Number Parameter

```
auxIOPortNumber OBJECT-TYPE
SYNTAX      INTEGER (1..255)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>Indicates the port number for the associated port
      type.  Port numbers are used sequentially from one to max for each
      port type.  There can be a port 1 for analog port and port 1 for
      digital port.

      <DescriptiveName>AuxIOPort.number:identifier

      <DataConceptType>Data Element"
::= {auxIOEntry 2}
```

2.8.3.3 Auxiliary Description Parameter

```
auxIOPortDescription OBJECT-TYPE
SYNTAX      DisplayString (SIZE (0..255))
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>Informational text field describing the device at the
      associated auxiliary I/O

      <DescriptiveName>AuxIOPort.description:text

      <DataConceptType>Data Element"
::= {auxIOEntry 3}
```

2.8.3.4 Auxiliary Resolution Parameter

```
auxIOPortResolution OBJECT-TYPE
SYNTAX      INTEGER (1..32)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>Defines number of bits used for the IO-port (e.g.
      width of digital, resolution of analog). Thus, this feature allows
      the digital monitoring (via NTCIP) of an analog port on the agent.

      <DescriptiveName>AuxIOPort.resolution:quantity

      <DataConceptType>Data Element

      <Unit>bit"
::= {auxIOEntry 4}
```

2.8.3.5 Auxiliary Value Parameter

```
auxIOPortValue OBJECT-TYPE
SYNTAX      INTEGER (0..4294967295)
ACCESS      read-write
STATUS      mandatory
DESCRIPTION
      "<Definition>For input or bidirectional ports, this contains the
      current value of the input.  For output ports, this is the last
      commanded value of the port.  A genError shall be generated, if
      this object is set and the port is an input. The actual value
      exchanged shall not exceed [2^(auxIOPortResolution) – 1]; any SET
      operation to a value in excess of this number shall result in a
      genErr and any GET response in excess of this value shall be
      considered erroneous.

      <DescriptiveName>AuxIOPort.value:number

      <DataConceptType>Data Element"
::= {auxIOEntry 5}
```

2.8.3.6 Auxiliary Port Direction Parameter

```
auxIOPortDirection OBJECT-TYPE
SYNTAX      INTEGER {
                   output (1),
                   input (2),
                   bidirectional (3)}
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>Indicates whether state of this port can be set (output),
      read (input) or both (bidirectional).

      <DescriptiveName>AuxIOPort.direction:code

      <DataConceptType>Data Element"
::= {auxIOEntry 6}
```

2.8.3.7 Auxiliary Port Last Commanded State Parameter

```
auxIOPortLastCommandedState OBJECT-TYPE
SYNTAX      INTEGER (0..4294967295)
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
      "<Definition>For bi-directional ports, this object shall indicate
      the last state to which the auxIOPortValue object was set.  For
      output ports, this value shall always be equal to the
      auxIOPortValue object.  For input ports, this value shall always
      be zero (0).

      <DescriptiveName>AuxIOPort.lastCommandedState:number

      <DataConceptType>Data Element"
::= {auxIOEntry 7}

END
```

Section 3
Conformance

NOTE—The conformance requirements previously included in 1201 v01 have been removed from this standard. This document only defines the data that may be useful for a given device; any requirements for supporting a specific piece of data is defined in device-specific standards, such as NTCIP 1202.

Annex A
Concept of Operations
(Normative)

This Annex provides examples of how a management station may interface with a device complying with this standard as envisioned by the authors. Any device claiming conformance with the subject features depicted in these figures shall support the exchanges as shown. However, the flexible design of the NTCIP protocols allows a large number of other possibilities and these figures do not limit any other requirements of these standards. These diagrams are merely provided to promote a common understanding of how systems may be designed in order to increase the likelihood of interchangeability in deployed systems.

Four use cases are presented, as shown in Figure A-1.

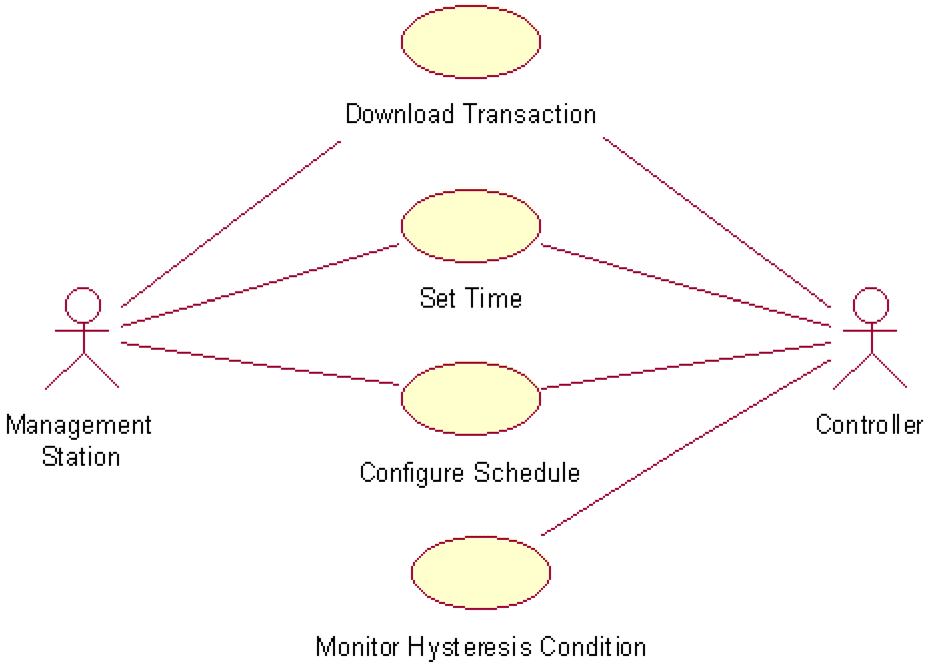

Figure A-1
Global Use Cases

A.1 DOWNLOAD TRANSACTION USE CASE

The first use case is for a Transaction. The intent of this use case is that a management station has a need to download several inter-related parameters to the controller. Because the parameters are inter-related, they must be set simultaneously in order for the set operation to be validated by the controller (e.g., the download may consist of a set of parameters, whose sum must equal the sum of another set of parameters; and the management station wishes to change the sum for both sets).

The parameters that require the use of the transaction mode are device-specific. Some devices may not require support of the transaction feature, while other devices may require SET operations on any database object to be within the transaction mode.

When used, the feature allows a device to buffer a series of set operations on database parameters and to implement all operations simultaneously in order to properly perform controller consistency checks.

The normal, fault-free process is shown in Figure A-2.

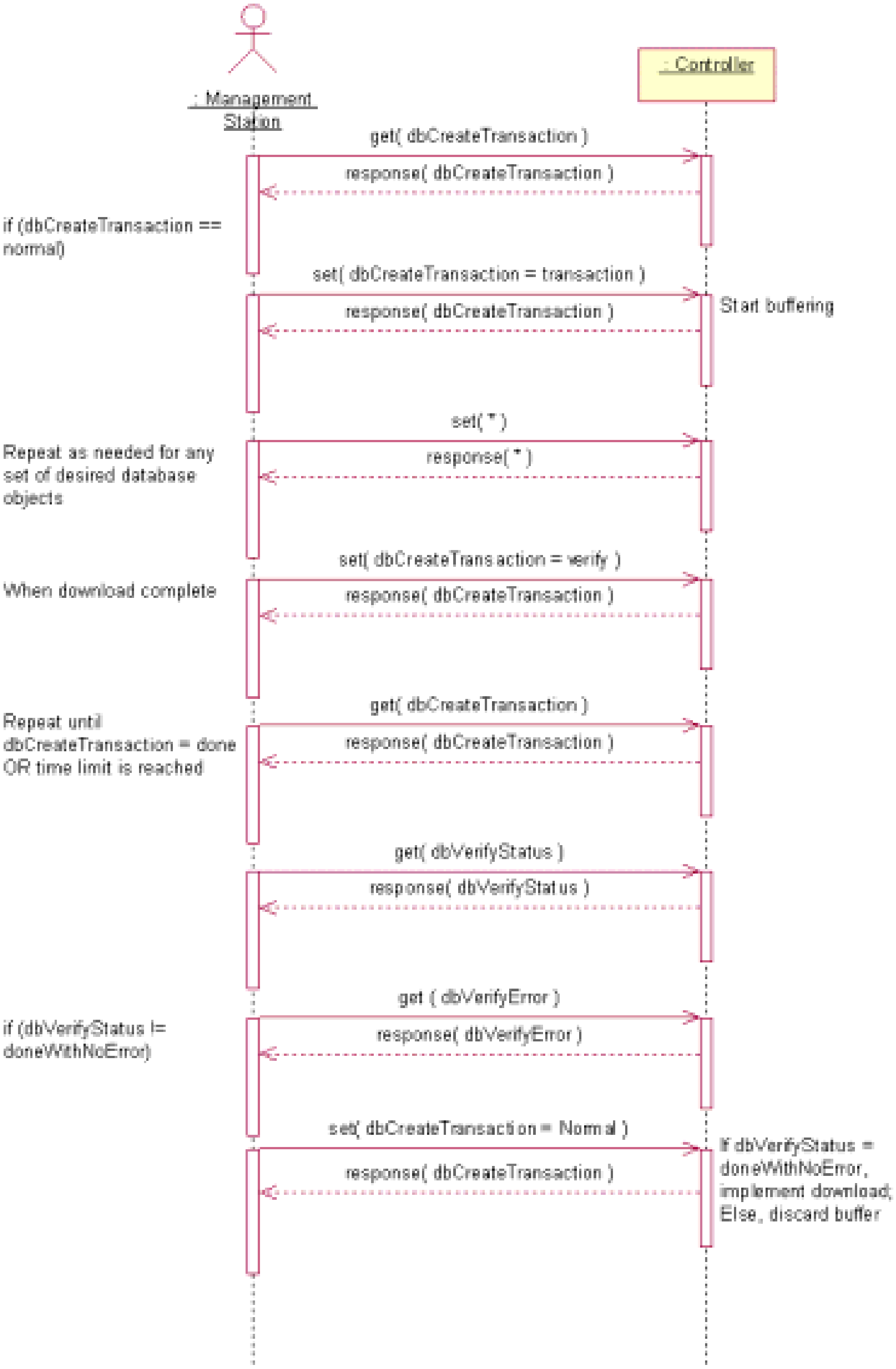
: Management Station
: Controller
get(dbCreateTransaction)
response(dbCreateTransaction)
if (dbCreateTransaction == normal)
set(dbCreateTransaction = transaction)
response(dbCreateTransaction)
Start buffering
set(*)
response(*)
Repeat as needed for any set of desired database objects
set(dbCreateTransaction = verify)
response(dbCreateTransaction)
When download complete
get(dbCreateTransaction)
response(dbCreateTransaction)
Repeat until dbCreateTransaction = done OR time limit is reached
get(dbVerifyStatus)
response(dbVerifyStatus)
get (dbVerifyError)
response(dbVerifyError)
if (dbVerifyStatus != doneWithNoError)
set(dbCreateTransaction = Normal)
response(dbCreateTransaction)
If dbVerifyStatus = doneWithNoError, implement download; Else, discard buffer

Figure A-2
Fault Free Process Dialog

Within this mode, the controller operates as a state machine as described in the definition of dbCreateTransaction. Figure A-3 supplements this definition and provides a formal UML representation of the state machine.

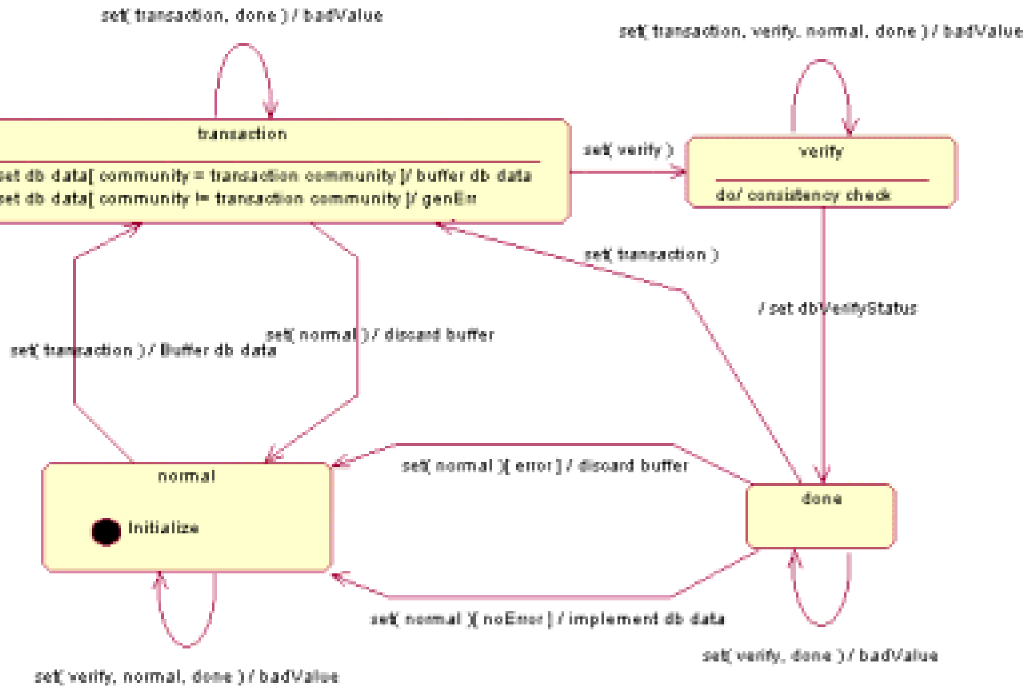
set(transaction, done) / badValue
set(transaction, verify, normal, done) / badValue
transaction
event:set db data[community = transaction community]/ buffer db data
event:set db data[community != transaction community]/ genErr
set(verify)
verify
do/ consistency check
set(transaction)
/ set dbVerifyStatus
set(normal) / discard buffer
set(transaction) / Buffer db data
normal
Initialize
set(normal)[error] / discard buffer
done
set(normal)[noError] / implement db data
set(verify, done) / badValue
set(verify, normal, done) / badValue

Figure A-3
Controller State Machine

A.2 SET TIME

The second use case is to set the time in a controller. There are three key parameters that affect the local time stored in the controller:

- globalTime (which is time in UTC)
- globalDaylightSavings (which is a flag to indicate if daylight savings is active)
- controllerStandardTimeZone (which is the offset between local Standard Time and UTC)

All three of these parameters are independent from one another and thus a controller shall allow a management station to set any or all of these parameters in any order using one or more set operations and may additionally combine these parameters in any fashion with other parameters.

When setting any one of these values, the indicated object shall be set to the indicated value and the value of controllerLocalTime shall be updated to reflect this new value; but none of the other time objects shall be affected.

A.2.1 Example 1 – Changing Global Time

Original Values:
globalTime: 1023278400 (12:00 noon 5 June 2002)
globalDaylightSavings: disable
controllerStandardTimeZone: -21600

controllerLocalTime: 1023256800 (6:00 AM 5 June 2002)

Set Operation
globalTime: 1023282000 (1:00 PM 5 June 2002)

Updated Values:
globalTime: 1023282000 (1:00 PM 5 June 2002)
globalDaylightSavings: disable
controllerStandardTimeZone: -21600
controllerLocalTime: 1023260400 (7:00 AM 5 June 2002)

A.2.2 Example 2 – Changing Daylight Savings

Original Values:
globalTime: 1023278400 (12:00 noon 5 June 2002)
globalDaylightSavings: disable
controllerStandardTimeZone: -21600
controllerLocalTime: 1023256800 (6:00 AM 5 June 2002)

Set Operation
globalDaylightSavings: enableUSDST

Updated Values:
globalTime: 1023278400 (12:00 noon 5 June 2002)
globalDaylightSavings: enableUSDST
controllerStandardTimeZone: -21600
controllerLocalTime: 1023260400 (7:00 AM 5 June 2002)

A.2.3 Example 3 – Changing Time Zone

Original Values:
globalTime: 1023278400 (12:00 noon 5 June 2002)
globalDaylightSavings: disable
controllerStandardTimeZone: -21600
controllerLocalTime: 1023256800 (6:00 AM 5 June 2002)

Set Operation
ControllerStandardTimeZone: -18000

Updated Values:
globalTime: 1023278400 (12:00 noon 5 June 2002)
globalDaylightSavings: disable
controllerStandardTimeZone: -18000
controllerLocalTime: 1023260400 (7:00 AM 5 June 2002)

A.2.4 Example 4 – Changing All Three Parameters

Original Values:
globalTime: 1023278400 (12:00 noon 5 June 2002)
globalDaylightSavings: disable
controllerStandardTimeZone: -21600
controllerLocalTime: 1023256800 (6:00 AM 5 June 2002)

Set Operation
globalTime: 1023282000 (1:00 PM 5 June 2002)
globalDaylightSavings: enableUSDST
controllerStandardTimeZone: -18000

Updated Values:

globalTime: 1023282000 (1:00 PM 5 June 2002)
globalDaylightSavings: enableUSDST
controllerStandardTimeZone: -18000
controllerLocalTime: 1023267600 (9:00 AM 5 June 2002)

A.3 CONFIGURE SCHEDULER

This use case depicts an approach to configuring the time base schedule. Figure A-4 indicates that the device-specific action table should be configured first, followed by the day plan parameters, followed by the time base schedule entries. This approach minimizes the likelihood of an invalid reference occurring during download (i.e., a schedule entry referencing an invalid day plan). However, other approaches may be supported.

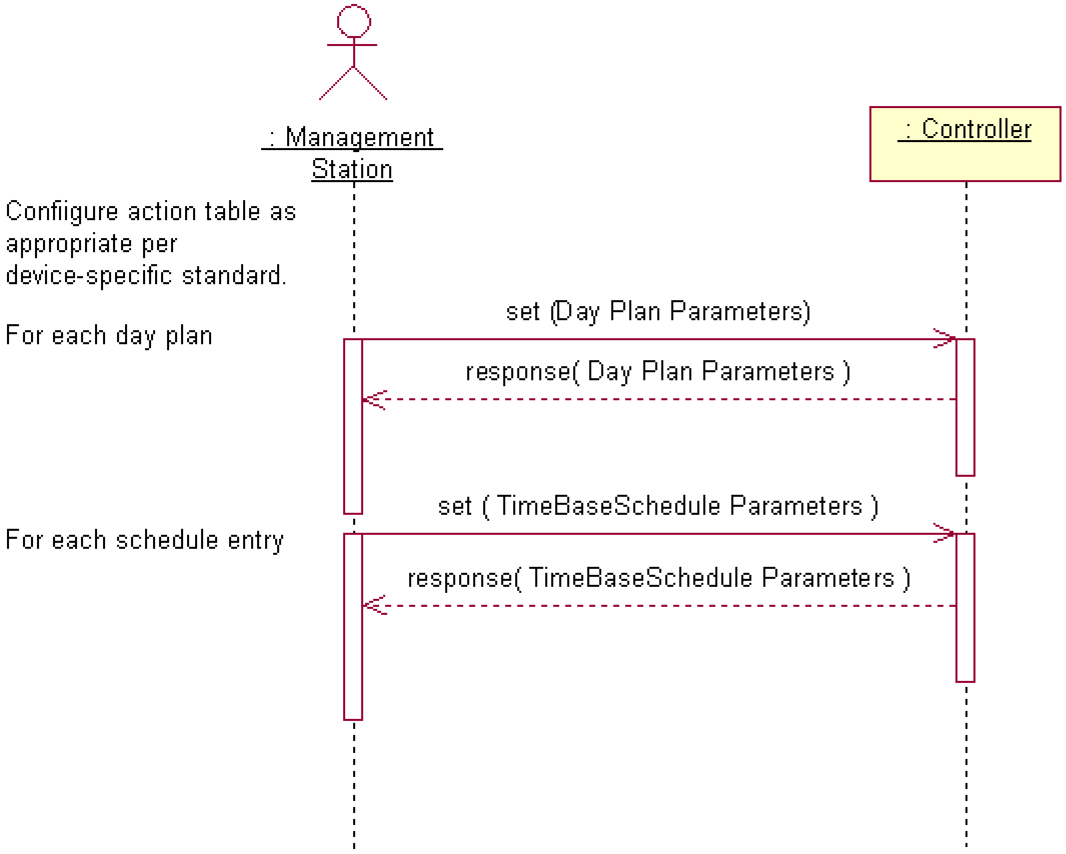
: Management Station
: Controller
Configure action table as appropriate per device-specific standard.
For each day plan
set (Day Plan Parameters)
response(Day Plan Parameters)
set (TimeBaseSchedule Parameters)
For each schedule entry
response(TimeBaseSchedule Parameters)

Figure A-4
Scheduler Dialog

A.4 MONITOR HYSTERESIS CONDITION

This clause depicts the operation of a controller monitoring a hysteresis condition. This use case would be active whenever a controller is configured to log an event that is configured for the hysteresisBound mode.

The times at which the controller would log an event are depicted in Figure A-5. The straight lines depict the lower and upper bounds that are defined in the configuration table for the associated event. The curved line represents the value of the referenced object as it changes over time. The heavy dots indicate those times at which the controller registers a log in the event log table.

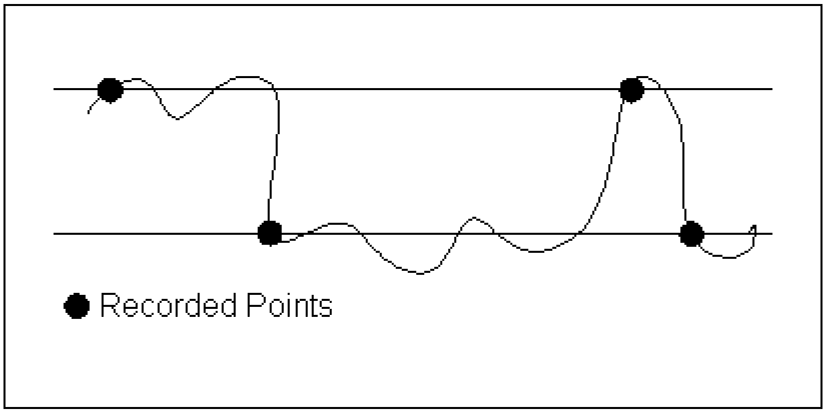

Figure A-5
Example of Controller Event Logging

Annex B
Documentation of Revisions
(Informative)

This annex identifies the changes that have been made to the NTCIP 1201 standard that have required the deprecation of objects. The NTCIP effort makes reasonable efforts to ensure that the standards are as backwards compatible as possible, but the primary purpose of the standard is to provide interoperability by developing standards in a consensus environment. When changes are required to meet these objectives, the problematic objects are deprecated and, in most cases, are replaced with new objects. This annex identifies why each of these changes have been made. New implementations should support the new/replacement objects; they may also support deprecated objects.

B.1 TRANSACTION MODE

The transaction mode process was modified by NTCIP 1201:1996 Amendment 1, which was approved in 2001. Implementations discovered that the original process did not provide for the desired operation in the presence of multiple management stations (e.g., a central and a local laptop). Specifically, there were problems with the second management station killing the first operation in order to issue a control command. The solution deprecated dbErrorID, dbTransactionID, and dbMakeID; revised the definition of dbCreateTransaction, and created two new objects labeled dbVerifyStatus and dbVerifyError.

B.2 LOCAL TIME

The process to set and retrieve the local time was modified by this version of NTCIP 1201. Implementations discovered that there was an ambiguity to the meaning of the object when setting the globalLocalTimeDifferential object during the one-hour period of the fall daylight savings time transition. As a result, many implementations imposed restrictions on how a management station could set time within the controller. For example, several implementations defined precise dialogs that had to be followed to set the time. Unfortunately, the different restrictions imposed by different implementations resulted in interoperability problems of this feature. The solution deprecated the globalLocalTimeDifferential object and added two new objects, labeled controllerStandardTimeZone and controllerLocalTime.

B.3 HDLC GROUP ADDRESS

The definition of the HDLC Group Address Table was modified by this version of NTCIP 1201. Experience demonstrated that the definition of hdlcGroupAddress had been interpreted differently by different implementers. Several implementers had interpreted the definition to suggest that the stored value would be reformatted for PMPP, while others interpreted the definition to require the stored value to be in PMPP format. These differences resulted in interoperability problems. The solution deprecated the hdlcGroupAddress object and created the hdlcGroupAddressNumber object.

Annex C
Class Diagrams
(Informative)

This annex provides an overview of the data defined by this standard through the use of UML Class Diagrams. The information presented in this annex is formally defined elsewhere in this standard; however, these figures concisely depict key characteristics of these definitions in a concise manner and are provided as a useful reference tool for system designers.

The diagrams conform to the modeling conventions defined by ISO 14817 and were used to develop the ISO 14817 conforming Descriptive Names as shown within each object definition in Section 2 of this standard. The ObjectClassTerm of the descriptive name is indicated by the name of each box within the figures and the propertyTerm is shown as being an item within the box. These Descriptive Names are also used by the on-line ITS Data Registry as the primary name of each data concept.

NOTE—While the discussion within this section indicates that virtually every feature is optional, in order to claim conformance with various NTCIP standards, support for many of these features may be mandatory.

C.1 CONFIGURATION INFORMATION

Figure C-1 depicts the configuration data stored by a controller.

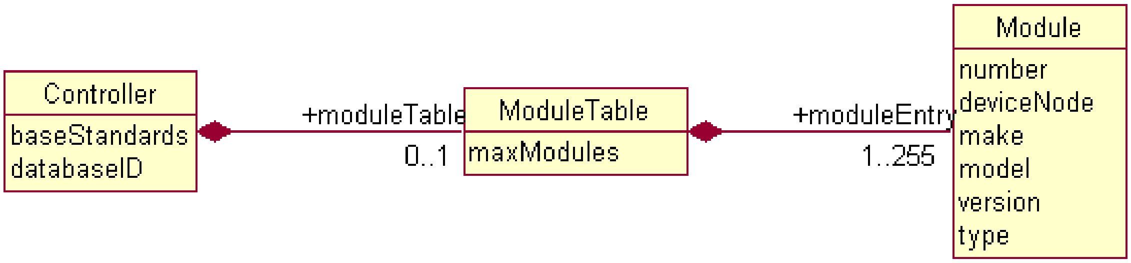

Figure C-1
Class Diagram of the Configuration Data

The figure indicates a controller may have a database identifier and zero or one module tables. If there is a module table, then the controller may additionally support an object defining the maximum number of modules supported within the table, which may be between one and 255, as indicated by the link to the Module class. For each module, the controller may support a variety of information, including:

- The module number
- The device node to which the module relates
- The make of the module
- The model of the module
- The version of the module, and
- The type of module

C.2 TRANSACTION INFORMATION

Figure C-2 depicts the transaction state data stored by a controller.

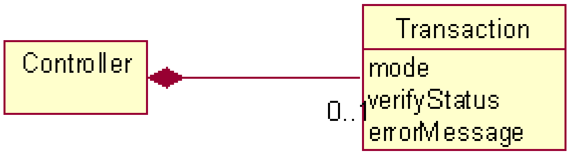

Figure C-2
Class Diagram of the Transaction Service

The figure indicates a controller may have support a transaction feature. The feature is characterized by the following information:

- A mode
- A status, and
- An error code

C.3 TIME INFORMATION

Figure C-3 depicts the time related data stored by a controller.

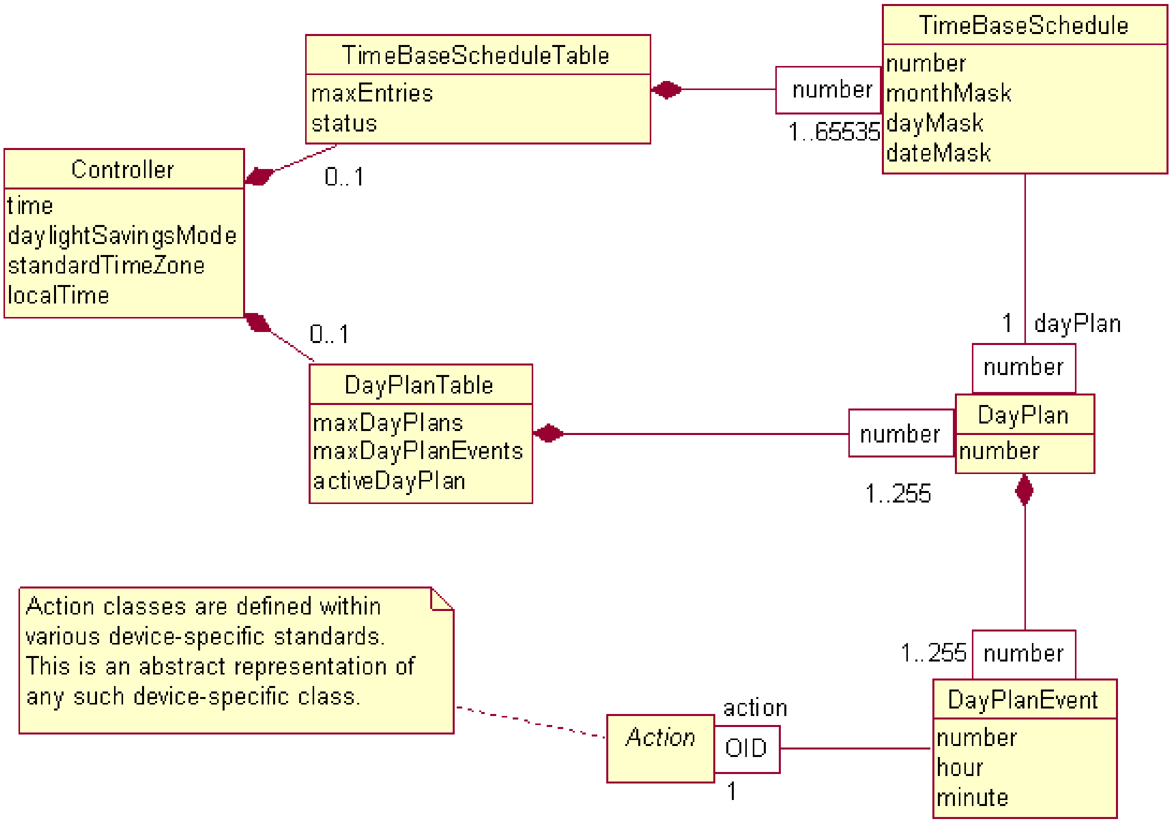

Figure C-3
Class Diagram of Time Information

The figure indicates a controller may store time information, including:

- The current time in UTC
- An indication of the daylight savings mode
- An indication of the time zone when in standard time
- An indication of the local time

The controller may also support a timebase schedule table. If this is supported, it is characterized by the maximum number of entries that it may contain, which must be at least one and may be no greater than 65535, and a status. For each entry, the following information may be stored:

- A schedule number
- A month mask indicating which months the schedule may be active
- A day mask indicating which days of the week the schedule may be valid
- A date mask indicating which dates of the month the schedule may be active
- A link to a day plan record

In order to have a link to a day plan, the day plan must also be supported; which in turn requires that its container class, the day plan table must also be supported. The day plan table is characterized by:

- The maximum number of day plans that may be stored, which must be between one and 255,
- The maximum number of events that may occur during a day, which must be between one and 255
- An indication of the day plan that is currently active

The day plan itself only consists of the day plan number and a link to between one and 255 day plan events. Each day plan event is described by:

- A number,
- The hour during which the event occurs
- The minute during which the event occurs
- The status of the action, and
- A link to the specific action to be performed

The specific action to be performed is defined elsewhere due to the device specific nature of actions.

C.4 REPORT INFORMATION

Figure C-4 depicts the report data stored by a controller.

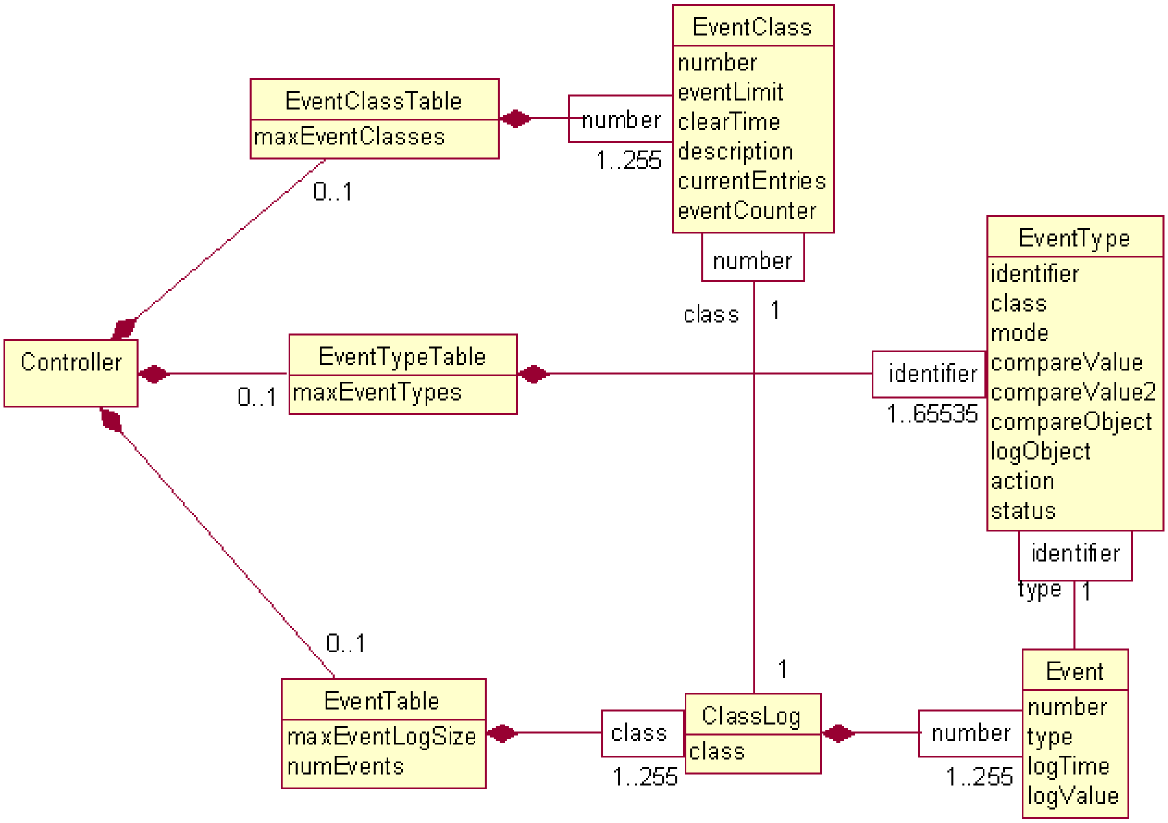

Figure C-4
Class Diagram of the Event Log Service

The figure indicates a controller may support an event class table. If this is supported, it is characterized by the maximum number of entries that it may contain, which must be at least one and may be no greater than 255. For each entry, the following information may be stored:

- An event class number
- The maximum number of events that may be stored for that class
- A time field that clears all entries older than that time
- A description field that allows a user to provide a textual explanation of the class
- A count of the current number of events for the class

For each EventClass, there is exactly one ClassLog, which must be contained by the EventTable. The EventTable is characterized by the maximum number of events it may store and it may contain one to 255 ClassLogs. Each ClassLog is merely a container for one to 255 events and each event is defined by:

- A number
- A time at which the event occurred
- A value that is recorded along with the log entry, and
- An identifier of the type of event

The type of event is a user-definable concept defined by the EventType Class. Each EventType is characterized by:

- An identifier, used by the Event Class to reference this specific EventType
- A class indicating in which ClassLog the event will be recorded
- A mode indicating what type of event will cause a log entry

- Two compare values and an object that may be used as a part of the logic to detect an event
- A reference to an object that is logged in the value field of the Event when an event is logged, and
- An action field that indicates whether the EventType is enabled or disabled.

The standard allows a device to support anywhere from one to 65535 EventTypes. Each EventType is stored within the EventTypeTable. The EventTypeTable also is characterized by the maximum number of EventTypes that the given implementation supports.

C.5 AUXILIARY INPUT/OUTPUT INFORMATION

Figure C-5 depicts the auxiliary input/output data stored by a controller.

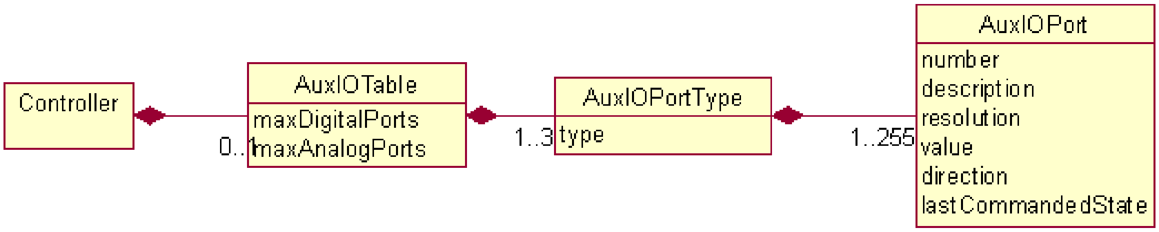

Figure C-5
Class Diagram for Auxiliary Input/Output Services

The figure indicates a controller may support an auxiliary input/output table. If this is supported, it is characterized by the maximum number of digital and analog ports supported by the device. Each port type is allocated to its own sub-table in the AuxIOPortType table, which contains multiple entries, one for each port, where each port is characterized by:

- A number
- A description
- A resolution of the data supported by the port, and
- A value

Annex D
Summary of Changes
(Informative)

To the extent reasonable, the NTCIP community attempts to minimize the number of changes to a document in order to minimize interoperability problems among different versions of the same standard. However, on occasion, problems are identified with existing standards that necessitate a change. The NTCIP effort rectifies such problems while attempting to minimize the impact on existing implementations. This annex explains the problem identified resulting in each change, a description of the change made, and an analysis of the impact of each change on implementations.

The changes in this version of the standard reflect lessons learned from the deployment of version 01 of the standard, incorporate better documentation (in the Annex) of some of the logic required to implement the standards, and to add new features requested by the ITS community. Specific changes made to this standard between the first version (1996) and this version are documented in the following clauses.

D.1 UPDATED ISO TREE

Due to the various other changes in the standard, the ISO Tree contained in Clause 1.5 was updated to properly reflect the contents of the standard. The update to this figure should not cause any interoperability problems.

D.2 UPDATES TO CONFORM WITH NTCIP 8004

The data stored in field devices are often retrieved by a central system and then may be exchanged with other centers as a part of regional communications. These center-to-center communications use protocols other than SNMP and require the data to be defined according to either IEEE 1489 (or its recently approved update known as ISO 14817). The fact that the original version of NTCIP did not define data in this format created ambiguities for center-to-center implementations.

In order to ensure that there would be a single definition for all NTCIP data, regardless of what context it was used in (e.g., center-to-center vs. center-to-field), the NTCIP community defined an enhanced MIB format, as defined in NTCIP 8004, to be used for all new and updated NTCIP standards.

The additions that this update creates (e.g., the <DEFINITION> tags, etc.) should not cause any interoperability problems.

D.3 UPDATED NAME OF THE MIB

Changes to a MIB can affect the way other MIBs import data. Thus, when a MIB imports data from another MIB, it should be able to unambiguously reference the specific version of the MIB that it wants to import. Therefore, every update to an NTCIP standard results in an update to the name of the MIB according to the rules in NTCIP 8004.

The update to the MIB name should not cause any interoperability problems, and in fact prevents ambiguity as to which version of this MIB may be referenced from another MIB.

D.4 ADDED DEFAULT VALUE STATEMENTS

Interoperability problems can arise when different controllers initialize differently. As a result, this standard has standardized the default initialization value of several configuration and control parameters.

This is a change to the standard and may result in some version 01 devices performing slightly differently than version 02 devices. However, this *reduces* interoperability problems overall. Current implementations operate differently from one another and any central system must be customized to handle this uniqueness for each manufacturer. By defining the default value within the standard, this customization can be avoided in the future.

D.6 ENHANCED MODULE VERSION DEFINITION

The module table is intended to provide basic information about the make, model, and version of the controller. However, the original version of the standard provided a generic format for the version that did not adequately allow for proper configuration management of software. The new standard defines a detailed format for the presentation of the version information.

While this is a change to the standard to which some version 01 devices may not conform, it does not present any real interoperability problems between version 01 and version 02 devices.

D.7 ADDED AN OBJECT TO IDENTIFY SUPPORTED STANDARDS

Several integrators have expressed concerns over the ability to be able to quickly determine to which standards and which versions of standards a device claims conformance. By being able to query the device to determine which standards it supports, a central system will be able to quickly determine how to manage the device. Therefore an object providing this information in a standard format has been added.

This addition should not create any interoperability problems. A central system will be able to readily identify any version 01 device since it will return a noSuchName error.

D.8 CORRECTED THE DATABASE TRANSACTION FEATURE

Experience with the version 01 Database Transaction feature revealed many ambiguities and problems resulting in version 01 implementations from different manufacturers that were not interoperable. The lessons learned from these implementations were discussed and the working group revised the design to address the problems identified. This included clarifying the definition of the dbCreateTransaction object and replacing several objects of the transaction feature.

This change was made to resolve existing interoperability problems. While version 01 implementations will have to be changed in order to conform to the new standard, this is an improvement in the sense that the version 01 feature did not work as intended.

D.9 ADDED SUPPORT FOR ADDITIONAL DAYLIGHT SAVING MODES

Several parties located outside of the U.S. are now deploying NTCIP for various devices and have pointed out that the NTCIP should support all of the various daylight savings plans. Thus, these have been added to the daylight savings object.

This addition is fully backwards compatible and should not cause any interoperability problems. It will have no effect on systems in the U.S.; version 01 systems outside of the US have not had a way to offer support of other daylight saving modes in a standard way, but with the version 02 enhancement, they will now be able to offer this feature.

D.10 ADDED A SCHEDULE STATUS OBJECT

Some agencies have wanted to be able to monitor the logic of the timebase schedule a little closer and as a result, we have added a status object to the timebase schedule table. This is an extra feature that is fully backwards compatible.

D.11 CLARIFIED DEFINITIONS OF DAY PLAN OBJECTS

Various questions had been raised about the precise meaning of the object definitions for the day plan table. Version 02 clarifies these definitions in response to these questions. However, the clarifications reflect actual implementations and should not result in any interoperability problems; rather they are likely to prevent interoperability problems in the future.

D.12 CORRECTED PROBLEMS WITH THE LOCAL TIME LOGIC

A problem was discovered with the time differential logic in that if the globalTime was set during the one-hour fall-back period of the daylight savings logic; there was an ambiguity as to what time was intended. Manufacturers overcame this ambiguity in their own implementations in a variety of ways, many of which created interoperability problems with other manufacturers. Several options were considered to correct this flawed logic, but they all resulted in some level of interoperability problems. Thus, the working group concluded that the best solution was to produce the cleanest design which required deprecating the global time differential object and adding new objects for local time and time zone.

Version 02 corrects an existing interoperability problem. This does result in a minor compatibility problem between the variety of version 01 interpretations and the version 02 design, but the working group was unable to find an alternative solution that adequately corrected the problem without presenting new problems. By deprecating objects and creating new objects, any central system will quickly discover (by receiving a noSuchName error) if it tries to access the feature using the wrong version.

D.13 CLARIFIED DEFINITIONS RELATED TO THE EVENT LOG

The WG received a variety of detailed comments about the exact definitions used for objects in the event log. As a result of these comments, the working group made several clarifications, but in all cases, these merely clarified the text and explained how manufacturers had implemented the features. It is not expected that any of these clarifications will result in interoperability problems.

D.14 REORDERED CLAUSES FOR THE EVENT LOG

The order of the subclauses related to the event log proved confusing to some readers and the WG therefore decided to reorder the subclauses. However the OBJECT IDENTIFIERs for the objects have not changed and this is merely an editorial change; therefore there should be no interoperability problems created by this reordering.

D.15 ADDED SUPPORT FOR ANOTHER MODE TO THE EVENT LOG

Based on requests from implementers, the WG added a new mode for the event log configuration table (andedWithValue) and provided better explanations of the definitions of each mode.

This addition is fully backwards compatible and the explanations will hopefully prevent future interoperability problems.

D.16 ADDED ERROR VALUE TO THE EVENT CONFIGURATION STATUS

Based on requests from implementers, the WG added an error code to the status object of the event configuration table in order to ensure that the controller is not programmed to repeatedly check an invalid condition. The WG also added logic to the object that requires a consistency check whenever the configuration of the row changes.

This change presents only a minor compatibility challenge between versions, but prevents interoperability problems where some manufactures had used the 'other' code to mean error. The WG determined that this was the least problematic solution.

D.17 CORRECTED SYNTAX OF EVENT LOG SIZE OBJECT

The original standard indicated that the lower bound of the event log size was zero; however, if the size was zero, there would be no table and this object should not be supported. Thus, in order to avoid this contradiction, the lower limit was redefined to be one.

This will not create any interoperability problems between versions since any implementation supporting this feature will have a value greater than one.

D.18 REPLACED THE GROUP ADDRESS OBJECT

Version 01 of the standard had an object defining the PMPP group address to which the device belonged; however the meaning of the value had been interpreted in two different ways. One group held that the intent was that the value was supposed to be the group address number that was encoded in the PMPP address field. The other group contended that the value was the encoded PMPP address field. Due to this conflict and resulting non-interoperability in deployed systems, the existing object was deprecated and a new object defined to resolve the issue.

While the solution of replacing the existing object presents a minor interoperability issue, the solution does provide an unambiguous definition of the object and any central system will be able to readily identify version 01 implementations since they will not support the version 02 object.

D.19 ADDED GENERIC AUXILIARY I/O OBJECTS

The development of the DMS standard identified the need to support auxiliary I/O ports. This need was later realized by several other groups, including the ESS WG. As a result, the auxiliary objects defined in NTCIP 1203 were refined and enhanced and added to the global object standard.

This is a new feature for version 02 and will not create any interoperability problems with version 01 deployments.

D.20 REMOVED CONFORMANCE STATEMENTS

Deployments using the version 01 NTCIP standards highlighted problems with writing procurement specifications using conformance groups. Problems also arose as working groups began updating their standards as the user needs and requirements for each feature were typically not defined in a clear fashion within the subject version 01 standards.

As a result, the NTCIP community has changed the format of NTCIP standards to follow an outline that is based on a systems engineering process (SEP). The result of this change is that the conformance groups have been eliminated from the standards and replaced with a Protocol Requirements List (PRL). In the case of NTCIP 1201, the PRL is located in the subject device standard that references NTCIP 1201. This table, combined with the referenced Requirements Traceability Matrix (RTM) now defines the conformance requirements for the standard rather than the conformance groups and conformance statement used in version 01 standards.

This change should not present any interoperability problems with version 1 implementations.

D.21 ADDED A CONCEPT OF OPERATIONS

One of the problems that many implementers had in deploying the first version of the standard was in understanding the intended operations of some features. In order to address this issue, the WG included Annex A to explain how various features were intended to operate.

Although this is normative text, these are intended to clarify the text that already existed in the version 01 standard and are therefore not expected to produce any interoperability problems.

D.22 PREPARED COMMUNICATION OBJECTS TO BE MOVED TO 1103

The development of NTCIP 1103 resulted in the realization that the security objects and the event log objects should be moved to the new standard because they relate more to application layer issues than to the end-application. While the WG concluded that they should be moved to NTCIP 1103, they have been left in this standard until the NTCIP 1103 standard is approved and published.

D.23 ADDED ANNEX B TO DOCUMENT DEPRECATED OBJECTS

All objects that have been deprecated from the standard are documented in Annex B so that future developers can understand objects that may exist in or are used by legacy equipment.

The inclusion of this information will not result in any interoperability problems and may assist in newer systems being able to communicate with version 01 devices.

D.24 ADDED CLASS DIAGRAMS

Many users of version 01 documentation found it difficult to understand the context of the various objects defined in the Management Information Base (MIB). While the various object definitions provided the detailed definition of each object, it was difficult for them to readily obtain a high-level view of how the data worked together.

Annex C (Informative) was added to the standard in order to provide the reader with high-level graphical images that depict the various relationships among all of the data defined by the standard, including the rules on multiplicity (i.e., how many of one object might exist for a given instance of another object). While all of this information was (and still is) recorded within the textual definition of the objects, providing high-level graphical depictions of these relationships facilitate this understanding.

This addition is marked informative and is only intended for clarification. It is not believed to have any impact on existing systems.

§