NTCIP 1101:1996

National Transportation Communications for ITS Protocol-Simple Transportation Management Framework

Joint Standard of AASHTO, ITE, and NEMA

formerly numbered **NEMA TS 3.2-1996**







NTCIP Standards Publication

A Joint Publication of
American Association of State Highway and Transportation Officials (AASHTO)
Institute of Transportation Engineers (ITE)
National Electrical Manufacturers Association (NEMA)

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS

Not for Resale

NEMA Standards Publication TS 3.2-1996

National Transportation Communications for ITS Protocol (NTCIP)
Simple Transportation Management Framework

Published by

National Electrical Manufacturers Association
1300 N. 17th Street
Rosslyn, Virginia 22209

© 1996 by National Electrical Manufacturers Association

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS

Not for Resale

CONTENTS

Page

Foreword ii

Section 1 GENERAL
1.1 Scope 1
1.2 References 1
1.2.1 Normative References 1
1.2.2 Other References 2
1.3 Terms 2
1.4 Abbreviations and Acronyms 2
Section 2 SIMPLE TRANSPORTATION MANAGEMENT FRAMEWORK
2.1 Integrated Management 6
2.2 Registration Authority Hierarchy 7
Section 3 NEMA STRUCTURE AND IDENTIFICATION OF MANAGEMENT INFO
3.1 Introduction 9
3.2 The NEMA Authority 9
3.3 Structure and Identification of Management Information 10
3.3.1 Names 10
3.3.2 Syntax 12
3.3.3 Managed Objects 12
3.3.4 Extensions to the MIB 13
3.3.5 Groups 14
Section 4 TRANSPORTATION MANAGEMENT INFORMATION BASE
4.1 Transportation Type Definitions 15
4.1.1 Byte and UByte Types 15
4.1.2 Short and UShort Types 15
4.1.3 Long and ULong Types 15
4.1.4 EntryStatus Type 15
4.1.5 OwnerString Type 16
4.2 Structure of Transportation Management Information 16
4.2.1 Protocols subtree 16
4.2.2 Devices subtree 19
Section 5 PROTOCOL DEFINITIONS
5.1 Simple Transportation Management Protocol 21
5.1.1 Protocol Data Unit 21
5.1.2 Operations and Semantics 25
5.2 Simple Network Management Protocol 26
5.3 Transport Mappings 26
Section 6 CONFORMANCE LEVEL
6.1 General Conformance 27
6.1.1 Conformance Level 1 27
6.1.2 Conformance Level 2 27
6.2 Management Application Conformance 27
Annex A SMI FOR NEMA 29
Annex B TRANSPORTATION MANAGEMENT INFORMATION BASE 31

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Foreword

This publication describes the Simple Transportation Management Framework used for managing and communicating information between management stations and transportation devices. It presents the rules by which information is described and encoded for transmission. It also describes the Simple Transportation Management Protocol that is used to transfer information across communication links.

The text including the annexes consists of mandatory requirements which are classified as NEMA Standard Publication and information that is in conformance with NEMA Authorized Engineering Information. Both are defined in the back of each NEMA Standard Publication.

User inputs are solicited by canvass on each NEMA standards proposal for Transportation Management and Associated Control Equipment. This procedure provides guidance before industry action. Every comment received is acknowledged and the action taken reported to the submitter. Proposed or recommended revisions should be submitted to:

Vice President Engineering
National Electrical Manufacturers Association
1300 N. 17th Street
Rosslyn, Virginia 22209

This standards publication was developed by the Transportation Management Systems and Associated Control Devices section of the National Electrical Manufacturers Association. Section approval of the standard does not necessarily imply that all section members voted for its approval or participated in its development. At the time it was approved, the Transportation Management Systems and Associated Control Devices section had the following members.

ADDCO Manufacturing Co.—St. Paul, MN
American Electric Sign—Spokane, WA
Automatic Signal/Eagle Signal Corporation—Austin, TX
BI Tran Systems, Inc.—Sacramento, CA
Cylink Corporation—Sunnyvale, CA
Eberle Design, Inc.—Phoenix, AZ
Econolite Control Products, Inc.—Anaheim, CA
Fiberoptic Display Systems, Inc.—Smithfield, RI
Intersection Development Corp.—Fullerton, CA
ITS Product Group—Ormond Beach, FL
McCain Traffic Supply, Inc.—Vista, CA
P B Farradyne Inc.—Rockville, MD
Peek Traffic - Transyt—Tallahassee, FL
Rockwell Automation—Mayfield Heights, OH
Skyline Products, Inc.—Colorado Springs, CO
3M Traffic Control Systems—St. Paul, MN
Traffic Sensor Corp.—Corona, CA
Vultron, Inc.—Rochester Hills, MI

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Section 1
GENERAL

1.1 SCOPE

The scope covers integrated management of transportation networks, networking devices and transportation specific equipment attached to NTCIP based networks. This encompasses previous scopes addressed by other networking management frameworks within the Internet community, including the network management applications. It extends the scope beyond the Internet network management by addressing transportation equipment, networks and transportation management applications.

This includes the structure and identification of transportation information within transportation equipment, the protocols that act on defined information entities and the management applications used for controlling the transportation equipment. An extension to the Internet-standard Network Management Framework, called Simple Transportation Management Framework (STMF), allows for the operation, control, monitoring and configuration of NTCIP networks and the transportation equipment participating on the network. This includes networks utilizing legacy communication mediums such as 1200 bps FSK. This framework also details differences between Simple Transportation Management Framework (STMF) and the original Internet-Network Management Framework.

1.2 REFERENCES

The following publications are adopted in part, by reference in this publication, and are available from the organizations below.

1.2.1 Normative References

The following normative documents contain provisions, which through reference in this text, constitute provisions of this Standards Publication. By reference herein these publications are adopted, in whole or in part as indicated, in this SP.

International Organization for Standardization (ISO)
Case postale 56
CH-1211 Geneve 20
Switzerland

ISO 8824 *Specification of Abstract Syntax Notation One (ASN.1)*, December 1987

ISO 8825 *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*, December 1987

Internet Activities Board
available via electronic file transfer
use anonymous FTP from
NIC.DDN.MIL or DS.INTERNIC.NET

RFC 1155 *Structure and Identification of Management Information for TCP/IP based internets*, Rose M., and K. McCloghrie, May 1980.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

RFC 1157 *Simple Network Management Protocol*, Case, J., Fedor, M., Schoffstall, M., and J. Davin, May 1990.

RFC 1212 *Concise MIB Definitions*, Rose M., and K. McCloghrie, Editors, March 1991.

1.2.2 Other References

The following publication(s) may be used to answer questions not covered by this standard.

Stallings, William, *SNMP, SNMPv2, and CMIP: The Practical Guide to Network-Management Standards*, Massachusetts, Addison-Wesley Publishing Company, 1993, ISBN 0-201-63331-0

1.3 TERMS

agent: The entity which receives commands and transmits responses to the received commands.

proxy agent: An entity which receives and responds to network management commands on behalf of another entity.

manager: The entity which sends commands to entities and processes their responses.

management information base: A structured collection or database of related managed objects defined using Abstract Syntax Notation One (ASN.1).

object: A representation of a resource that is managed.

OBJECT_TYPE: A macro used to define objects in Management Information Bases (MIB).

OBJECT IDENTIFIER: A unique name (identifier) that is associated with each type of object in a MIB. This is a defined ASN.1 type.

1.4 ABBREVIATIONS AND ACRONYMS

ANSI—American National Standards Institute, a standardization group that develops or adopts standards for the United States.

ASCII—American National Standard Code for Information Interchange.

ASN.1—Abstract Syntax Notation One, a formal language for describing information to be processed by computer, an ISO standard.

BER—Basic Encoding Rules, rules for encoding data for transmission used with ASN.1, an ISO standard.

CCITT—International Consultative Committee on Telegraph and Telephony

IAB—Internet Activities Board, group in charge of authorizing RFC's for the purpose of standardizing Internet operations.

IANA—Internet Activities Numbering Authority.

IETF—Internet Engineering Task Force, a group chartered by the IAB to develop certain RFC's for standardization.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

IP—Internet Protocol, primary network layer protocol used by Ethernet based LAN's.

ISO—International Organization for Standardization, a group chartered with developing and authorizing worldwide standards.

LAN—Local Area Network, a network that is limited in distance, typically operates within a building or floor.

MIB—Management Information Base, a collection of management objects written in a subset of ASN.1 notation.

NVT—Network Virtual Terminal.

OSI—Open Systems Interconnection, a standardized model of communications protocols.

PDU—Protocol Data Unit, a part of transmitted data that contains information used by the protocol at a particular layer in the OSI stack.

PER—Packed Encoding Rules, a variation of BER developed for use on low bandwidth communications links, specified in this SP.

RFC—Request For Comments, the name given to correspondence and standards by the IAB.

SMI—Structure of Management Information, a definition of how to create management objects and a hierarchical (tree like) definition of nodes where management objects will be attached for unique identification.

SNMP—Simple Network Management Protocol, a communications protocol developed by the IETF, used for configuration and monitoring of network devices.

SNMPv2—Simple Network Management Protocol version 2, recent modification of SNMP that is undergoing evaluation by the Internet community.

SP—Standards Publication, a formal compilation of NEMA text (NEMA Standard, Authorized Engineering Information, Suggested Standard for Future Design) on the same subject, which has gone through a consensus process and been approved according to NEMA Standardization Policies and Procedures.

STMF—Simple Transportation Management Framework.

STMP—Simple Transportation Management Protocol, a variation of SNMP developed by NEMA to address low bandwidth communication links and real time device monitoring.

TCP—Transmission Control Protocol, a transport level protocol in common use on the Internet.

TLV—Type, Length, Value encoding.

TMIB—Transportation Management Information Base.

UDP—User Datagram Protocol, a simple transport level protocol in common use on the Internet.

WAN—Wide Area Network, a network that spans large geographical areas.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Section 2
SIMPLE TRANSPORTATION MANAGEMENT FRAMEWORK

The Simple Transportation Management Framework (STMF) specifies a set of rules and protocols for organizing, describing and exchanging transportation management information between transportation management applications and transportation equipment such that they interoperate with each other. It is based on the Internet-standard Network Management Framework. It is intended that a high level of compatibility, including inter-operability, be maintained between STMF and the Internet-standard Network Management Framework.

There are four major components within the STMF:

a. Structure and Identification of Management Information (SMI)

   This clause is based on the RFC 1155, SMI for TCP/IP. It does not define objects, but instead defines groups (branches) within the management information tree.

   Objects within the greater Internet community included are defined in the Management Information Base II (MIBII), RFC 1213.

b. Transportation Management Information Base (TMIB)

   TMIB contains the groups (branches) below the SMI groups. It contains objects defined by the NEMA NTCIP committee. Annex B contains the ASN.1 definition for the TMIB.

   TMIB also includes the NEMA MIB which contains the groups that NEMA controls, of which transportation is one subgroup. Annex A contains the ASN.1 definition for the NEMA MIB.

   MIBs are created in conformance with RFC 1212. RFC 1212 is a good guide for creating MIB modules or utilizing table column and row creates and deletes.

c. Simple Network Management Protocol version 1 (SNMP)

   SNMP is fully specified in RFC 1157, Simple Network Management Protocol (SNMP).

d. Simple Transportation Management Protocol (STMP)

   STMP is unique to this document and is specified in section 5.

Means exist for defining management information that is exchanged between a device being managed and an application managing the device. The unit of management information is termed a *managed object*. A *managed object* is the smallest entity that can be exchanged between a device and a management application. A collection of related managed objects is defined in a document termed a *Management Information Base* (MIB) module. A single module can define the complete Management Information Base; more often multiple modules are involved. A MIB or collection of MIBs, if organized in a hierarchical manner, constitute a tree like structure.

A MIB can be organized in various methods. For the purposes of compliance with the pre-existing Internet structure, and because of its flexibility and extensibility, a tree structure is utilized. This tree structure was created within the ISO-CCITT-OSI community and is utilized for OSI System Management,

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

standards management (ISO, CCITT, ISO-CCITT, ANSI, IANA) and a variety of other information organization tasks. Each object can be uniquely identified by the location of the object within the tree.

Although the scope of a MIB within the Internet is all information within the Internet administrative tree, another common usage reduces the scope to that of the module or information within a subtree. In the STMF, a MIB can refer to either a tree or subtree within any administrative domain.

The SMI describes the common structures and identification scheme for the definition of management information. The SMI is specified in Abstract Syntax Notation One (ASN.1) and can be compiled by MIB compilers, but is not a MIB module. In order to compile MIB modules, SMI definitions (ASN.1 specifications) are included in a MIB module. The distinction is made because SMI defines how to create managed objects. SMI defines how to utilize ASN.1 in order to create and identify management information (objects) within a tree like structure.

After all of the information is identified, created and organized the objects must be communicated. This requires an application layer communications protocol. That is the purpose of the SNMP and STMP components of STMF. The STMP protocol is specifically designed to satisfy some unique technical requirements of the transportation industry. SNMP is used because of its flexibility. These protocols are intended to work together in order to satisfy diverse requirements.

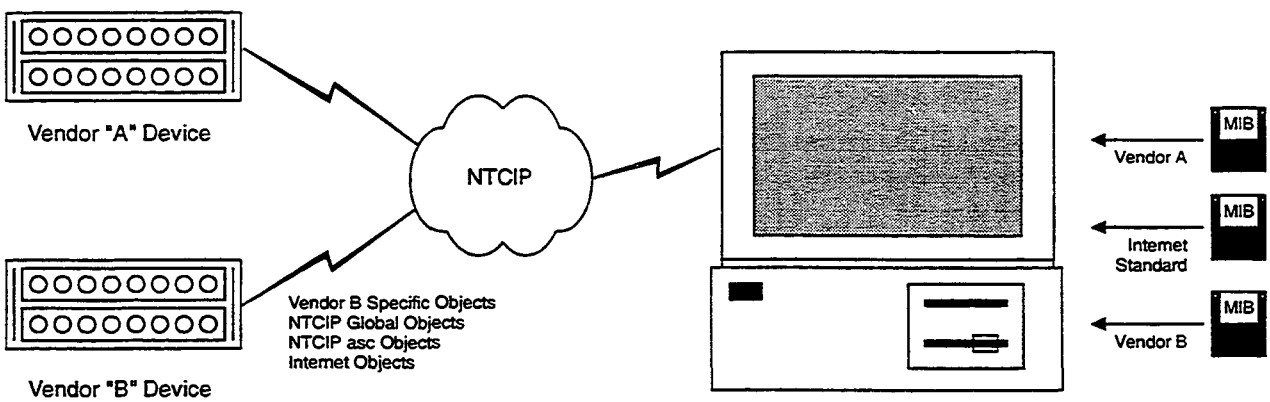


Figure 2-1
MIB INTEGRATION

2.1 INTEGRATED MANAGEMENT

The result of creating and identifying managed objects using ASN.1 is a computer readable description of the information. These modules are ASCII-NVT text, not binary. MIB modules should be available for transportation products conforming to this standard. Users who purchase the equipment with a MIB module can utilize the module to update their management applications.

Management applications running on central control hosts can read this module, and other modules such as controller MIBs and manufacturer specific MIBs, in order to gain information about the capabilities of remote devices. See Figure 2-1. Many SNMP management applications can dynamically load and unload modules (MIBs) describing the information within remote networking devices.

Manufacturers supplying transportation equipment should make available manufacturer specific MIB modules for the equipment. All manufacturers need to obtain a vendor number from either NEMA or IANA

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

prior to creating MIB modules that have manufacturer specific data. Transportation equipment includes a list of supported MIBs.

2.2 REGISTRATION AUTHORITY HIERARCHY

The STMF is entirely compatible with the International Standards that cover the structure and management of information and the Internet-standard Network Management Framework. NEMA is registered in the Internet network management part of the *global name tree*. The official number for all NEMA management information is prefixed with, 1.3.6.1.4.1.1206, which can be represented by the textual naming convention, *iso.org.dod.internet.private.enterprises.nema* as shown in Figure 2-2.

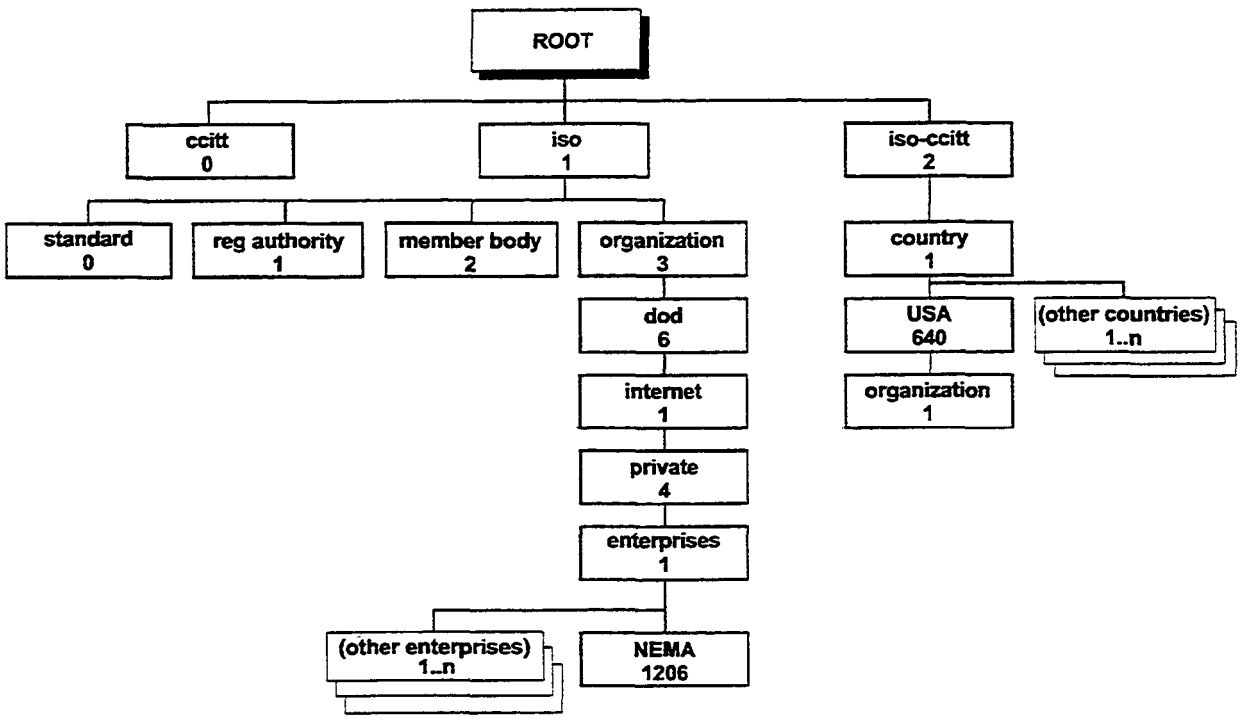


Figure 2-2
THE INTERNET AUTHORITY HIERARCHY

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Section 3
NEMA STRUCTURE AND IDENTIFICATION OF MANAGEMENT INFORMATION (NEMA_SMI)

3.1 INTRODUCTION

This section describes the common structures and identification scheme for the definition of management information used in NEMA standards. Included are descriptions of an object information model for network management along with a set of generic types used to describe management information. Formal descriptions of the structure are given using Abstract Syntax Notation One (ASN.1).

This section is largely concerned with organizational issues and administrative policy; it neither specifies the objects which are managed, nor the protocols used to manage those objects. These concerns are addressed by other sections.

This section is based on RFC 1155.

3.2 THE NEMA AUTHORITY

All the areas under the *nema* node shall be under NEMA control and constitute the NEMA MIB (See Figure 3-1). In context this NEMA tree is a subtree within the Internet tree which is itself a subtree within the ISO tree.

NEMA defines what is in the *nema* subtree. SMI defines the specification of objects and also where all other MIBs or MIB modules are attached. New MIB modules utilized within the pre-existent tree are called MIB extensions. These do not require modification of the SMI. The only time the SMI is modified is if one of the basic groups (branches or subtrees) with the SMI is modified or a change is made to the definition or structure of object types. NEMA_SMI is an extension to the SMI. The NEMA_SMI does not define any objects, it only defines nodes upon which other MIBs may place objects.

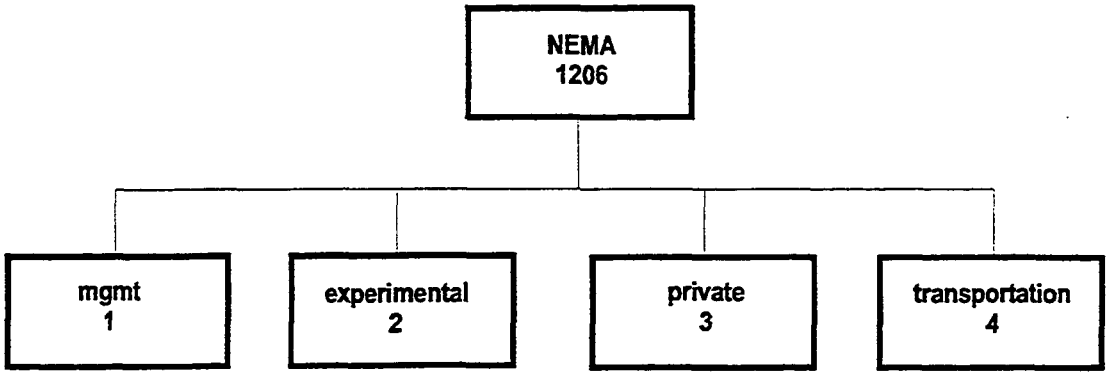


Figure 3-1
NEMA NODE

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

The OBJECT IDENTIFIERs for the major groups of the NEMA_SMI shall be:

1.3.6.1.4.1.1206.mgmt(1)
1.3.6.1.4.1.1206.experimental(2)
1.3.6.1.4.1.1206.private(3)
1.3.6.1.4.1.1206.transportation(4)

All information under *transportation* is part of the TMIB, not the NEMA_SMI.

Additional objects or groups can be added at any time. The NEMA_SMI is maintained by NEMA. NEMA may designate areas as *not subject to NEMA* authority and NEMA can define methods for adding to this structure without NEMA's involvement. This is similar to what the Internet does with the *private.enterprises* section of the SMI. Each manufacturer is given a section of the tree, after which it is up to the manufacturer to define it. This allows the definition of device specific MIBs to be delegated to an autonomous group. Since all information defined by the autonomous group, would be within the delegated area, e.g., *experimental.widget*, it is ensured to be uniquely identified.

3.3 STRUCTURE AND IDENTIFICATION OF MANAGEMENT INFORMATION

Managed objects shall be accessed via the Management Information Base (MIB). Objects in the MIB shall be defined using Abstract Syntax Notation One (ASN.1) and shall be in conformance with RFC 1212.

Each object type shall have a name, a syntax, and an encoding. The name shall be represented uniquely as an OBJECT IDENTIFIER. An OBJECT IDENTIFIER shall be an administratively assigned name. The administrative policies discussed in RFC 1155 shall be used for assigning names and identifiers.

The syntax for an object type shall define the abstract data structure corresponding to that object type. For example, the structure of a given object type might be an INTEGER or OCTET STRING. Although in general, any ASN.1 construct should be available for use in defining the syntax of an object type, this standard purposely restricts the ASN.1 constructs which may be used. These restrictions are made solely for the sake of simplicity.

The encoding of an object type determines how instances of that object type are represented when transmitted on the network. Encoding rules are discussed in Section 5.

3.3.1 Names

Names shall be used to identify managed objects. This subclause specifies names which shall be hierarchical in nature. The OBJECT IDENTIFIER concept shall be used to model this notion. OBJECT IDENTIFIERs are a means of identifying some object, regardless of the semantics associated with the object (e.g., a network object, a standards document, etc.). See RFC 1155 for a more complete description of OBJECT IDENTIFIER.

An OBJECT IDENTIFIER is a sequence of integers which are used to traverse a global tree. The tree consists of a root connected to a number of labeled nodes via edges. Each node may, in turn, have branches of its own which are labeled. In this case, we may term the node a subtree. This process may continue to an arbitrary level of depth. A label is a pairing of a brief textual description and an integer. The format used to show this pairing shall be *text*(integer).

In the global tree the root node itself is unlabeled, but has three branches directly under it: one node is administered by the ISO, with label *iso*(1); another is administrated by the CCITT, with label *ccitt*(0); and the third is jointly administered by the ISO and the CCITT, *iso-ccitt*(2).

Under the *iso(1)* node, the ISO has designated one subtree for use by other (inter)national organizations, *org*(3). Of the children nodes present, two have been assigned to the U.S. National Institutes of Standards and Technology (NIST). One of these subtrees has been transferred by the NIST

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

to the U.S. Department of Defense, *dod*(6). The Internet has been assigned a node under *dod* labeled *internet*(1). This node is administered by the Internet Activities Board (IAB). The IAB has assigned a node called *private*(4) which is used to define nodes that have the administration transferred to other organizations. Under *private* a node called *enterprises(1)* has been defined to hold nodes that are assigned to private enterprises. NEMA has been assigned a node called *nema*(1206) under the *enterprises* node. The IAB has transferred authority for the *nema* node to NEMA.

In ASN.1 syntax, the definition of the *nema* node shall be:

```
nema OBJECT IDENTIFIER ::=
{ iso(1) org(3) dod(6) internet(1) private(4) enterprises(1) 1206 }
```

Four nodes have been assigned under the *nema* node. The definitions shall be:

a. mgmt OBJECT IDENTIFIER ::= { nema 1}
b. experimental OBJECT IDENTIFIER ::= { nema 2}
c. private OBJECT IDENTIFIER ::= { nema 3 }
d. transportation OBJECT IDENTIFIER ::= { nema 4 }

Therefore the full object identifier for *mgmt* would be: *iso.org.dod.internet.private.enterprises.nema*.1 or in numeric values only 1.3.6.1.4.1.1206.1. The periods are used to separate the text(integer) for each identifier to unambiguously show the hierarchy.

3.3.1.1 Mgmt

The *mgmt(1)* subtree is used to identify objects which are defined in NEMA-approved documents.

3.3.1.2 Experimental

The *experimental(2)* subtree is used to identify objects used in NEMA experiments. New MIBs, prior to being assigned within the TMIB, are put here. In the Internet community multiple manufacturers must utilize a MIB within the experimental area before it can move to a permanent location. As a part of the assignment process, NEMA will make requirements as to how that subtree is used.

For example, NEMA experiments might include an initial proposed MIB for a loop-detector or any other previously un-MIBed device. An experimenter might receive number 17, and would have available the OBJECT IDENTIFIER { experimental 17} or 1.3.6.1.4.1.1206.2.17.

3.3.1.3 Private

The *private(3)* subtree is used to identify objects defined unilaterally. Administration of the *private(3)* subtree is by NEMA. NEMA will assign nodes to enterprises for the purpose of defining enterprise specific MIBs.

Upon receiving a node the enterprise may, for example, define new MIB objects under this node. In addition, it is strongly recommended that the enterprise will also register its transportation devices under this subtree, in order to provide an unambiguous identification mechanism for use in management protocols. For example, if the "ABC, Inc." enterprise produced transportation devices, then they could request a node under the *private* subtree from NEMA. Such a node might be numbered:

1.3.6.1.4.1.1206.3.42

The "ABC, Inc." enterprise might then register their "Widget Controller" under the name of 1.3.6.1.4.1.1206.3.42.1 ensuring a unique identification. Thereafter, each enterprise is responsible for ensuring unique identification of information objects within their subtree. NEMA delegates the role of assigning numbers under each *private* node to those to which they are assigned, except of course for the initial enterprises number.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

3.3.1.4 Transportation

All objects within the Transportation MIB (TMIB) shall be put here. The TMIB subtrees are defined in Section 4. This SMI creates the *transportation* node of the *nema* subtree, as shown below.

transportation OBJECT IDENTIFIER ::= { nema 4 }

The *transportation*(4) subtree is used by the NEMA Transportation Management Systems and Associated Control Devices Section to create subtrees for different classes of transportation equipment. The objects defined under each of these subtrees are standard objects specific to each class of transportation device. Please refer to Section 4 for more information on the TMIB.

3.3.2 Syntax

Syntax is used to define the structure corresponding to object types. RFC 1155, ASN.1 constructs shall be used to define this structure, although the full generality of ASN.1 as discussed in RFC 1155 is not permitted.

3.3.2.1 Primitive Types

Only the ASN.1 primitive types INTEGER, OCTET STRING, OBJECT IDENTIFIER, and NULL shall be permitted. These are sometimes referred to as non-aggregate types.

3.3.2.2 Guidelines for Enumerated INTEGERs

If an object type is listed as an enumerated INTEGER, then a named number having the value zero shall not be present in the list of enumeration. Use of this value is prohibited.

3.3.2.3 Constructor Types

The ASN.1 constructor type SEQUENCE shall be permitted only to generate either lists or tables. Only the short or long definite length format shall be permitted per RFC 1155.

For lists, the syntax shall take the form:

SEQUENCE { *<type1>*, ..., *<typeN>* }

where each *<type>* shall resolve to one of the ASN.1 primitive types listed in 3.3.2.1. The DEFAULT and OPTIONAL clauses shall not appear in the SEQUENCE definition.

For tables, the syntax shall take the form:

SEQUENCE OF *<entry>*

where <entry> shall resolve to a list constructor. Lists and tables are referred to as aggregate types.

3.3.3 Managed Objects

Although it is not the purpose of this section to define objects in the MIB, RFC 1212 does specify a format to be used by documents which define objects. All objects shall be defined using the OBJECT-TYPE macro described in RFC 1212.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

The OBJECT-TYPE macro definition shall contain at least the following five fields:

a. Object Name
b. Syntax
c. Access
d. Status
e. Description

3.3.3.1 Object Name

This field shall consist of a textual name, termed the OBJECT DESCRIPTOR, for the object type being defined. It shall precede the invocation of the OBJECT-TYPE macro. This Object Name takes on the value of the specified OBJECT IDENTIFIER.

3.3.3.2 Syntax

This must resolve to an instance of the ASN.1 type *ObjectSyntax* as described in RFC 1212.

3.3.3.3 Access

Per RFC 1212 shall be one of:

a. read-only
b. read-write
c. write-only
d. not-accessible

3.3.3.4 Status

The object status shall indicate its current status.

Status shall be one of:

a. mandatory
b. optional
c. obsolete
d. deprecated

A status of mandatory shall mean that any agent that implements this MIB module must implement this object. A status of optional shall mean that this object may be optionally implemented in an agent. A status of obsolete shall mean that this object is no longer used and agents are no longer required to implement it. A status of deprecated shall mean that the object is being discouraged and in future releases of the standard may be marked obsolete.

3.3.3.5 Description

Description is an ASCII-NVT string which defines the semantics of the object type. The object description shall provide an unambiguous definition of what the object does. The description field shall be present in all OBJECT_TYPE definitions.

3.3.4 Extensions to the MIB

Every NEMA-Standard MIB document obsoletes all previous such documents. New versions shall:

a. declare old object types deprecated, but not delete their names;
b. declare previously deprecated object types obsolete (if necessary), but not delete their names;
c. augment the definition of an object type corresponding to a list by appending non-aggregate object types to the object types in the list;
d. define entirely new object types.

New versions shall not change the semantics of any previously defined object without changing the name of that object.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

3.3.5 Groups

A group is a basic unit of conformance and is used to specify a collection of related managed objects. The group designation applied to a set of objects provides a systematic means for determining which objects are required to support a function. If a device has multiple functions, a group will be defined for each function. Group definitions will be found in the NTCIP Object Definition Standard documents. The object definition Standard may define a group with objects that are not in lexicographic order and only apply to devices of that type.

The related managed objects of a group may include mandatory and/or optional objects. Mandatory objects within a group shall be implemented. Optional objects shall be implemented only if a defined function of the device requires that particular object.

For example, assume a device implements an asynchronous RS-232 interface. It must implement all the mandatory objects in the asynchronous group of the RS-232 MIB. It would not have to implement the synchronous group of objects unless it also provided a synchronous interface.

Assume also that the asynchronous group has a *CRC error counter* object that is optional. The *CRC error counter* object would not have to be implemented unless the device used CRC checking on the asynchronous interface.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Section 4
TRANSPORTATION MANAGEMENT INFORMATION BASE

The Transportation MIB (TMIB) provides a framework for locating objects for transportation specific equipment. The TMIB is located under the *nema* node of the global object tree. This MIB is controlled by NEMA Transportation Management Systems and Associated Control Devices Section (3TS). NEMA TS may designate subnodes of the TMIB to be administered by other groups or organizations.

4.1 TRANSPORTATION TYPE DEFINITIONS

The TMIB defines several ASN.1 data types to help clarify information about various objects that can be expected to reside in transportation devices.

4.1.1 Byte and UByte Types

The Byte and UByte types shall be used to define single octet size integers, where Byte is a signed integer that ranges from -127 to +127, and UByte is an unsigned integer that ranges from 0 to 255.

4.1.2 Short and UShort Types

The Short and UShort types shall be used to define double octet size integers, where Short is a signed integer that ranges from -32767 to +32767, and UShort is an unsigned integer that ranges from 0 to 65535.

4.1.3 Long and ULong Types

The Long and ULong types shall be used to define quad octet size integers, where Long is a signed integer that ranges from -2147483647 to +2147483647, and ULong is an unsigned integer that ranges from 0 to 4294967295.

4.1.4 EntryStatus Type

The EntryStatus type shall be used to manage tables that allow new rows to be created by management applications running remotely. For each row in the table there shall be a columnar object that is defined with a SYNTAX of EntryStatus (e.g., dynamic object definitions). EntryStatus shall be an enumerated INTEGER that can have one of four values: *valid*(1), *createRequest*(2), *underCreation*(3) and *invalid*(4). Refer to RFC 1271 for more information on EntryStatus types.

Other objects in the row shall have operations limited by the current value of the EntryStatus object in the row. The meaning of the values is as follows:

If the status object is *invalid*(4) then the information in this row shall be considered undefined. Setting the status object to invalid has the effect of invalidating the corresponding row. It is implementation specific whether the agent removes an invalidated row from the table. Therefore, managers must be prepared to receive tabular information from agents that correspond to invalid rows. Proper interpretation of such rows requires examination of the relevant EntryStatus object.

An existing EntryStatus object cannot be set to *createRequest*(2). Rows may only be set to *createRequest*(2) when the they are created. When this object is created, the agent may wish to create supplemental object instances to complete a conceptual row in this table. After completing the create operation, the agent must set this object to *underCreation*(3).

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Rows shall exist in the *underCreation*(3) state until the management station is finished configuring the row and sets the EntryStatus object to *valid*(1), or aborts and sets this object to *invalid*(4). If the agent determines that an entry has been in the *underCreation*(3) state for an abnormally long time, it may set this object to *invalid*(4) to reclaim the row.

A management station may create a row, configure all objects in the row and set the corresponding EntryStatus object to *valid*(1) in a single set operation.

4.1.5 OwnerString Type

The OwnerString type shall be used to allow the owner of a resource to store information about themselves in the object. This information must be taken from the NVT ASCII character set These objects shall be used to store information about who has control of the associated resources. The object shall be capable of storing 127 NVT ASCII characters.

4.2 STRUCTURE OF TRANSPORTATION MANAGEMENT INFORMATION

The TMIB defines very few objects. It defines the tree structure upon which other Standards place objects. Annex B contains the complete ASN.1 description of the TMIB. In ASN.1 notation the *transportation* node is defined as:

transportation OBJECT IDENTIFIER ::= { nema 4 }

The TMIB shall have two nodes located under the *transportation* node. The *protocols* node shall be the beginning of a subtree that holds information about various communications protocols. The *devices* node shall be the beginning of a subtree that holds information about various standard transportation device objects. The ASN.1 notation for these two nodes shall be defined as:

protocols OBJECT IDENTIFIER ::= { transportation 1 }
devices OBJECT IDENTIFIER ::= { transportation 2 }

4.2.1 Protocols subtree

The *protocols* subtree contains nodes that have information about specific implementations of STMP on different protocols. The protocols tree shall have three nodes in it which are defined as:

a. layers—OBJECT IDENTIFIER ::= { protocols 1 }
b. profiles—OBJECT IDENTIFIER ::= { protocols 2 }
c. dynObjMgmt—OBJECT IDENTIFIER ::= { protocols 3 }

4.2.1.1 Dynamic Object Management (dynObjMgmt)

The *dynObjMgmt* subtree is used by the STMP to reduce the bandwidth requirements of communicating sets of objects between a management station and agents. The *dynObjMgmt* tree contains two nodes, *dynObjDef* and *dynObjData.*

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

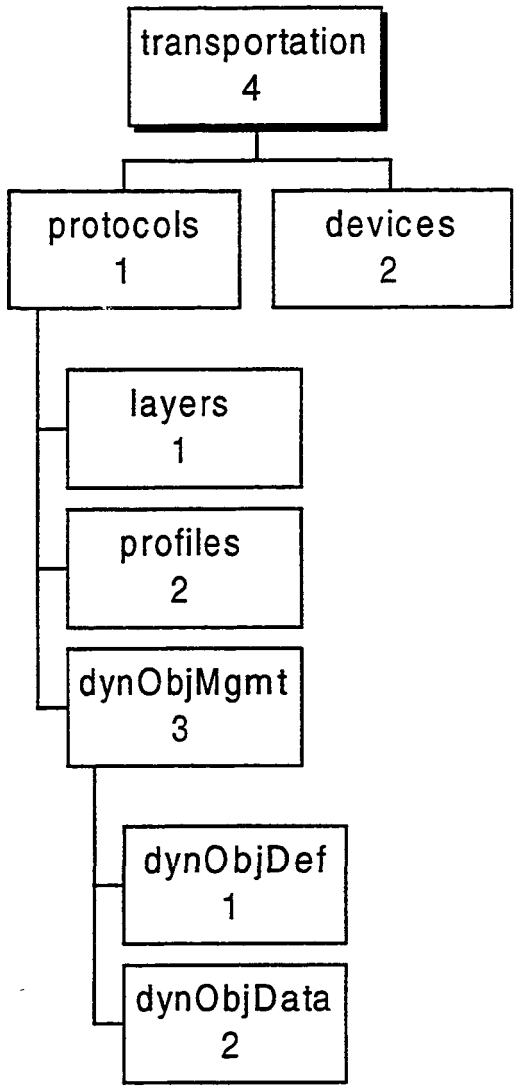


Figure 4-1
TMIB TREE STRUCTURE

4.2.1.1.1 Dynamic Object Definition (dynObjDef)

The *dynObjDef* node is a table of OBJECT IDENTIFIERs that make up the 13 dynamic objects supported by STMP. Each row (entry) of the table shall identify a specific object within a dynamic object, and its position(order) in the dynamic object. The dynamic object definition table has rows that contain the following objects:

dynObjNumber	dynObjIndex	dynObjVariable	dynObjOwner	dynObjStatus

The INDEX for a particular row in the *dynObjDef* table is defined as a concatenation of both the *dynObjNumber* and *dynObjIndex* elements

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

The *dynObjNumber* shall be of type **INTEGER (1...13).** It identifies which of the 13 dynamic objects this row of the table is associated with. This field shall consist of an integer between 1 and 13. Values other than these will cause an error to be returned.

The *dynObjIndex* shall be of type **Byte.** It identifies the position in which the object pointed to by the *dynObjVariable* field will be located in the specified dynamic object. This value shall start at 1 for the first object in the dynamic object and shall increase by one for each additional object defined for the specified dynamic object. During dynamic object creation, values that are not sequential shall cause an error to be returned. An agent shall allow set operations with a null in the value field of *dynObjIndex* to be automatically set to the next sequential *dynObjIndex* value. The agent shall verify that the desired *dynObjIndex* value is the next sequential value for this dynamic object number. If it is not then an error of *badValue* shall be returned. If the value is the next sequential value or if the *dynObjIndex* value is set to null, then the agent shall set the *dynObjIndex* to the next sequential value for this dynamic object number. During dynamic object deletion, rows can be marked invalid in any order.

The *dynObjVariable* shall be of type **OBJECT IDENTIFIER.** It must point to an object supported by the device.

The *dynObjOwner* shall be of type **OwnerString.** It shall be used to determine who is the owner of this object. An agent shall return an error during a set operation if the *dynObjOwner* element is different from previously defined rows for the same *dynObjNumber*. This is to ensure that two management stations do not attempt to define the same dynamic object simultaneously. The default value shall be null.

The management station may optionally set the *dynObjOwner* to any value it desires. However the management station shall use the same *dynObjOwner* value for all rows that define the same dynamic object. The agent will ensure that the *dynObjOwner* element of a row is set to null when the row is invalid, so management stations that do not set a *dynObjOwner* value will not receive errors due to values of *dynObjOwner* elements that do not match. The agent shall return a *badValue* error if a set operation on the *dynObjOwner* element does not match the *dynObjOwner* elements of previously defined of the same dynamic object.

The *dynObjStatus* shall be of type **EntryStatus.** It shall be used as described in 4.1.4. When the row is marked *invalid(4)* it shall not participate in the dynamic object operations. The management station shall set the value of *dynObjStatus* for the newly defined row to *valid*(1) once all the elements are properly set. Once the row is set to *valid*, any requests to the dynamic object data shall include all the objects defined in the *dynObjDef* table for the associated dynamic object.

When a management station is finished using the dynamic object, it may set all the rows that define the dynamic object to *invalid(4).* If a management station sets the first row (index = 1) to *invalid*, then the agent shall set all rows for the corresponding dynamic object to *invalid.* If the dynObjOwner field does not match on the set request, then an error of *badValue* shall be returned.

4.2.1.1.2 Dynamic Object Data (dynObjData)

This tree shall contain 13 objects, where each object corresponds to one of the 13 dynamic objects. The information contained within these objects shall be determined by corresponding rows in the *dynObjDef* table.

The STMP is specifically designed to properly interpret the values of these dynamic objects. STMP can properly separate the actual objects that are contained within the dynamic object. If SNMP is used to get or set these objects, it will treat them as a single large OCTET STRING.

The thirteen objects are defined identically except for the integer used in the name, description field and OBJECT IDENTIFIER fields (See Annex B).

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

An STMP get request of *dynObjN* is equivalent to an SNMP get of each of the individual objects that are defined (in the *dynObjDef* table) for dynamic object *N*. Likewise an STMP set request of *dynObjN* is equivalent to an SNMP set of each of the individual objects that are defined for dynamic object N. Management applications shall ensure that the dynamic objects are properly defined prior to using them.

4.2.1.2 Profiles subtree

This subtree shall be used for standard objects that affect or describe various communications profiles that may be defined for use with STMF. The definition of the objects in this subtree is left to other documents. Each profile must be assigned a number under this node by NEMA and all objects defined under these nodes must be reviewed by NEMA before they are used.

4.2.1.3 Layers subtree

This subtree shall be used for standard objects that affect or describe various communication protocol layers that may be defined for use with STMF. The definition of the objects in this subtree is left to other documents. Each layer must be assigned a number under this node by NEMA and all objects defined under these nodes must be reviewed by NEMA before they are used.

4.2.2 Devices subtree

This subtree shall be used for standard objects that are defined for specific types of transportation equipment. Each type of equipment (i.e. actuated traffic controller, variable message sign, camera control, etc.) will have it's own subtree in the devices tree. The definition of the objects under this tree is left to other documents. Each device type must be assigned a number under this node by NEMA and all objects defined under these nodes must be reviewed by NEMA before they are used.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Section 5
PROTOCOL DEFINITIONS

Applications conforming to this standard shall support the Simple Transportation Management Protocol (STMP) and optionally support the Simple Network Management Protocol (SNMP). The conformance level of a device shall determine whether it supports only STMP or both STMP and SNMP.

The Protocol Data Unit (PDU) shall consist of a PDU Header field and when required, a PDU Information field. The high order bit of the PDU Header field shall designate the PDU format as follows:

BIT	CONTENTS	Description
7	*PDU Format*	
	0	Format dependent upon seven lower order bits
	1	Standard STMP format

If the PDU Format bit is 0, then the seven low order bits of the STMF PDU Header field are defined as follows:

BIT	CONTENTS	Description
6-0	*Detailed PDU Format*	
	000 0000 - 010 1111	Reserved
	011 0000	Standard SNMP Format
	011 0001 - 111 1111	Reserved

If the PDU Header field equals 0x30 (Standard SNMP format) then the PDU Header field shall be considered the first byte of the SNMP message.

5.1 SIMPLE TRANSPORTATION MANAGEMENT PROTOCOL

The STMP is modeled after the SNMP, however, several features are modified so that the STMP can provide more efficient operation.

5.1.1 Protocol Data Unit

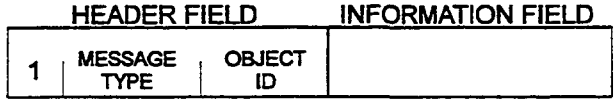


Figure 5-1
STMP PDU FIELDS

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

For STMP the PDU Header field shall conform to the following format:

BIT	CONTENTS	Description
7	*PDU Format*	
	1	Standard STMP format
6-4	*Message Type*	
	000	GET request is contained in the packet.
	001	SET request is contained in the packet.
	010	SET request no reply is contained in the packet.
	011	GET NEXT request is contained in the packet.
	100	GET response is contained in the packet (positive ACK).
	101	SET response is contained in the packet (positive ACK).
	110	ERROR response is contained in the packet.
	111	TRAP response is contained in the packet.
3-0	*Object ID*	
	0000	NEMA node used as root
	0001-1101	ID of STMP "dynamic object"
	1110	absolute root used
	1111	reserved

5.1.1.1 PDU format bit

The value of the PDU Format bit shall be used to determine if the protocol is STMP as described in 0.

5.1.1.2 Message type bit field

The message type field shall be used to determine what type of STMP message is contained in the packet. There are eight types of messages: *get request, set request, set request no reply, get next request, get response, set response, error response*, and *trap response.*

A *get request* shall be sent from a manager to an agent to request the values of specified objects.

A *set request* shall be sent from a manager to an agent to request that specified objects be set to specified values.

A *set request no reply* shall be sent from a manager to an agent to request that specified objects be set to specified values.

A *get next* request shall be sent from a manager to an agent to request the value of the next object instance in lexicographic order.

A *get response* shall be used to send specified object data from an agent to a manager. This response contains the object values that correspond to the prior *get request* or *get next* request from the manager. *Get response* messages shall not be sent unless the agent has received a *get request* packet from the manager.

A *set response* shall be sent from an agent to the manager to indicate that the *set request* for the specified objects was completed without error.

The *get error response* shall be sent from the agent to the manager in response to a get request that contained an error. The error information will follow the PDU byte as described in 0.

An *error response* shall be sent from the agent to the manager in response to a *set request, set request*, or *get next* request that contained an error.0

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

A *trap response* shall be used to send specified object data from an agent to a manager. The trap is generated due to some event occurring within the agent, and is not a response to any manager request.

The STMP *trap response* message is under development and will be completed under a future revision of this Standard.

5.1.1.3 Object ID field

The object ID field shall be used to determine how the following information in the packet is encoded. STMP supports two encoding rules (BER and PER) and provides some compression of OBJECT IDENTIFIERs to reduce the bandwidth requirements of the protocol.

If the value of the OBJECT ID field is 0x0 then the following information in the packet shall be encoded using normal BER except that the OBJECT IDENTIFIERs shall not have the prefix *iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).nema(1206)* encoded with them. This reduces each OBJECT IDENTIFIERs length by seven octets, but limits those objects that can be accessed to the *nema* subtree.

If the value of the OBJECT ID field is in the range 1...13 (0x1...0xD) then the following information in the packet is encoded using Packed Encoding Rules (PER). Only one of thirteen dynamic objects can be accessed in this mode. The value of the OBJECT ID field selects one of the thirteen objects under the *iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).nema(1206).transportation(4).protocols(1).dynObjMgmt(3).dynObjData(2).*

The *dynObjData* objects can be defined to point to any objects supported by the agent. This is accomplished using the *dynObjDef* table to specify the OBJECT IDENTIFIERs that will make up the dynamic object. Thirteen sets of objects can be defined and operated on with minimal overhead. The *dynObjDef* table is discussed in section 4 *Transportation Management Information Base*. In addition the PER eliminates other redundant information to further reduce the amount of data that is transmitted to operate on objects. The PER is discussed in 5.1.2.2 below.

If the value of the OBJECT ID field is 14 (0xE) then the following data in the packet shall be encoded using normal BER.

If the value of the OBJECT ID field is 15 (0xF) then the packet shall be ignored.

All STMP messages shall conform to the Operations and Semantics specified in 5.1.2.

5.1.1.4 PDU Information Field

The STMP PDU Information field shall use the same common constructs as defined in 4.1.1 of RFC 1157 with the following additional constructs:

```
DynObjVar ::=
    value
      ObjectSyntax

DynObjVarList ::=
    SEQUENCE OF
      DynObjVar
```

The form of the PDU Information field for a *get request, set request, set request no reply, get next request*, and *get response* shall conform to one of the following formats, as appropriate:

a. Variable length values shall be encoded with length.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

b. If the Object ID field of the PDU Header field is in the range 1...13 (0x1...0xD), the PDU Information field shall only contain the sequence of values of the objects referenced by the dynObjVariables. This is equivalent to DynObjVarList. Variable length strings shall be encoded with length.

c. If the Object ID field of the PDU Header field is 0 (0x0) or 14 (0xD) and if the request is for a single object, the PDU Information field shall only contain the name and value of the object. This is equivalent VarBind as defined in RFC 1157.

d. If the Object ID field of the PDU Header field is 0 (0x0) or 14 (0xD) and if the request is for multiple objects, the PDU Information field shall only contain the sequence of name-value pairs. This is equivalent VarBindList as defined in RFC 1157.

The PDU Information field of an STMP *error response* shall only contain the error status and error index:

```
STMPErrorResponse ::=
    IMPLICIT SEQUENCE {
        error-status
          ErrorStatus,

        error-index
          ErrorIndex
    }
```

Set response messages shall not include a PDU Information field.

The form of the PDU Information field for an STMP *trap response* is still under development and will be completed under a future revision of this Standard.

5.1.1.5 Error responses

STMP shall not return the information that was included in the request packet when an error is returned. STMP shall return an error number and an index number only. The manager should retain the last request transmitted to an agent in order to determine what part of the request caused the error.

The error number shall be encoded as a single octet. A type and length are not encoded. The possible errors shall be:

a. *tooBig(1):* this error is returned if the PDU was larger than expected. The index number shall be set to zero.

b. *noSuchName(2):* the object identifier indicated by the index number is not supported by the agent.

c. *badValue(3):* this error can only occur during a set operation. The object indicated by the index number value shall be the first that is not valid (out of range).

d. *readOnly(4):* this error can only occur during a set operation. The index number indicates which object could not be written.

e. *genErr(5):* this error indicates that some other error has occurred that does not conform to one of the specified errors above. It is application specific and requires referencing the agents documentation to determine what the error may be.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

The index number shall be encoded as a single octet for index number values ranging from 0 through 127 (0x7F) For index values larger than 127, the first octet shall have the most significant bit set to 1 and the remaining seven bits set to indicate how many octets follow for the index number. For example for an index value of 192 (0xC0) the index number would be encoded as two octets with the values 0x81 and 0xC0 This encoding is identical to the length encoding used by BER. Since most requests contain less than 127 fields, it should be rare to encounter an index number that exceeds a single octet.

5.1.2 Operations and Semantics

The operations of the STMP are identical to SNMP with the following differences:

a. STMP *get requests* and *get next requests* shall not include the type, length, or value as a place holder for the object value.

b. When a STMP *get request, get next request,* or *set request* message is rejected by an agent, the STMP *error response* PDU Information field shall only consist of an error number and index number as specified in 5.1.1.5.

c. An agent receiving a *set request noreply* shall not respond under any condition.

d. The OBJECT-IDENTIFIER fields shall be interpreted as specified in 5.1.1.3.

5.1.2.1 Basic Encoding Rules (BER)

Once an instance of an object type has been identified, its value shall be transmitted by applying the basic encoding rules of ASN.1 to the syntax for the object type. BER is specified in ISO/IEC 8825.

5.1.2.2 Packed Encoding Rules (PER)

The PER only works with STMP and only on dynamic objects. The STMP utilizes a more compact encoding of data to transmit *dynamic objects.* This compact encoding is important to improve the bandwidth utilization for sequences of objects that are frequently requested.

Normal BER uses a three *tuple* to encode data for transmission. The first element of the tuple is an octet that specifies what type of data follows. The second element of the tuple is an octet(s) that specifies how many octets the data occupies. The third and final element of the tuple is the actual data being transmitted. This provides a very flexible method of encoding information for transmission. This encoding is sometimes referred to as *TLV* encoding which stands for type length value.

PER takes advantage of the definition of dynamic objects to eliminate certain elements of the BER encoding. Since dynamic objects must be created at run time, both the manager and the agent know what each type is and for types of fixed length (e.g. Integer32) what the length of the data is. This allows the PER encoding to eliminate the type element and on fixed length data objects the length element can be eliminated as well.

Some rules can be applied to determine the proper encoding based on BER.

a. All TYPE fields shall be removed in the PDU Information field.

b. All objects that resolve to the Primitive type NULL or INTEGER that are explicitly sized shall be encoded as a fixed length field. The length is determined by the maximum value that an object can support. For example, the maximum size of an INTEGER (SIZE (4)) is four octets and the maximum size of an INTEGER (0..255) is one octet.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

c. All objects that resolve to the Primitive type OCTET STRING or OBJECT IDENTIFIER or unsized INTEGER shall be encoded as a variable length field.

d. If the type is fixed length then the length information shall be removed.

e. If the type is variable length, except for SEQUENCE and SEQUENCE OF, then the length information shall be encoded identically to the BER. If the type is SEQUENCE or SEQUENCE OF then the sequence information shall be removed.

5.2 SIMPLE NETWORK MANAGEMENT PROTOCOL

SNMP shall conform to RFC 1157.

5.3 TRANSPORT MAPPINGS

STMP protocol can operate over a connectionless protocol. However, like SNMP, STMP has no checksum or CRC and relies on a lower layer protocols to detect and discard corrupted packets.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Section 6
CONFORMANCE LEVEL

6.1 GENERAL CONFORMANCE

Devices shall conform to either level 1 or 2 as described in 6.1.1 and 6.1.2.

Minimum and maximum ranges different from the values of the SYNTAX field may be enforced by an application running on a device.

A device which enforces range limits within the bounds specified by the values of the SYNTAX field shall not be categorized as being non-compliant with NTCIP.

6.1.1 Conformance Level 1

In order to be at conformance level 1 the following shall be supported:

a. STMP: All STMP operations shall be supported.

b. TMIB: All mandatory objects and groups defined within the TMIB shall be present.

c. Dynamic objects including: definition creation; definition deletion and object get and set data functions.

6.1.2 Conformance Level 2

In order to be at conformance level 2 the following shall be supported:

a. Conformance level 1

b. SNMP version 1: All supported objects including Internet standard objects are accessible. This level shall support the Internet system (group) objects.

6.2 MANAGEMENT APPLICATION CONFORMANCE

Management Applications shall conform to either Conformance level 1 or 2 as described in 6.1.1 and 6.1.2. In addition, management applications must be able to read ASCII text ASN.1 MIB modules such that the management applications may update their MIB views when appropriate.

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
(Normative)
SMI FOR NEMA

```
NEMA_SMI DEFINITIONS ::= BEGIN

-- Initial definitions for NEMA Structure of Management Information.
-- This contains only structure information, no managed objects are defined

IMPORTS
    enterprises
        FROM RFC1155-SMI


EXPORTS -- EVERYTHING
-- Major groups of the NEMA_SMI
    nema, nema.mgmt, nema.experimental, nema.private, transportation

-- NEMA has received ID 1206 from IANA
-- NEMA starts at { iso 3 6 internet private enterprises 1206 } in the global naming tree.

    nema            OBJECT IDENTIFIER ::= { enterprises 1206 }

    mgmt            OBJECT IDENTIFIER ::= { nema 1 }
    experimental        OBJECT IDENTIFIER ::= { nema 2 }
    private         OBJECT IDENTIFIER ::= { nema 3 }
    transportation      OBJECT IDENTIFIER ::= { nema 4 }

-- The mgmt subtree is used for standard NEMA object definitions that span
-- different NEMA sections.
-- The experimental subtree is used for technical committees developing objects
-- that may become standard in the future.
-- The private subtree is used for unilaterally defined objects. One common use is
-- for manufacturer specific or customer specific MIB definitions.
-- The transportation subtree is used by the NEMA 3TS Section to define standard
-- objects specific for the transportation industry.

END -- NEMA_SMI Definitions
```

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(Normative)
TRANSPORTATION MANAGEMENT INFORMATION BASE

```
    TMIB DEFINITIONS ::= BEGIN
--
-- Initial MIB for Standard Transportation Management Framework
-- structure of management information (STMF).
--
IMPORTS
    ObjectName
        FROM RFC1155-SMI
    DisplayString
        FROM RFC1158-MIB
    OBJECT-TYPE
        FROM RFC1212
-- NEMA has received ID 1206 from IANA
    nema, transportation
        FROM NEMA_SMI

EXPORTS -- EVERYTHING
-- Major groups of the TMIB.
    protocols, dynObjMgmt, dynObjDef, dynObjData, profiles, layers, devices
    dynObjEntry, dynObjNumber, dynObjIndex, dynObjVariable, dynObjOwner,
    dynObjStatus, dynObj1, dynObj2, dynObj3, dynObj4, dynObj5, dynObj6,
    dynObj7, dynObj8, dynObj9, dynObj10, dynObj11, dynObj12, dynObj13;

--This MIB module uses the extended OBJECT-TYPE macro as
-- defined in RFC 1212;


---- ************************************************************
----      Special Object Type Definitions
---- ************************************************************


-- The following are subtypes of conventional types formed by
-- subtyping ASN types.


Byte ::=    INTEGER (-127..127)
UByte ::=   INTEGER (0..255)

Short ::=   INTEGER (-32782..32782)
UShort ::=  INTEGER (0..65535)

Long ::=    INTEGER (-2147483647..2147483647)
ULong ::=   INTEGER (0..4294967295)
```

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

```
EntryStatus ::= INTEGER
    { valid (1),
      createRequest (2),
      underCreation (3),
      invalid (4)
    }

-- The EntryStatus shows the status of a table entry
-- Setting this object to the value invalid(4) has the effect of invalidating the
-- corresponding entry. That is, it effectively disassociates the mapping
-- identified with said entry.
-- It is an implementation specific matter as to whether the agent removes
-- an invalidated entry from the table. Accordingly managers must be
-- prepared to receive tabular information from agents that corresponds to
-- entries currently not in use. Proper interpretation of such entries
-- requires examination of the relevant EntryStatus object.
--
-- An existing instance of this object cannot be set to createRequest(2). This
-- object may only be set to createRequest(2) when the instance is created.
-- When this object is created, the agent may wish to create supplemental object
-- instances to complete a conceptual row in this table. Immediately after
-- completing the create operation, the agent must set this object to underCreation(3).
--
-- Entries shall exist in the underCreation(3) state until the management station is
-- finished configuring the entry and sets this object to valid(1) or aborts, and sets this
-- object to invalid(4). If the agent determines that an entry has been in the
-- underCreation(3) state fro an abnormally long time, it may decide that the
-- management station has crashed. If the agent makes this decision, it may set
-- this object to invalid(4) to reclaim the entry. A prudent agent will understand that
-- the management station may need to wait for human input and will allow for that
-- possibility in its determination of this abnormally long period.


OwnerString ::= DISPLAY STRING ( size (0..127))
-- This data type is used to model an administratively assigned name of the owner of
-- a resource. This information is taken from the NVT ASCII character set.
-- It is suggested that this name contain one or more of the following:
-- management station name, manager's name, location or phone number.
--
-- SNMP access control is articulated entirely in terms of the contents of MIB views;
-- access to a particular SNMP object instance depends only upon its presence or
-- absence in a particular MIB view and never upon its value or the value of related object
-- instances. Thus, objects of this type afford resolution of resource contention only among
-- cooperating managers; they realize no access control function with respect to
-- uncooperative parties.
--
-- By convention, objects with this syntax are declared as having
--
--   SIZE ( 0..127 )
```

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

```
-- ****************************************************************
--        GROUPS WITHIN THE TMIB
-- ****************************************************************
-- ****************************************************************
--        Protocol Group
-- ****************************************************************
-- The protocols group contains information about specific implementations
-- of different protocols.
--
protocols       OBJECT IDENTIFIER ::= { transportation 1 }

layers      OBJECT IDENTIFIER ::= { protocols 1 }

profiles        OBJECT IDENTIFIER ::= { protocols 2 }

dynObjMgmt   OBJECT IDENTIFIER ::= { protocols 3 }
    dynObjData     OBJECT IDENTIFIER ::= { dynObjMgmt 2 }

--*****************************************************************
--        Device Group
--*****************************************************************
devices         OBJECT IDENTIFIER ::= { transportation 2 }


--*****************************************************************
--        Dynamic Object Group
-- ****************************************************************


dynObjDef OBJECT-TYPE
    SYNTAX  SEQUENCE OF dynObjEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "A list of objects to be included in dynamic objects"
::= { dynObjMgmt 1 }

    dynObjEntry OBJECT-TYPE
        SYNTAX  DynObjEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
            "A list of OBJECT IDENTIFIERs that make up a dynamic object"
        INDEX  { dynObjNumber  dynObjIndex }
    ::= { dynObjDef 1 }

DynObjEntry ::= SEQUENCE {
    dynObjNumber     INTEGER (1..13),
    dynObjIndex      UByte,
    dynObjVariable   OBJECT IDENTIFIER,
    dynObjOwner      OwnerString,
    dynObjStatus     EntryStatus }
```

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

dynObjNumber OBJECT-TYPE
SYNTAX INTEGER (1..13)
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The dynamic object number that this entry is to be associated with. The order in which objects are transmitted for a given dynamic object number is based on the value of the dynObjIndex. Lower values of dynObjIndex are transmitted first (for the same value of dynObjNumber).

This object may not be modified if the associated dynObjStatus object is equal to valid(1)."
::= { dynObjEntry 1 }

dynObjIndex OBJECT-TYPE
SYNTAX UByte
ACCESS read-only
STATUS mandatory
DESCRIPTION
"An index that uniquely identifies an entry in the dynamic object table. Each entry defines an object that is to be associated with a dynamic object number. The dynObjIndex and dynObjNumber together determine the order in which objects are transmitted for the dynamic object. The lower dynObjIndex numbers are transmitted before larger numbers for entries with the same dynObjNumber."
::= { dynObjEntry 2 }

dynObjVariable OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The object identifier of the particular variable to be included in the specified dynamic object number. Care must be taken when defining dynamic objects that the maximum size of all the objects included in a dynamic object do not exceed the maximum packet size of the communications network.

This object may not be modified if the associated dynObjStatus object is equal to valid(1)."
::= { dynObjEntry 3 }

dynObjOwner OBJECT-TYPE
SYNTAX OwnerString
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The entity that configured this entry and is therefore using the resources assigned to it.
This object may not be modified if the associated dynObjStatus object is equal to valid(1)."
::= { dynObjEntry 4 }

dynObjStatus OBJECT-TYPE
SYNTAX EntryStatus

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

```
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "The status of this dynamic object definition entry. See description of
            EntryStatus above for restrictions on accesses"
    ::= { dynObjEntry 5 }

    --*********************************************************************
    --      Dynamic Object Data
    --*********************************************************************

    dynObj1 OBJECT-TYPE
        SYNTAX  OCTET STRING
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "The value of this object is determined by the dynObjDef entries with
            dynObjNumber equal to 1. Packed Encoding Rules are utilized to encode
            the objects for transmission. This object is intended for use with the Simple
            Transportation Management Protocol, and provides little advantage if used
            with SNMP.
            If no objects are defined for this dynamic object number, then an error of
            noSuchName shall be returned by the agent"
    ::= { dynObjData 1 }

    dynObj2 OBJECT-TYPE
        SYNTAX  OCTET STRING
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "The value of this object is determined by the dynObjDef entries with
            dynObjNumber equal to 2. Packed Encoding Rules are utilized to encode
            the objects for transmission. This object is intended for use with the Simple
            Transportation Management Protocol, and provides little advantage if used
            with SNMP.
            If no objects are defined for this dynamic object number, then an error of
            noSuchName shall be returned by the agent"
    ::= { dynObjData 2 }

        dynObj3 OBJECT-TYPE
            SYNTAX  OCTET STRING
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                "The value of this object is determined by the dynObjDef entries with
                dynObjNumber equal to 3. Packed Encoding Rules are utilized to encode
                the objects for transmission. This object is intended for use with the Simple
                Transportation Management Protocol, and provides little advantage if used
                with SNMP.
                If no objects are defined for this dynamic object number, then an error of
                noSuchName shall be returned by the agent"
        ::= { dynObjData 3 }

    dynObj4 OBJECT-TYPE
```

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 4. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 4 }

dynObj5 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 5. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 5 }

dynObj6 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 6. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 6 }

dynObj7 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 7. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 7 }

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

dynObj8 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 8. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 8 }

dynObj9 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 9. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 9 }

dynObj10 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 10. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 10 }

dynObj11 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 11. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

noSuchName shall be returned by the agent"
::= { dynObjData 11 }

dynObj12 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 12. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 12 }

dynObj13 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The value of this object is determined by the dynObjDef entries with dynObjNumber equal to 13. Packed Encoding Rules are utilized to encode the objects for transmission. This object is intended for use with the Simple Transportation Management Protocol, and provides little advantage if used with SNMP.
If no objects are defined for this dynamic object number, then an error of noSuchName shall be returned by the agent"
::= { dynObjData 13 }

END -- TMIB DEFINITIONS

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS
Not for Resale

A joint publication of:

American Association of State Highway and Transportation Officials, 444 N. Capitol St., N.W., Ste. 249, Washington, DC 20001 http://www.aashto.c
Institute of Transportation Engineers, 525 School St., S.W. Suite 410, Washington, D.C. 20004-2797 http://www.vax.ite.org
National Electrical Manufacturers Association, 1300 N. 17th Street, Suite 1847, Rosslyn, VA 22209 http://www.nema.org

Copyright National Electrical Manufacturers Association
Provided by IHS under license with NEMA
No reproduction or networking permitted without license from IHS

Not for Resale