

DATE: April 24, 2009

TO: AASHTO, ITE, and NEMA

SUBJECT: **Recommended Standard of NTCIP 1103 v02,
Transportation Management
Protocols (TMP) – Version 02**

ACTION: Request for Ballot to Approve

RECOMMENDATION. This Bulletin reports that the Joint Committee on the NTCIP accepted the proposed NTCIP 1103 v02, *Transportation Management Protocols (TMP) – Version 02*, as a Recommended Standard, and agreed to refer the document to AASHTO, ITE, and NEMA for balloting and approval. The recommendation started at the regularly scheduled Joint Committee meeting in November 2007, when the Joint Committee instructed the BSP2 WG to revise NTCIP 1103 into two versions, v02 (without Traps for consideration as a proposed Recommended Standard) and v03 (with Traps for consideration as a proposed User Comment Draft).

In their January 2008 teleconference, the Joint Committee authorized an electronic ballot to accept the revised NTCIP 1103 v02.14b, (without Traps), as a Recommended Standard. In their February 2008 Joint Committee teleconference, the e-ballot was reported concluded and won acceptance by the required two-thirds majority of the 18-member committee, as the question to recommend the standard carried with 15 affirmative votes, no negative votes, and 3 members not responding. Therefore, the proposed draft was ACCEPTED by the required 12/18 super majority.

The Joint Committee on the NTCIP contributes to the joint standards development process by accepting a proposed draft document at two stages in its development – first as a User Comment Draft and then again as a Recommended Standard.

BACKGROUND. This document was developed to be a Joint AASHTO / ITE / NEMA Standards Publication, part of the NTCIP family of standards and profiles. The document is in the 1100-series of NTCIP document numbers, and is a Base Standard. NTCIP 1100-series Base Standards define the basic details of data handling – such as procedures and data formats.



The NTCIP 1103 defines a composite, application-layer protocol for the management of transportation equipment. The composite protocol consists of three component protocols: (1) the Internet-standard Simple Network Management Protocol (SNMP), (2) the Simple Fixed Message Protocol (SFMP), and (3) the Simple Transportation Management Protocol (STMP). These three protocols are concerned with the procedures for exchanging information as well as the format in which the information is exchanged.

All three protocols provide the same base services, but are designed for different needs. Also, the NTCIP 1103 standard defines a limited number of data elements necessary to manage these protocols.

When related to the widely known, "seven layer" ISO OSI Reference model, the three protocols in NTCIP 1103 are concerned with the upper three layers (the Application, Presentation and Session layers) in the OSI model.

From 1996 to 2004, parts of NTCIP 1103, including the definition of STMP, were defined in NTCIP 1101 (previously known as NEMA TS 3.2, STMF). However, in order to provide a more organized and modular set of standards, NTCIP 1101 was separated into three distinct standards: (1) NTCIP 1103, TMP, which includes the definition of STMP; (2) NTCIP 1102, which defines the Octet Encoding Rules (OER); and (3) NTCIP 8004, which defines the Structure and Management of Transportation Information (SMI). These three standards completely replaced NTCIP 1101 / NEMA TS 3.2. The STMP defined within NTCIP 1103 is one hundred percent consistent with the STMP definition in NTCIP 1101.

A "Trap" is an event-reporting operation in SNMP, used to report the occurrence of defined events. The Base Standards and Profiles WG did not include "Trap management" in NTCIP 1103 version v01, which was jointly approved in November 2005.

Although NTCIP 1103 v02 is a "major version" revision of the NTCIP 1103, Traps are not included in v02. NTCIP 1103 v02 does include revisions for: the referencing of the report and security node objects, use of the community-name field and security threats, an update of references to related documents, and correction of typographic errors.

Issues remain with the definition of the Trap objects and the use of Traps. The Trap mechanism may be most applicable with higher speed Ethernet communications, and less applicable with the lower speed RS-232 and FSK-connected equipment. Besides being used in actuated signal controllers, there is also potential use for Traps in TSS and DCM devices. Traps are now defined in the draft version v03, to be issued for review and comment.

DOCUMENT ORGANIZATION. The NTCIP 1103 v02 Section 2 describes the overall structure of the Transportation Management Protocols and how the three component protocols merge to form a single identifiable protocol. Section 3 provides an overview of how the Internet standard Simple Network Management Protocol (SNMP) works. Section 4 describes how the Simple Fixed Message Protocol (SFMP) enhances the concepts of SNMP in order to provide for a more compact encoding of data while still providing a simple design but with the loss of some flexibility. Section 5 describes how the Simple Transportation Management Protocol (STMP) provides for a flexible compact encoding of data but with some loss of simplicity.

NTCIP 1103 v02 Section 6 is the placeholder for the specification of "Traps" in version v03. Section 7 discusses when to use NTCIP logical names. Section 8 describes the security issues related to the TMP. Section 9 defines conformance to the standard. Annex A is normative and defines the Transportation Management Protocols Management Information Bases (MIBs). Annex B list deprecated objects. Annex C lists explains relative object identifiers. And Annex D explains the deprecation of Entry Status Type.

NTCIP 1103 v02 was developed by the Joint Committee's Base Standards and Profiles Working Group, which is chaired by Bob De Roche (Bob De Roche Consulting). The BSP2 WG has 10 members, and seeks new members from both the public and private sectors.

The Standards Development Report, required by the ITE standards development process, was not available at the time this NTCIP Standards Bulletin was prepared. The SDR will be provided separately. The SDR provides a summary of the document development milestones, the working group members, and other standards that have been examined with regard to harmonization and duplication of content.

An electronic copy of the Recommended Standard is posted in the LIBRARY area of the NTCIP Website, at www.ntcip.org.

USER COMMENT DISPOSITION. All user comments regarding NTCIP 1103 v02 were recorded and the disposition of each comment is reported. In the report, official "User Comments" are designated with a "UC0000" number in the User Comment Number field.

The report from the User Comment database is posted in the NTCIP Web site's Forum for Standards. Click to the conference for NTCIP 1103. The user comment database report includes the disposition of the comments received on the document.

NOTES. The Joint Committee e-ballot accepted NTCIP 1103 version 02.14c as the Recommended Standard. Administrative delay, waiting for the SDR, plus additional technical editing (to make version v02.16b) all postponed the issue of this Standards Bulletin.

NAVIGATE THE PDF. To help users navigate the large document, the PDF file was authored with embedded links. The links can be found in the Table of Contents.

To use the links when browsing with Adobe® Reader®, select the Hand tool, and the Hand will change to the pointing finger when over a link. To return to the prior page after clicking a link, use Alt + Left Arrow to return to the Previous View. See the reader's View menu for more shortcuts.

BALLOT INSTRUCTIONS. The eligible voting representatives who receive this Bulletin are asked to follow the ballot instructions of their standards organization.

**ATTACH-
MENTS:** 1. NTCIP 1103 v02.16b, *Transportation Management
Protocols (TMP) – Version 02*, ~117 pages.

FROM: NTCIP Coordinator
NEMA
1300 North 17th Street, Suite 1752
Rosslyn, VA 22209-3806
fax (703) 841-3331
email <ntcip@nema.org>

CC: Via www.ntcip.org Web site and notices to NTCIP email reflectors.

###

A Recommended Standard of the Joint Committee on the NTCIP

NTCIP 1103 version v02.16

National Transportation Communications for ITS Protocol Transportation Management Protocols (TMP)

v02.16 March 2009

This is a draft pre-standard document, which is distributed for review and ballot purposes only. You may reproduce and distribute this document within your organization, but only for the purposes of and only to the extent necessary to facilitate review and ballot to AASHTO, ITE, or NEMA. Please ensure that all copies include this notice. This pre-standard contains recommended information that is subject to approval.

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

© 2009 by the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE), and the National Electrical Manufacturers Association (NEMA). All intellectual property rights, including, but not limited to, the rights of reproduction, translation and display are reserved under the laws of the United States of America, the Universal Copyright Convention, the Berne Convention, and the International and Pan American Copyright Conventions. Except as licensed or permitted, you may not copy these materials without prior written permission from AASHTO, ITE, or NEMA. Use of these materials does not give you any rights of ownership or claim of copyright in or to these materials.

Visit www.ntcip.org for other copyright information, for instructions to request reprints of excerpts, and to request reproduction that is not granted below.

PDF File License Agreement

To the extent that these materials are distributed by AASHTO / ITE / NEMA in the form of an Adobe® Portable Document Format (PDF) electronic data file (the "PDF file"), AASHTO / ITE / NEMA authorizes each registered PDF file user to view, download, copy, or print the PDF file available from the authorized Web site, subject to the terms and conditions of this license agreement:

- (a) you may download one copy of each PDF file for personal, noncommercial, and intraorganizational use only;
- (b) ownership of the PDF file is not transferred to you; you are licensed to use the PDF file;
- (c) you may make one more electronic copy of the PDF file, such as to a second hard drive or burn to a CD;
- (d) you agree not to copy, distribute, or transfer the PDF file from that media to any other electronic media or device;
- (e) you may print one paper copy of the PDF file;
- (f) you may make one paper reproduction of the printed copy;
- (g) any permitted copies of the PDF file must retain the copyright notice, and any other proprietary notices contained in the file;
- (h) the PDF file license does not include (1) resale of the PDF file or copies, (2) republishing the content in compendiums or anthologies, (3) publishing excerpts in commercial publications or works for hire, (4) editing or modification of the PDF file except those portions as permitted, (5) posting on network servers or distribution by electronic mail or from electronic storage devices, and (6) translation to other languages or conversion to other electronic formats;
- (i) other use of the PDF file and printed copy requires express, prior written consent.

Data Dictionary and MIB Distribution Permission

To the extent that these materials are distributed by AASHTO / ITE / NEMA in the form of a Data Dictionary ("DD") or Management Information Base ("MIB"), AASHTO / ITE / NEMA extend the following permission:

You may make and/or distribute unlimited copies, including derivative works, of the DD or MIB, including copies for commercial distribution, provided that:

- (i) each copy you make and/or distribute contains the citation "Derived from NTCIP 0000 [insert the standard number]. Used by permission of AASHTO / ITE / NEMA.";

- (ii) the copies or derivative works are not made part of the standards publications or works offered by other standards developing organizations or publishers or as works-for-hire not associated with commercial hardware or software products intended for field implementation;
- (iii) use of the DD or MIB is restricted in that the syntax fields may be modified only to reflect a more restrictive subrange or enumerated values;
- (iv) the description field may be modified but only to the extent that: (a) only those bit values or enumerated values that are supported are listed; and (b) the more restrictive subrange is expressed.

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 the user is unwilling to accept the foregoing restrictions, he or she should immediately return these materials.

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 and/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 standard are the trademarks of their respective owners.

Reference Status Table 1 (To be deleted on publication)

Reference Designation	Title	Type (Norm, Other)	Status	Pending
NTCIP 1201 v03	<i>Global (GO) Object Definitions</i>	Norm.	Publication anticipated. Needs 1201 v03. All refs in 1103 v02 are to 1201 v03.	Awaiting SDO ballot.
NTCIP 8004 v02	<i>Structure and Identification of Management Information (SMI)</i>	Other	Publication anticipated. Needs 8004 v02. Refs in 1103 v02 are to 8004 v02.	Awaiting SDO ballot
NTCIP 8005 v01	<i>Procedures for Creating Management Information Base Files and a Functional Area Data Dictionary</i>	Norm.	Publication anticipated. Needs 8005 v01. Ref (Annex A) in 1103 v02 is to 8005 v01.	Awaiting SDO ballot.

< Blank page to be deleted on publication. >

ACKNOWLEDGEMENTS

NTCIP 1103 v02 was prepared by the NTCIP Base Standards and Profiles 2 Working Group (BSP2 WG), which is a subdivision of the Joint Committee on the NTCIP. The Joint Committee on the NTCIP 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 NTCIP 1103 v02 was prepared, the following individuals were active members of the NTCIP BSP2 WG:

- Ralph Boaz
- Robert DeRoche (Chair)
- Bud Kent
- Jim Mahugh
- Greg Mizell
- Alex Mousadi
- Robert Rausch
- Peter Ragsdale
- Joerg 'Nu' Rosenbohm

Other individuals providing input to NTCIP 1103 v02 include:

- Blake Christie
- Keith Patton

In addition to the many volunteer efforts, recognition is also given to those organizations that supported the efforts of BSP2 WG by providing comments and, including:

- CalTrans
- Econolite
- Iteris, Inc.
- MasTec
- Ministry of Transportation Ontario
- Noblis (formerly known as Mitretek)
- Pillar Consulting
- Quixote Traffic Corporation
- Robert DeRoche Consulting
- Siemens
- Southwest Research Institute
- Texas Department of Transportation
- Telvent Farradyne (formerly known as PB Farradyne)
- Trevilon Corporation
- TransCore
- URS Corporation
- U.S. Department of Transportation, Research and Innovative Technology Administration
- Washington State Department of Transportation

FOREWORD

NTCIP 1103 v02 defines a composite application layer protocol for the management of transportation equipment. The composite protocol consists of three component protocols: the Internet-standard Simple Network Management Protocol (SNMP), the Simple Fixed Message Protocol (SFMP), and the Simple Transportation Management Protocol (STMP). The protocols are concerned with the procedures for exchanging information as well as the format in which the information is exchanged. When related to the ISO OSI Reference model, these protocols are concerned with the upper three layers (application, session and presentation layer). NTCIP 1103 v02 contains two normative and two informative annexes. NTCIP 1103 v02 uses only metric units.

NTCIP 1103 v02 is also an NTCIP Base Standard. NTCIP Base Standards provide formal definitions of the procedures and data formats for use within NTCIP systems. For more information about NTCIP standards, visit the NTCIP Web Site at 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 22209-3806
e-mail: ntcip@nema.org

A User Comment should be submitted in the following form:

Standard Publication number and version:
Page:
Section, Paragraph, or Clause:
Comment:
Editorial or Substantive?:
Suggested Alternative Language:

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

Approvals

TBD This standards publication version v02 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; ***TBD***
ITE—Software Standard; ***TBD***
NEMA—Standard; ***TBD***

History

From 1996 to 2004, parts of NTCIP 1103 v02, including the definition of STMP, were defined in NTCIP 1101:1996 / NEMA TS 3.2. However, to provide a more organized and modular set of standards, NTCIP 1101:1996 was separated into three distinct standards: NTCIP 1103 v02, which includes the definition of STMP; NTCIP 1102:2004, which defines the Octet Encoding Rules (OER); and NTCIP 8004 v02, which defines the Structure and Management of Transportation Information (SMI). These three standards completely replace NTCIP 1101:1996 / NEMA TS 3.2.

NTCIP 1103 v01.25, October 2004—Accepted as Recommended Standard by the Joint Committee on the NTCIP.

NTCIP 1103 v01.27, January 2009— In June 2005, pre-ballot comment on community name were addressed and v01.26 was sent for SDO balloting. In November 2005, NTCIP 1103 v01 was Jointly Approved. In December 2008, NTCIP 1103 v01.27 was edited and published.

NTCIP 1103 v02.10, May 2006—Developed and incorporated the NTCIP Trap mechanism and proposed as version 02 User Comment Draft. December 2006—Standards Bulletin B0117 sent v02.10b for review and comment. January 2008—Addressed user comments in v02.14.

NTCIP 1103 v02.16, March 2009—In December 2007, the NTCIP Joint Committee agreed to remove the trap mechanism from NTCIP 1103 v02, and to increment to a proposed new major version NTCIP 1103 v03 to define traps; therefore, the NTCIP 1103 v02.14 was revised as a proposed Recommended Standard, which in February 2008 was accepted by the NTCIP Joint Committee as a Recommended Standard. From 2008 to March 2009, NTCIP 1103 v02.16 was edited for SDO balloting and approval.

Compatibility of Versions

To distinguish NTCIP 1103 v02 (as published) from previous drafts, NTCIP 1103 v02 also includes NTCIP 1103 v02.16 on each page header. 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.

NTCIP 1103 v02 is designated, and should be cited as, NTCIP 1103 v02. Anyone using NTCIP 1103 v02 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 NTCIP 1103 v02 should also consult

NTCIP 8004 v02 for specific guidelines on compatibility.

STMP, as defined within NTCIP 1103 v02 is 100% consistent with the definition contained in NTCIP 1101:1996 / NEMA TS 3.2; however, the protocol has now been extended to support the new Simple Fixed Message Protocol. NTCIP 1103 v01 has also been expanded to address several other issues and to incorporate some of the protocol-specific data originally defined in NTCIP 1201:2005.

INTRODUCTION

NTCIP 1103 v02 provides the definition of a composite application layer protocol that consists of three component protocols. All three protocols provide the same base services, but are designed with different needs in mind. NTCIP 1103 v02 also defines a limited number of data elements necessary to manage these protocols. The data elements are defined according to the rules of NTCIP 8004 v02. Data elements in NTCIP 1103 v02 have been updated in accordance with and to reflect changes introduced in NTCIP 8004 v02.

NTCIP 1103 v02 defines requirements that are applicable to all NTCIP environments and also contains optional and conditional sections that are applicable to specific environments for which they are intended.

The following keywords apply to NTCIP 1103 v02: AASHTO, ITE, NEMA, NTCIP, protocol, message, transportation, simple, TMP, SNMP, SFMP, STMP, trap, traps.

In 1992, the NEMA 3-TS Transportation Management Systems and Associated Control Devices Section began the effort to develop the NTCIP. The Transportation Section's purpose was to respond to user needs to include standardized systems communication in the NEMA TS 2 standard, *Traffic Controller Assemblies*. 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 Intelligent Transportation Systems (ITS) network.

In September 1996, an agreement was reached among AASHTO, ITE, and NEMA to jointly develop, approve, and maintain NTCIP Standards. In late 1998, the Base Standards and Protocols Working Group was tasked with the effort to develop and maintain base standards for the NTCIP. In late 2003, the Joint Committee on the NTCIP merged the Base Standards and Protocols Working Group with the Profiles Working Group and the new group was designated the Base Standards, Protocols, and Profiles Working Group. The first meeting of the merged working group was held in January 2004.

TABLE OF CONTENTS

	Page
SECTION 1 GENERAL	1
1.1 Scope.....	1
1.2 References	1
1.2.1 Normative References.....	1
1.2.3 Contact Information	2
1.3 Terms.....	3
1.4 Acronyms.....	5
1.5 NTCIP 1103 v02 Layout	6
SECTION 2 TRANSPORTATION MANAGEMENT PROTOCOL	7
2.1 Composition of TMP.....	7
2.2 Simultaneous Processing.....	8
2.3 Protocol Identification Logic.....	8
SECTION 3 SIMPLE NETWORK MANAGEMENT PROTOCOL	11
3.1 Overview.....	11
3.2 Definition.....	11
3.2.1 General Rules	12
3.2.2 Set Operations on a Read-Only Variable.....	12
3.2.3 Extra Data Prohibition	12
3.2.4 Response Time	12
3.2.5 Trap Restrictions	12
SECTION 4 SIMPLE FIXED MESSAGE PROTOCOL (SFMP)	13
4.1 Overview.....	13
4.1.1 Data Identification.....	13
4.1.2 Packet Structure.....	13
4.1.3 Encoding.....	13
4.2 Definition.....	14
4.2.1 Rules	14
4.2.2 Elements of Procedure.....	14
4.2.3 SFMP Data Packet Structures	17
4.2.4 Definitions of Data Structure Fields.....	19
4.3 Examples	22
4.3.1 Get an Object Example	22
4.3.2 Get Block Object with Community Name Example.....	23
4.3.3 SFMP Set Example	24
4.3.4 SFMP Set Block Object Example.....	24
4.3.5 Get Error Example.....	25
4.3.6 Set Error Example	26
SECTION 5 SIMPLE TRANSPORTATION MANAGEMENT PROTOCOL (STMP)	28
5.1 Overview.....	28
5.1.1 Dynamic Objects	28
5.1.2 Other Truncations.....	30
5.2 Definition.....	30
5.2.1 Rules	30
5.2.2 Elements of Procedure.....	30
5.2.3 STMP Data Packet Structure	34
5.2.4 Dynamic Object Configuration	35
5.3 Examples	36

5.3.1	Configuring a Dynamic Object	37
5.3.2	Getting a Dynamic Object	37
5.3.3	Setting a Dynamic Object.....	38
SECTION 6 NTCIP TRAPS		39
SECTION 7 LOGICAL NAMES		40
SECTION 8 SECURITY		41
8.1	SNMP and SFMP Security	41
8.2	STMP Security.....	42
SECTION 9 CONFORMANCE STATEMENT		43
ANNEX A (NORMATIVE) TRANSPORTATION MANAGEMENT PROTOCOLS MANAGEMENT INFORMATION BASE.....		44
A.1.	TMP SNMP MIB Header	44
A.1.1.	Objects for SNMP.....	46
A.1.2.	Maximum SNMP Packet Size Parameter	46
A.2.	TMP SFMP MIB Header.....	47
A.2.1.	Number of Incoming SFMP Packets.....	49
A.2.2.	Number of Outgoing SFMP Packets.....	49
A.2.3.	Number of Incoming SFMP Packets with Bad Version Numbers.....	49
A.2.4.	Number of Incoming SFMP Packets with Bad Community Names	49
A.2.5.	Number of Incoming SFMP Packets with Bad Use of a Community Name.....	50
A.2.6.	Number of Incoming SFMP Packets with Parsing Errors	50
A.2.7.	Reserved	50
A.2.8.	Number of Incoming SFMP Packets indicating a Too Big Error	50
A.2.9.	Number of Incoming SFMP Packets indicating a No Such Name Error.....	50
A.2.10.	Number of Incoming SFMP Packets indicating a Bad Value Error	51
A.2.11.	Number of Incoming SFMP Packets indicating a Read-Only Error.....	51
A.2.12.	Number of Incoming SFMP Packets indicating a General Error	51
A.2.13.	Reserved	51
A.2.14.	Reserved	51
A.2.15.	Number of Incoming SFMP Get Requests	51
A.2.16.	Reserved	52
A.2.17.	Number of Incoming SFMP Set Requests	52
A.2.18.	Number of Incoming SFMP Get Responses	52
A.2.19.	Reserved	52
A.2.20.	Number of Outgoing SFMP Packets indicating a Too Big Error	52
A.2.21.	Number of Outgoing SFMP Packets indicating a No Such Name Error	53
A.2.22.	Number of Outgoing SFMP Packets indicating a Bad Value Error	53
A.2.23.	Number of Outgoing SFMP Packets indicating a Read-Only Error.....	53
A.2.24.	Number of Outgoing SFMP Packets indicating a General Error	53
A.2.25.	Number of Outgoing SFMP Get Requests	54
A.2.26.	Reserved	54
A.2.27.	Number of Outgoing SFMP Set Requests	54
A.2.28.	Number of Outgoing SFMP Get Responses	54
A.2.29.	Number of Outgoing SFMP Trap Messages	54
A.2.30.	Reserved	55
A.2.31.	Number of Incoming SFMP Set Requests – No Replies.....	55
A.2.32.	Number of Incoming SFMP Set Responses.....	55
A.2.33.	Number of Incoming SFMP Error Responses	55
A.2.34.	Number of Outgoing SFMP Set Requests – No Replies.....	55
A.2.35.	Number of Outgoing SFMP Set Responses.....	56
A.2.36.	Number of Outgoing SFMP Error Responses.....	56

A.3.	TMP STMP MIB Header	56
A.3.1.	Type Definitions	58
A.3.2.	Objects for STMP	58
A.3.3.	Maximum Dynamic Object Table Entries	58
A.3.4.	Dynamic Object Definition Table	59
A.3.5.	Dynamic Object Data	60
A.3.6.	Dynamic Object Configuration	61
A.4.	TMP STMP-Statistics MIB Header	62
A.4.1.	STMP Statistics	64
A.5.	TMP STMP Configuration MIB Header	71
A.5.1.	STMP Configuration	73
A.6.	TMP LogicalNames MIB Header	74
A.6.1.	Maximum Logical Name Translations	76
A.6.2.	Logical Name Translation Table	76
A.7.	TMP REPORT MIB Header	78
A.7.1.	Report Parameter Node	80
A.7.2.	Maximum Event Classes Parameter	80
A.7.3.	Event Class Table	80
A.7.4.	Maximum Event Log Configurations Parameter	83
A.7.5.	Event Log Configuration Table	83
A.7.6.	Maximum Event Log Size Parameter	88
A.7.7.	Event Log Table	88
A.7.8.	Total Event Log Counter Parameter	90
A.8.	TMP Security MIB Header	91
A.8.1.	Community Name Administrator Parameter	93
A.8.2.	Maximum Community Names Parameter	93
A.8.3.	Community Names Table	93
A.9.	TMP Trap MIB Header	95
ANNEX B DEPRECATED OBJECTS (NORMATIVE)	96	
B.1.	Deprecated Type Definitions	96
B.2.	Deprecated Object Types	96
B.2.1.	Dynamic Object Definition	96
B.2.2.	Dynamic Object Data	96
ANNEX C (INFORMATIVE) AN EXPLANATION OF RELATIVE OBJECT IDENTIFIERS	101	
ANNEX D (INFORMATIVE) ENTRY STATUS TYPE	102	

SECTION 1 GENERAL

1.1 SCOPE

The Transportation Management Protocol (TMP) specifies an NTCIP Application Layer service. TMP defines a set of rules and procedures for exchanging transportation management information between transportation management applications and transportation equipment such that they interoperate with each other. The transportation management information that is exchanged using TMP is defined elsewhere according to the rules defined in NTCIP 8004 v02. Messages conforming to TMP may be exchanged using any appropriate transport mechanism.

TMP was carefully designed to provide 100% interoperability with the Internet-standard Simple Network Management Protocol (SNMP), but extends SNMP structure to meet the needs of the transportation environment. Analysis of the transportation environment has revealed the need for protocol *simplicity*, *flexibility*, and *minimal data packet size*; however, in many cases these three requirements are at odds.

After a careful review of existing protocols, it was decided to pursue the development of TMP, which combines the capabilities of three component protocols, one of which is. Each component protocol has been designed to maximize two of the three requirements at the expense of the third requirement.

1.2 REFERENCES

Normative references contain provisions that, through reference in this text, constitute provisions of NTCIP 1103 v02. Other references in NTCIP 1103 v02 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.

1.2.1 Normative References

IAB Std. 15/RFC 1157	Simple Network Management Protocol
ISO/IEC 8824-1:2002	Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation
AASHTO / ITE / NEMA NTCIP 1102:2004	<i>Octet Encoding Rules (OER) Base Protocol</i> published October 2005
AASHTO / ITE / NEMA NTCIP 8005 v01	<i>Procedures for Creating Management Information Base Files and a Functional Area Data Dictionary</i> (publication anticipated)

1.2.2 Other References

AASHTO / ITE / NEMA NTCIP 1201 v03	<i>Global (GO) Object Definitions</i> (publication anticipated)
---------------------------------------	--

AASHTO / ITE / NEMA
NTCIP 2301 v02 *Simple Transportation Management Framework
Application Profile—Version 02*
published March 2008

AASHTO / ITE / NEMA
NTCIP 8004 v02 *Structure and Identification of Management Information (SMI)*
(publication anticipated)

For approved revisions, contact:

NTCIP Coordinator
NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION
1300 North 17th Street, Suite 1752
Rosslyn, VA 22209-3801

For proposed revisions, which are under discussion by the relevant NTCIP Working Group, and revisions recommended by the NTCIP Joint Committee are available on the World Wide Web at <http://www.ntcip.org>.

1.2.3 Contact Information

1.2.3.1 NTCIP Standards

For revision information on this NTCIP standard, contact:

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

For draft revisions to this NTCIP standard, and recommended revisions of the NTCIP Joint Committee, visit www.ntcip.org.

1.2.3.2 IAB Documents

For Internet Advisory Board (IAB) documents, contact:

Internet Architecture Board (IAB)
www.rfc-editor.org
www.rfc-editor.org/repositories.html

1.2.3.3 ISO/IEC Standards

Members of ISO maintain registers of currently valid ISO/IEC International Standards. For the USA, the member of ISO is the American National Standards Institute (ANSI), which may be contacted as follows:

ANSI
11 West 42nd Street, 13th Floor
New York, NY 10036
(212) 642-4900
<http://global.ihf.com>

1.3 TERMS

For the purposes of this standard, the following terms and definitions apply. For terms not defined in this clause, English words are used in accordance with their definitions in the latest edition of *Webster's New Collegiate Dictionary*. Electrical and electronic terms not defined in this clause or in *Webster's New Collegiate Dictionary* are used in accordance with their definitions in IEEE Std 100-19962.

agent	The entity that receives commands and transmits responses to the received commands.
Block Object	An SNMP object with syntax of OerString. An OerString is defined to be an ASN.1 OCTET STRING that contains a Data Frame or Data Element that has already been serialized according to the rules of OER. This allows a structure of data elements to be treated as a single Object. NOTE—See ISO/IEC 8824-1:2002 for ASN.1 information.
compatible	The ability of two or more systems or components to exchange information (IEEE Std. 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology).
data	Information before it is interpreted.
datagram	A self-contained unit of data transmitted independently of other datagrams.
Data Packet	A serialized Message. In other words, a data packet is the byte stream representing a logical structure which is termed a message.
deprecated	This term is defined in NTCIP 8004 v02.
Dynamic Object	A simple grouping of data elements (i.e., the equivalent of an ASN.1 Sequence) defined at run-time for the purpose of referring to the group with a single short identifier and encoded according to the rules of OER, and thereby providing a mechanism to efficiently exchange the group of data within a protocol. This is conceptually similar to a Data Frame, except that the contents of Dynamic Objects are defined at run-time. Dynamic Objects are only accessible using the STMP protocol, and therefore these are not true objects in the SNMP sense.
file	A grouping of individual or block database objects into a single sequence of bytes that can be referred to by file operations. It exists nominally in a directory and can have a path associated with it.
Fixed Message	A fixed message is any message (i.e., data packet) as used in the Simple Fixed Message Protocol (SFMP). The term fixed emphasizes that the message only contains one Object, versus SNMP with any number of objects, and that the structure of the Object is not dynamic, as is the case in a Dynamic Object.
Internet	A large collection of connected networks, primarily in the United States, running the Internet suite of protocols. Sometimes referred to as the <i>DARPA Internet</i> , <i>NSF/DARPA</i> , <i>Internet</i> , or the <i>Federal Research Internet</i> .
Internet Protocol	The network protocol offering a connection-less mode network service in the Internet suite of protocols.

Interoperate	The ability of two or more systems or components to exchange information and use the information that has been exchanged (IEEE Std. 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology).
IP address	A 32-bit quantity used to represent a point of attachment in an internet.
management information base	A structured collection or database of related managed objects defined using Abstract Syntax Notation One (ASN.1).
manager	The entity that sends commands to entities and processes their responses.
Meta	A word denoting a description that is one level of abstraction above the entity being described.
MIB View	A set of objects within a MIB. Different MIB views may be defined for each community name. A set of objects do not need to be confined to a single node on the ISO global naming tree.
network	A collection of subnetworks connected by intermediate systems and populated by end systems.
network layer	That portion of an OSI system responsible for data transfer across the network, independent of both the media comprising the underlying subnetworks and the topology of those subnetworks.
Object	A specific instance of an object type that may be managed by SNMP. Thus, an object may be either a data element or a data frame
OBJECT IDENTIFIER	A unique name (identifier) that is associated with each type of object in a Management Information Base. This is a defined ASN.1 type.
object type	A data structure used to describe the attribute or properties of an object or a group of objects.
Object Type	<p>A classification of one or more objects that share a common definition and representational form. Synonymous with the object-oriented term "class."</p> <p>Managed devices will frequently contain tables of information; if each row is considered a record, then each column would be an object type and each cell would be an object (i.e., an instance of the object type as defined by the column heading). For objects that are not contained in tables, there is a one-to-one relationship between object type and object; the only difference is that the object type is the abstract concept whereas the object is the precise instance.</p> <p>NOTE—The definition of each object type includes a name and a syntax; the encoding of the object is defined by applying the protocol-specific rules to the logical syntax as defined in ASN.1.</p> <p>SNMP requires the contents of an object to be a simple ASN.1 Type; however, NTCIP has defined the concept of a Block Object, which is a serialized Data Frame, to circumvent this limitation of SNMP and to improve encoding efficiencies. Thus, within NTCIP, the term object may either refer to a data element or a data frame.</p>

obsolete	This term is defined in NTCIP 8004 v02.
protocol	A specific set of rules, procedures, and conventions defining the format and timing of data transmissions between devices that must be accepted and used to understand each other.
Referenced Object	An object instance that is supported by a device and that may be referenced from a dynamic object via the use of the dynObjVariables field.
Response Time	The time between the receipt of the last byte of the request and the start of the transmission of the first byte of the response.
Simple Network Management Protocol	A communications protocol developed by the Internet Engineering Task Force, used for configuration and monitoring of network devices.
Simple Transportation Management Framework	Describes the organization of the information within devices and the methods of retrieving or modifying any information within the device. STMF also explains how to generate and utilize computer readable information organization descriptions.
subnet/subnetwork	A physical network within a network on which all devices share the same physical media.
Trap	A particular form of a data packet that will provide pre-defined (dynamically configured) information to requests from a management station based on “events” within the device. These particular data packets are initialized and, potentially, transmitted (depending on the transmission media) without a request from a management station.
Variable Binding	A structure used within SNMP to couple the identifier for a piece of data with its value.
Variable Binding List	A sequence of variable bindings.

1.4 ACRONYMS

AID	Application Identifier
IETF	Internet Engineering Task Force
ITS	Intelligent Transportation Systems
NTCIP	National Transportation Communications for ITS Protocol
SFMP	Simple Fixed Message Protocol
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
STMP	Simple Transportation Management Protocol
T2	Transportation Transport Profile per NTCIP 2201
TMIB	Transportation Management Information Base
TMP	Transportation Management Protocol

1.5 NTCIP 1103 V02 LAYOUT

The remainder of NTCIP 1103 v02 is divided into the following sections:

Section 2 describes the overall structure of the Transportation Management Protocols (TMPs) and how the three component protocols merge to form a single identifiable protocol.

Section 3 provides an overview of how the Internet standard Simple Network Management Protocol (SNMP) works.

Section 4 describes how the Simple Fixed Message Protocol (SFMP) enhances the concepts of SNMP to provide for a more compact encoding of data while still providing a truly simple design with the loss of some flexibility.

Section 5 describes how the Simple Transportation Management Protocol (STMP) provides for a flexible compact encoding of data at the expense of some loss of simplicity.

Section 6 is a placeholder where a subsequent version of NTCIP 1103 v02 is expected to describe how TMP handles exception- or event-driven reporting (i.e., to allow an agent to send unsolicited information to the management station within a semi-controlled environment). This is referred to as traps in SNMP terminology.

Section 7 discusses how logical names are defined.

Section 8 describes security issues related to TMP.

Section 9 defines conformance to NTCIP 1103 v02.

Annex A is normative and provides the formal definition of the TMPs Management Information Bases (MIBs), which in their entirety replace the Transportation Management Information Base (TMIB) previously defined in NEMA TS 3.2/NTCIP 1101:1996. Annex A has also been enhanced from NTCIP 1103 v01 to conform to NTCIP 8004 v02.

Annex B is normative and provides information about deprecated objects.

Annex C is informative and provides information about the new ASN.1 Syntax of Relative OID.

Annex D is informative and provides a description of the EntryStatus type that NTCIP standards defined previously, but has now been deprecated.

SECTION 2 TRANSPORTATION MANAGEMENT PROTOCOL

TMP is a combination of three distinct protocols all providing nearly identical services, but designed to meet different data exchange and processing requirements. The three component protocols are:

- a) Simple Network Management Protocol (SNMP), version 1, as defined in IAB 15—RFC 1157 and according to the rules defined in Section 3.
- b) Simple Fixed Message Protocol (SFMP), as defined in Section 4.
- c) Simple Transportation Management Protocol (STMP), as defined in Section 5.

TMP was carefully designed to provide 100% interoperability with the Internet-standard SNMP, but extends this protocol structure to provide for additional requirements of the transportation environment. While the SNMP met the flexibility and simplicity requirements of the transportation industry, it produces a very verbose encoding that does not meet the functional requirements of existing communication infrastructures. As a result, NTCIP produced two additional protocols that produce a much more compact encoding, sacrificing either simplicity or flexibility. The relationships among these three protocols are described in Figure 1.

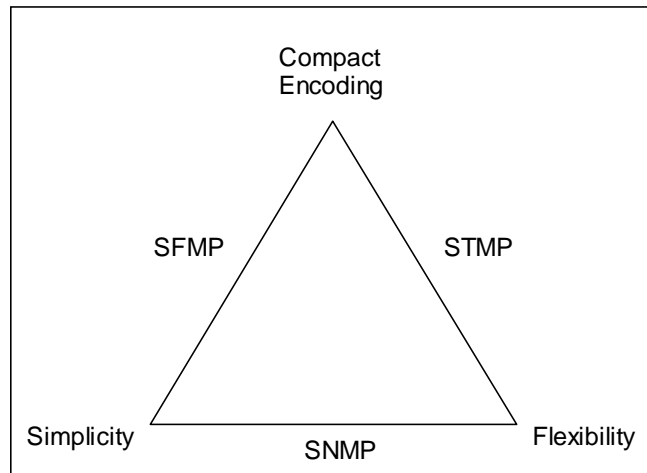


Figure 1 Requirements Relationships

The information exchanged by all three protocols shall be defined according to the rules set forth in NTCIP 8004 v02, which is fully compatible with RFC 1155 as used by SNMP.

2.1 COMPOSITION OF TMP

TMP refers to the data construct that allows all three protocols to coexist while using the same protocol identifier. This was achieved due to the fact that all SNMP messages start with an initial byte of 0x30 (i.e., SNMP uses Basic Encoding Rules and all SNMP messages are defined as a SEQUENCE of data). Thus, the TMP construct has been designed to use the value of this first byte to identify which protocol is being referenced. The value of 0x30 identifies an SNMP message. Both SFMP and STMP messages use the high-order nibble (i.e., highest order four bits) of the first byte to determine the type of message (e.g., get request, set request, etc.). The low-order nibble is then used to identify whether the message is a fixed

message (i.e., SFMP, indicated by the value of zero), or one of the 13 dynamic objects (i.e., STMP, indicated by a value of 1 through 13). The specific mapping of the first byte value is defined in Table 1.

Table 1 TMP Protocol Mapping

Protocol	Value
SNMP—All SNMP Messages (including traps)	0x30
SFMP—Get Request	0x80
SFMP—Set Request	0x90
SFMP—Set No Reply	0xA0
SFMP—Get Response	0xC0
SFMP—Set Response	0xD0
SFMP—Error Response	0xE0
SFMP—Reserved	0xF0
STMP—Get Request (for 13 dynamic objects)	0x81—0x8D
STMP—Set Request (for 13 dynamic objects)	0x91—0x9D
STMP—Set No Reply (for 13 dynamic objects)	0xA1—0xAD
STMP—Get Next (for 13 dynamic objects)	0xB1—0xBD
STMP—Get Response (for 13 dynamic objects)	0xC1—0xCD
STMP—Set Response (for 13 dynamic objects)	0xD1—0xDD
STMP—Error Response (for 13 dynamic objects)	0xE1—0xED
Reserved for compatibility with TP-T2	0x31
Reserved for compatibility with TP-T2	0x41
Reserved	0x00—0x29
Reserved	0x32—0x40
Reserved	0x42—0x7F
Reserved	0x8E—0x8F
Reserved	0x9E—0x9F
Reserved	0xAE—0xB0
Reserved	0xBE—0xBF
Reserved	0xCE—0xCF
Reserved	0xDE—0xDF
Reserved	0xEE—0xEF
Reserved	0xF1—0xFF

When decoding, TMP transmits the entire data stream, including the first byte, to the correct component protocol. When encoding, TMP simply transmits the entire data stream to the lower layer without changing the encoding from the component protocol.

2.2 SIMULTANEOUS PROCESSING

A management station takes into account the variable binding list processing nature of TMP. In TMP, all objects contained in a single set-request data packet appear to be set to their new values simultaneously. Therefore, a management station shall not combine a state change request with a request to set an instance value associated with that state change. If such an operation is attempted, the operation may not be correctly processed. For example, using a single set command to change both the status and contents of a dynamic object can have unpredictable results.

2.3 PROTOCOL IDENTIFICATION LOGIC

The structure of the mappings defined in Table 1 is based on the following general principles as depicted in Figure 2:

- a) If the first byte is 0x30, determine whether SNMP is supported in the device.

- 1) If SNMP is supported, the message is an SNMP message.
- 2) If SNMP is not supported, discard the data packet.
- b) If the first byte is not 0x30, determine whether the value of the first byte is greater than 0x80.
 - 1) If the first byte is smaller than 0x80, discard the data packet.
 - 2) If the first byte is larger than 0x80, proceed to step c.
- c) If the high order bit of the first byte is set, i.e., the value of the first byte is greater than or equal to 0x80, the message is either STMP or SFMP as defined by the following.
 - 1) If the low order nibble (i.e., the four low order bits) of the first byte is equal to 0, the message is SFMP.
 - i) If SFMP is supported, proceed accordingly.
 - ii) If SFMP is not supported, discard the data packet.
 - 2) If the low order nibble of the first byte is between 1 and 13 (0xD) inclusive and the high order nibble of the first byte is not 15 (0xF), the message is STMP.
 - i) If low order nibble of the first byte is greater than 13 (0xD), discard the data packet.
 - ii) If the high order nibble of the first byte is 15 (0xF), discard the data packet.
 - iii) If STMP is supported, proceed accordingly.
 - iv) If STMP is not supported, discard the data packet.

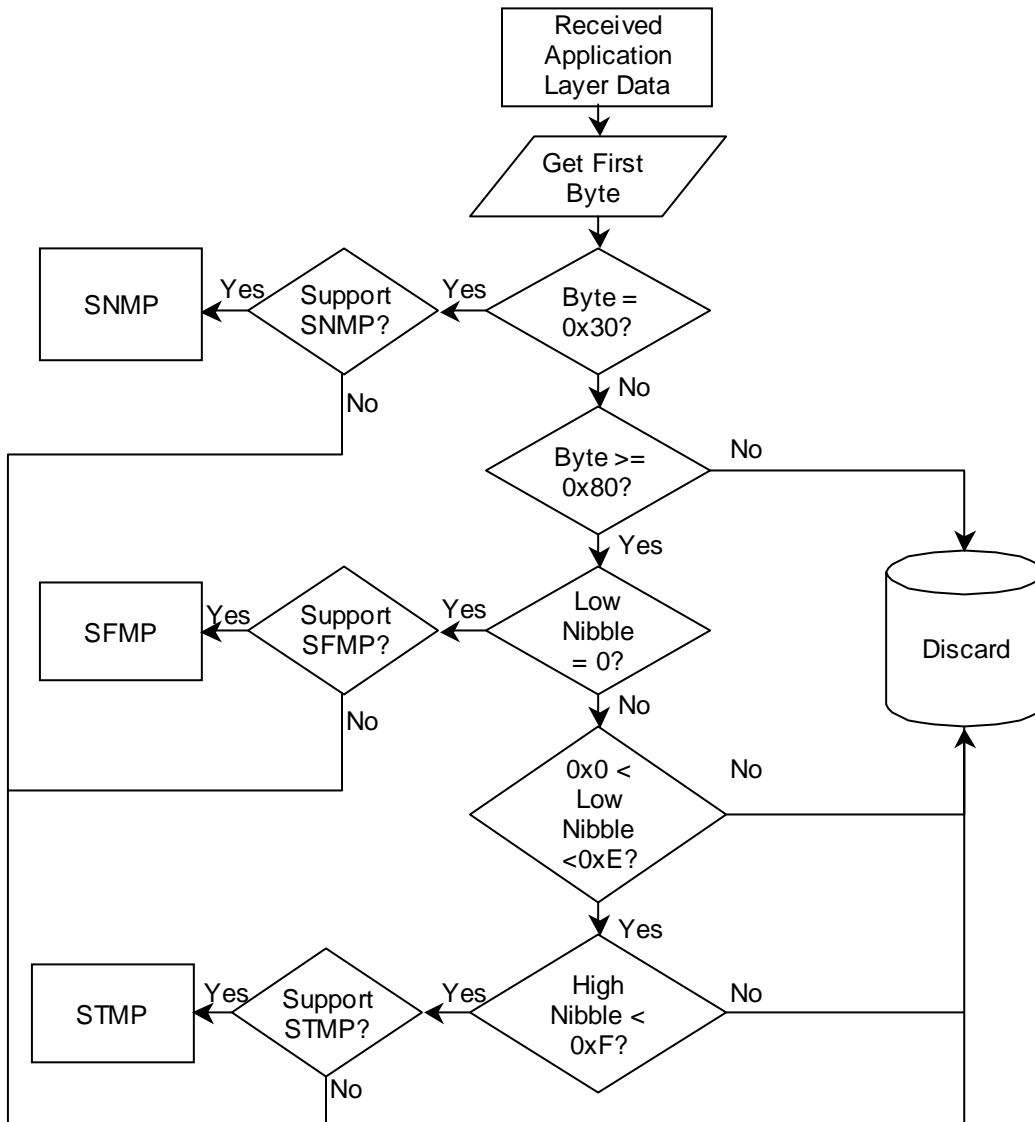


Figure 2 Process to Determine Component Protocol

NOTE—This value should not be confused with the AID of T2. Specifically, the values of 0x31 and 0x41 should never be received at the Application Layer of TMP; these values may be used by the T2 Transport Profile to identify an SNMP Trap or a header using port numbers, but this information is stripped by T2 prior to delivering the Application Layer Data to this process.

SECTION 3 SIMPLE NETWORK MANAGEMENT PROTOCOL

SNMP is a major standard developed by the Internet Engineering Task Force (IETF). NTCIP selected SNMP for use in the ITS industry due to its wide use within the Internet community, the flexibility SNMP provides management stations to define their own message content, and the simplicity of SNMP. While there were concerns about the encoding overhead that SNMP imposed on data communications, it was decided that SNMP provided a core set of functionality and that companion protocols could be developed to circumvent the overhead issues. While SNMP is only one of the three protocols employed by TMP, all three protocols follow the same basic get-set paradigm employed by SNMP. This paradigm is described briefly below and numerous textbooks describe SNMP in much greater detail.

3.1 OVERVIEW

SNMP uses a get-set paradigm to exchange individual pieces of data. Each piece of data stored within a device and that is accessible via the SNMP protocol is called an *object*. Each object consists of two parts: the *object type* and the *instance*. Some object types may only occur once within a device, these are called *scalar* objects and are assigned the instance of zero (0). Other objects, i.e., those that exist in a table, may have multiple instances; these objects are called *columnar* objects and their instance is determined based on which index (i.e., row of the table) they are associated with. The first row of a table typically has an index of 1.

Each object type stored within a device is formally defined in a computer readable file called a Management Information Base (MIB). The MIB associates each object type with a precise syntax, a definition, and an Object Identifier, which is generally about 15 bytes long. An object instance is identified by appending the instance number to this base Object Identifier. Thus, each piece of data within the device has a unique number associated with it.

An SNMP management station exchanges data by sending each subject object identifier along with the get or set request. A single SNMP message may, and typically does, include the request for multiple objects simultaneously. Thus, any one SNMP data packet is likely to contain several of these large object identifiers. Likewise, the response also returns the object identifiers along with the data, even for responses to set operations.

SNMP also allows an agent to transmit unsolicited information called a *trap*. Data transmitted along with the trap notification also includes Object Identifiers.

NOTE—NTCIP 1103 v02 does not define rules regarding traps in NTCIP environments; however, a future version of NTCIP 1103 v02 is expected to do so.

This is a reasonable approach to exchanging data when the data exchanges are infrequent and may change content from one request to another, which is typical of the Internet. However, within the ITS environment, the majority of the communications volume between a management application and an agent consists of a small number of messages that are repeatedly exchanged. In many cases, these exchanges occur frequently over dedicated channels. Thus, a significant reduction in the size of these frequently repeated messages could significantly reduce the size of the communications channel required for a link.

3.2 DEFINITION

SNMP shall be in accordance with SNMP version 1 as defined in IAB Std. 15/RFC 1157, and the requirements in Section 3.2, Section 6, and Section 7.

3.2.1 General Rules

A management station must take into account the variable binding list processing nature of TMP. In TMP, all objects contained in a single set-request data packet appear to be set to their new values simultaneously. Therefore, a management station shall not combine a state change request with a request to set an instance value associated with that state change. If such an operation is attempted, the operation may not be correctly processed.

3.2.2 Set Operations on a Read-Only Variable

Upon an agent receiving a SetRequest-PDU for a read-only object, the first condition listed under Clause 4.1.5 of IAB Std. 15/RFC 1157 shall apply.

NOTE—The definition of ErrorStatus in Clause 4.1.1 of IAB Std. 15/RFC 1157 and the wording of Clause 4.1.5 of IAB Std. 15/RFC 1157 have caused some confusion as to how a device should handle a SetRequest-PDU for a read-only object. For the purposes of conformance with NTCIP 1103 v02, the most literal meaning of the IAB Std. 15/RFC 1157 text applies. Thus, upon receipt of a SetRequest-PDU for a read-only object, an agent will respond with a noSuchName error. This interpretation is consistent with that of the broader Internet community as indicated in the clarifications provided by RFC 1213, which deprecates the snmpOutReadOnlys object and indicates that transmitting readOnly error is in fact a protocol error (within the definition of snmplnReadOnlys). The other TMP protocols have been designed to return the read-only error value.

A management station shall accept all of the following error codes as valid responses to an attempt to set a read-only object: noSuchName, readOnly, and genErr.

NOTE—This requirement ensures that a management station is able to work properly with any NTCIP version 1 device that may have interpreted IAB Std. 15/RFC 1157 differently.

3.2.3 Extra Data Prohibition

An agent receiving a get-request or get-next-request containing anything other than a NULL in a variable binding value field shall silently drop the data packet.

NOTE—Clause 4.1.1 of IAB Std. 15/RFC 1157 requires that any value in this field be ignored. NTCIP 1103 v02 further clarifies the intent of the IAB Std. 15/RFC 1157 clause by completely prohibiting the insertion of any data; thereby minimizing the size of the data packet.

3.2.4 Response Time

The SNMP agent shall process all requests in accordance with Section 3, including processing the request sufficiently to generate the transmission of the appropriate response (assuming that the SNMP agent has permission to transmit) within the maximum Response Time. If the specification does not indicate the maximum Response Time, the maximum Response Time shall be 100 milliseconds plus one millisecond for each byte in the response variable-bindings field.

3.2.5 Trap Restrictions

Support for processing SNMP traps is optional for the management station.

NOTE—If a management station wishes to manage the generic traps from its NTCIP devices, these are handled within the transportation management application rather than a network management application.

NOTE—NTCIP 1103 v02 does not define rules regarding traps in NTCIP environments; however, a future version of NTCIP 1103 v02 is expected to do so.

SECTION 4 SIMPLE FIXED MESSAGE PROTOCOL (SFMP)

4.1 OVERVIEW

SFMP can be viewed as a simplified, more compact version of SNMP. A careful analysis of SNMP reveals that the size and complexity of the data packets can be reduced by:

- a) Identifying the data contents within a data packet by using a single identifier that references a group of data elements rather than using a separate identifier in association with each data element in the data packet,
- b) Defining a data packet structure that only includes the information that is required for a given message type (e.g., a set response does not need to echo the values), and
- c) Using a set of encoding rules that are more efficient than the Basic Encoding Rules as used by SNMP.

4.1.1 Data Identification

SFMP decreases the size of the overhead consumed by data identification in two ways. First, SFMP is designed with the assumption that it will exchange a single composite object, i.e., an object that consists of a defined sequence of other objects. This approach decreases the overhead by allowing the use of a single object identifier rather than a separate identifier for each component object. Second, the design of SFMP incorporates the concept that all of the composite objects will be located under the NEMA node of the ISO tree. As such, it includes an encoding mechanism to shorten the object identifiers for objects under this node. Further, the complexity of the SFMP protocol is reduced due to the fact that an agent is not required to handle a get or set command with any combination of data in any order, it only is required to support one object at a time and if the object is a block object, the ordering of data is always fixed. This allows less powerful and less complex devices to support NTCIP.

4.1.2 Packet Structure

In SNMP, all of the data packets use a very similar data structure. While this provides for some advantages in code reusability, it also results in extra information being sent in many of the SNMP data packets. Due to the need to minimize the overhead for the most frequently exchanged messages and to minimize the processing requirements to decode these extra bytes; NTCIP developed the SFMP Data Packet Structure to more efficiently exchange the fixed messages while still providing necessary security functions. This packet structure is defined in detail in Section 4.2.

4.1.3 Encoding

SNMP encodes all of its information according to the ASN.1 Basic Encoding Rules (BER). BER uses a three *tuple* to encode data for transmission. The first element of the tuple, the type, specifies what type of data follows. The second element of the tuple, the length, specifies how many octets the data occupies. The third and final element of the tuple, the value, is the actual data being transmitted. This encoding is sometimes referred to as *TLV* encoding, which stands for 'type', 'length', 'value'. This provides a very flexible method of encoding information for transmission; however, if both sides have already agreed on a specific data structure, it includes unnecessary overhead by including the 'type' field and the 'length' field for fixed length data.

As a result, NTCIP has defined a separate set of ASN.1 encoding rules, known as Octet Encoding Rules (OER), as defined in NTCIP 1102:2004. OER eliminates the type field completely, and it eliminates the length field under those conditions where length is known. Given that most of the object definitions (data

elements) defined by NTCIP consist of INTEGERS in the range of 0 to 255, OER is able to significantly reduce the size of many NTCIP data packets.

4.2 DEFINITION

All SFMP interface implementations, on a device, shall be declared to be either a management station or an agent. A device can have multiple interfaces, each interface being declared as either a management station or an agent.

Management stations may transmit SFMP-GetRequest-PDU's, SFMP-SetRequest-PDU's, and SFMP-SetRequest-NoReply-PDU's as necessary in order to manage agents. Management stations shall be able to receive and process SFMP-GetResponse-PDU's, SFMP-SetResponse-PDU's, SFMP-ErrorResponse-PDU's per the rules defined within this clause.

Agents shall be able to receive and process SFMP-GetRequest-PDU's, SFMP-SetRequest-PDU's, and SFMP-SetRequest-NoReply-PDU's per the rules defined within this clause, including the ability to transmit SFMP-GetResponse-PDU's, SFMP-SetResponse-PDU's, and SFMP-ErrorResponse-PDU's as required.

The transmission of SFMP data packets shall be governed by the rules and procedures defined in Sections 4.2.1 and 4.2.2 and shall conform to the structures defined in Section 4.2.3.

4.2.1 Rules

Similar to SNMP, SFMP models all device functions as alterations (i.e., SETs) or inspections (i.e., GETs) of variables (i.e., objects). This strategy avoids the complexities of defining a different message type as a part of the protocol definition for each command desired.

Communication among protocol entities is accomplished by the exchange of protocol messages, each of which may be entirely and independently represented within a single datagram. An implementation of this protocol shall accept valid messages whose length does not exceed 484 octets. However, it is recommended that implementations support larger messages whenever feasible.

A management station must take into account the variable binding list processing nature of TMP. In TMP, all objects contained in a single set-request data packet appear to be set to their new values simultaneously. Therefore, a management station shall not combine a state change request with a request to set an instance value associated with that state change. If such an operation is attempted, the operation may not be correctly processed.

4.2.2 Elements of Procedure

Section 4.2.2 describes the actions of a protocol entity implementing SFMP; however, Section 4.2.2 is not intended to constrain the internal architecture of any conformant implementation.

An SFMP protocol entity transmits an SFMP message by passing the serialized message to a transport service capable of sending the message to the peer protocol entity.

An SFMP protocol entity receives an SFMP message as follows:

- a) It performs a rudimentary parse of the incoming datagram to build an ASN.1 object corresponding to the referenced Objects. If the parse fails, it discards the datagram and performs no further actions.
- b) It then verifies the version number of the SFMP message. If there is a mismatch, it discards the datagram and performs no further actions.
- c) It then authenticates the community name. If the authentication fails, the protocol records an authentication failure event by incrementing the *sfmp-inBadCommunityNames* object (see Annex A.4) and the *agentHealth-authenticationFailures* object, discards the datagram, and performs no further

actions.

- d) It then processes the message according to the rules defined in Sections 4.2.2.2 and 4.2.2.3 using the identified community.

4.2.2.1 Initiating a Request

An SFMP management station may, at any time, initiate a get or set request by generating an SFMP-Data-Packet containing an SFMP-GetRequest-PDU, an SFMP-SetRequest-PDU, or an SFMP-SetRequest-NoReply-PDU. The conditions that may result in the management station initiating such a request are the subject of the end-application functionality and are beyond the scope of NTCIP 1103 v02.

An SFMP agent shall not issue an SFMP-Data-Packet containing an SFMP-GetRequest-PDU, an SFMP-SetRequest-PDU, or an SFMP-SetRequest-NoReply-PDU. This restriction shall not preclude a single device from acting as both a management station and an agent.

4.2.2.2 Processing a Request

An SFMP management station shall silently drop (discard and perform no further action on the request) any SFMP-GetRequest-PDU, SFMP-SetRequest-PDU, or SFMP-SetRequest-NoReply-PDU.

The SFMP agent shall process all requests in accordance with the following subclauses, including processing the request sufficiently to generate the transmission of the appropriate response (assuming that the SFMP agent has permission to transmit) within the maximum Response Time. If the specification does not indicate the maximum Response Time, the maximum Response Time shall be 100 milliseconds plus one millisecond for each byte in the response SFMP-PDU data field.

4.2.2.2.1 Processing an SFMP Get Request

Upon receipt of an SFMP-GetRequest-PDU, an SFMP agent shall respond according to the following rules, in order:

- a) If the SFMP-GetRequest-PDU contains an information field, the agent shall silently drop the data packet.
- b) If the value of the message-OID field does not exactly correspond to an object available for get operations in the relevant MIB view, the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-ErrorResponse-PDU. The error-status field shall indicate noSuchName and the error-index field shall indicate zero (0).
- c) If the value of the message-OID field references an object that is of a non-accessible aggregate type (as defined by RFC 1155, a normative reference of NTCIP 8004 v02), the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-ErrorResponse-PDU. The error-status field shall indicate noSuchName and the error-index field shall indicate zero (0).
- d) If the size of the SFMP-GetResponse-PDU would exceed a local limitation, the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-ErrorResponse-PDU. The error-status fields shall indicate tooBig and the error-index field shall indicate zero (0).
- e) If the value of the object referenced by the message-OID field cannot be retrieved for reasons not covered by any of the foregoing rules, the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-ErrorResponse-PDU. The error-status field shall indicate genErr and the error-index field shall either indicate zero (0) or shall indicate the element within the structure that is preventing the operation.
- f) If none of the foregoing rules apply, the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-GetResponse-PDU such that the request number field shall be that used in the SFMP-GetRequest-PDU and the data field shall contain the requested information.

4.2.2.2 Processing an SFMP Set Request

Upon receipt of an SFMP-SetRequest-PDU, an SFMP agent shall respond according to the following rules, in order:

- a) If the SFMP-SetRequest-PDU does not contain a data field, the agent shall silently drop the data packet.
- b) If the object referenced by the value of the message-OID field is only available for get operations within the relevant MIB view, the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-ErrorResponse-PDU. The error-status field shall indicate readOnly and the error-index field shall indicate zero (0).
- c) If the value of the message-OID field does not exactly correspond to an object available for set operations in the relevant MIB view, the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-ErrorResponse-PDU. The error-status field shall indicate noSuchName and the error-index field shall indicate zero (0).
- d) If the contents of the data field cannot be parsed to fit the SYNTAX defined for the referenced object, the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-ErrorResponse-PDU. The error-status field shall indicate badValue and the error-index field shall indicate the field number at which the parsing first failed.
- e) If the value of the object referenced by the message-OID field cannot be altered for reasons not covered by any of the foregoing rules, the agent shall transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-ErrorResponse-PDU. The error-status field shall indicate genErr and the error-index field shall either indicate zero (0) or shall indicate the element within the structure that is preventing the operation.
- f) If none of the foregoing rules apply, the agent shall assign the requested value to the subject object. If the object is a block object, each sub-variable assignment specified by the request shall be effected as if simultaneously set with respect to all other assignments specified in the same message. The agent shall then transmit to the originator of the request an SFMP-Data-Packet containing an SFMP-SetResponse-PDU such that the request number field shall be that used in the SFMP-SetRequest-PDU.

4.2.2.3 Processing an SFMP Set Request-No Reply

Upon receipt of an SFMP-SetRequest-NoReply-PDU, an SFMP agent shall respond according to the following rules, in order:

- a) If the SFMP-SetRequest-NoReply-PDU does not contain a data field, the agent shall silently discard the data packet.
- b) If the object referenced by the value of the message-OID field is only available for get operations within the relevant MIB view, the agent shall silently discard the datagram and perform no further actions.
- c) If the value of the message-OID field does not exactly correspond to an object available for get or set operations in the relevant MIB view, the agent shall silently discard the datagram and perform no further actions.
- d) If the contents of the data field cannot be parsed to fit the SYNTAX defined for the referenced object, the agent shall silently discard the datagram and perform no further actions.
- e) If the value of the object referenced by the message-OID field cannot be altered for reasons not covered by any of the foregoing rules, the agent shall silently discard the datagram and perform no further actions.
- f) If none of the foregoing rules apply, the agent shall assign the requested value to the subject object. If the object is a block object, each sub-variable assignment specified by the request shall be effected as if simultaneously set with respect to all other assignments specified in the same message.

4.2.2.3 Confirmation of Request

An SFMP agent shall silently drop any SFMP-Data-Packet containing an SFMP-GetResponse-PDU, SFMP-SetResponse-PDU, or SFMP-ErrorResponse-PDU.

An SFMP management station should expect to receive a response message for each request transmitted, except a SetRequest-NoReply. As such, it should maintain a list of outstanding requests. Each new request should be assigned a request number that is not currently used by any outstanding request. As soon as this request number is assigned, the request should be added to the outstanding request list.

Upon receipt of an SFMP-SetResponse-PDU, the management station shall parse the data into the appropriate ASN.1 structure. If any information other than the request-number is present, the PDU shall be silently dropped. It should then remove the associated request from the outstanding request list. If the management station is unable to find the associated request, it should log the error and notify the end-application.

Upon receipt of a SFMP-GetResponse-PDU, the management station should remove the associated request from the outstanding request list and should parse the data field into the appropriate ASN.1 structure. If the management station cannot find the associated request or if the contents of the data field cannot be parsed properly, the error should be logged and the end-application notified; otherwise, the management station should provide the end-application with the parsed data.

Upon receipt of an SFMP-ErrorResponse-PDU, the management station should remove the associated request from the outstanding request list, log the error, and notify the end-application. If the management station is unable to find the associated request, it should log the error and notify the end-application.

The management station should periodically check the outstanding request list for abnormally old requests. Upon the discovery of any abnormally old request, the management station should remove the request from the outstanding request list, notify the end-application, and log the action.

NOTE—The maximum time for a response to be expected from an agent are specified depending on system design, communications infrastructure, and type of field device.

4.2.3 SFMP Data Packet Structures

All SFMP data packets shall conform to the general structure defined by SFMP-Data-Packet and shall be encoded according to NTCIP 1102:2004.

```
SFMP-Data-Packet ::= CHOICE {  
    sfmp-get                [0] SFMP-GetRequest-PDU,  
    sfmp-set                [16] SFMP-SetRequest-PDU,  
    sfmp-set-no-reply       [32] SFMP-SetRequest-NoReply-PDU,  
    sfmp-get-response       [PRIVATE 0] SFMP-GetResponse-PDU,  
    sfmp-set-response       [PRIVATE 16] SFMP-SetResponse-PDU,  
    sfmp-error              [PRIVATE 32] SFMP-ErrorResponse-PDU  
    ...  
}
```

Each of the above referenced PDU structures are based on the same core data structure, but they are distinguished by the formal rules that are defined for using this base structure. While this approach does not provide unique ASN.1 for each structure, it does facilitate the development of implementations by having one structure used for all operations. This core structure is the SFMP-PDU structure, defined as follows:

```
SFMP-PDU ::= SEQUENCE {  
    version          ENUMERATED{version-1(1), ...} DEFAULT version-1,  
    community-name   OCTET STRING           DEFAULT "public",  
    request-number   INTEGER (0..255)       OPTIONAL,  
    error-data       Error-Data             OPTIONAL,  
    message-OID      RELATIVE-OID           OPTIONAL,  
    data             OBJECT-TYPE.&Syntax     OPTIONAL  
    ...  
}
```

Sections 4.2.3.1 through 4.2.3.5 define the various rules and substructures of this data packet. Section 4.2.4 fully defines the meaning of each field.

4.2.3.1 Structure of SFMP-GetRequest-PDU

The SFMP get operation uses the SFMP-GetRequest-PDU.

```
SFMP-GetRequest-PDU ::= SFMP-PDU
```

The following rules shall apply to the SFMP-GetRequest-PDU:

- a) The request-number field shall be present
- b) The error-data field shall be absent
- c) The message-OID field shall be present
- d) The data field shall be absent

4.2.3.2 Structure of SFMP-SetRequest-PDU and SFMP-SetRequest-NoReply-PDU

The SFMP set and set-no-reply operations use an identical set of rules applied to the SFMP-PDU structure as indicated. The distinction between these two packets is made by the value of the encoded choice in the SFMP-Data-Packet, defined in Section 4.2.3.2, which always wraps the PDU structure.

```
SFMP-SetRequest-NoReply-PDU ::= SFMP-PDU  
SFMP-SetRequest-PDU ::= SFMP-PDU
```

The following rules shall apply to the SFMP-SetRequest-PDU and SFMP-SetRequest-NoReply-PDU:

- a) The request-number field shall be present
- b) The error-data field shall be absent
- c) The message-OID field shall be present
- d) The data field shall be present

4.2.3.3 Structure of SFMP-Get-Response

An SFMP get response uses the SFMP-GetResponse-PDU structure.

```
SFMP-GetResponse-PDU ::= SFMP-PDU
```

The following rules shall apply to the SFMP-GetResponse-PDU:

- a) The community-name field shall be absent, if the default value is used. The community name shall be present, if the community name is NOT "public", which is the default value
- b) The request-number field shall be present
- c) The error-data field shall be absent
- d) The message-OID field shall be absent
- e) The data field shall be present

4.2.3.4 Structure of the SFMP-Set-Response

An SFMP set response uses the SFMP-SetResponse-PDU structure.

```
SFMP-SetResponse-PDU ::= SFMP-PDU
```

The following rules shall apply to the SFMP-SetResponse-PDU:

- a) The community-name field shall be absent, if the default value is used. The community name shall be present, if the community name is NOT “public”, which is the default value
- b) The request-number field shall be present
- c) The error-data field shall be absent
- d) The message-OID field shall be absent
- e) The data field shall be absent

4.2.3.5 Structure of the SFMP-Error Message

An SFMP error response uses the SFMP-ErrorResponse-PDU structure.

```
SFMP-ErrorResponse-PDU ::= SFMP-PDU
```

```
Error-Data ::= SEQUENCE {  
    error-status      Error-Status,  
    error-index       Error-Index  
}
```

The following rules shall apply to the SFMP-ErrorResponse-PDU:

- a) The community-name field shall be absent, if the default value is used. The community name shall be present, if the community name is NOT “public”, which is the default value
- b) The request-number field shall be present
- c) The error-data field shall be present
- d) The message-OID field shall be absent
- e) The data field shall be absent

4.2.4 Definitions of Data Structure Fields

4.2.4.1 Definition of the Version Field

Definition: The version field shall define the version number of the SFMP data packet structure to which the data packet conforms. When responding, an agent shall use the same version number as used in the request. Currently, “version-1” is the only value defined for this field.

Representation: This field shall be encoded per the following ASN.1 Construct:

```
Version ::= ENUMERATED {version-1 (1), ...}
```

4.2.4.2 Definition of the Community Name Field

Definition: The SFMP community name is identical to the SNMP community name. It provides a simple non-encrypted password mechanism to prevent non-authorized users from accessing the agent database. The valid values of the SFMP community name and their allowed access are identical to those allowed within SNMP; within the context of NTCIP, these values are defined in the security table in Section 8 and Annex A.

NOTE—If multiple TMCs have access to the same device, the TMCs should use different community names to prevent problems arising during certain operations such as those dealing with the transaction mode.

Representation: This field shall be encoded per the following ASN.1 Construct:

```
Community-Name ::= OCTET STRING
```

4.2.4.3 Definition of the Request Number Field

Definition: The request number provides a mechanism by which the management station can reconcile incoming responses with outstanding requests. When an unreliable datagram service is used, the request number also provides a simple means of identifying messages duplicated by the network.

The request number is intended to be a relatively unique identification number for each request issued from a management station to a specific device. The management station may pick any appropriate algorithm for the selection of request numbers but a new request number should not duplicate outstanding requests that have not expired. An agent's response shall use the same request number as contained in the associated request.

Representation: This field shall be encoded per the following ASN.1 Construct:

```
Request-Number ::= INTEGER (0..255)
```

4.2.4.4 Definition of the Message OID Field

Definition: The SFMP message-OID field contains the Object Identifier of the object that is the subject of the message. This field shall include a valid instance number for the referenced object type.

Representation: This field shall be encoded per the following ASN.1 Construct:

```
Message-OID ::= RELATIVE-OID  
-- from {iso org dod internet private enterprises nema}
```

The RELATIVE-OID Data Type is a new ASN.1 Data Type that is able to encode only a portion of an overall OBJECT IDENTIFIER starting after the node specified by the associated comment field. The Relative OID is encoded identically to a normal OBJECT IDENTIFIER except that there are no special encoding rules for the first two tree nodes (e.g., the "iso org" nodes are normally encoded into a single sub-identifier with a value of 0X2B, no special rules are used with RELATIVE-OIDs).

4.2.4.5 Definition of the Data Field

Definition: The SFMP data field consists of the data referenced by the message OID.

Representation: This field shall be encoded per the SYNTAX field of the subject object's object-type macro. The formal reference in the ASN.1 is a reference to the Information Object Specification as shown:

```
objectType CLASS ::= {  
  &oid                OBJECT IDENTIFIER,  
  &Syntax  
}
```

The '&oid' field corresponds to the object identifier as defined in an object's macro (e.g., the ::= { security 1 } or similar field at the end of the OBJECT-TYPE macro). The &Syntax corresponds to the SYNTAX field of the OBJECT-TYPE macro.

NOTE—The "objectType.&Syntax" is technically a formal reference to an ASN.1 Information Object Specification that shall be considered the equivalent of the SNMP object definitions, which use the obsolete ASN.1 Macro format within its Management Information Bases. This anomaly is due to a problem in harmonizing different versions of ASN.1 within a single specification. The intent is that the "data" field shall be per the SYNTAX field of the subject object's object-type macro. See NTCIP 8004 v02.

4.2.4.6 Definition of the Error Status Field

Definition: The SFMP error status field identifies the type of error encountered by the agent while processing the associated request from central.

Representation: This field shall be encoded per the following ASN.1 Construct:

```
Error-Status ::= INTEGER {  
  tooBig      (1),  
  noSuchName (2),  
  badValue   (3),  
  readOnly   (4),  
  genErr     (5)  
}
```

NOTE—These error codes are consistent with those numbers assigned by SNMP (IAB Std. 15/RFC 1157).

The value of "noError (0)" as defined in SNMP is never valid within the design of SFMP and has been omitted from this list. If a management station receives any value not defined in this list, it shall treat it as a genErr.

The values are defined as:

- 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 nested field (or object identifier in the case of STMP) indicated by the index number is not supported by the agent.
- c) *badValue(3)*: this error can only occur during a set operation. The nested field (or object in the case of STMP) 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 nested field (or object in the case of STMP) 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 agent's documentation to determine what the error may be.

4.2.4.7 Definition of the Error Index Field

Definition: The SFMP error index field indicates the precise location of the data that resulted in the reported error status. In some cases, as detailed in Section 4.2.2, this value may be zero indicating that the error is due to a reason other than the value of the data field.

The value indicates the nested field number where the error was detected within the object. In this count, all optional and default objects are counted, even if they are not present in the encoding. In a sequence of, only those items present are counted (i.e., if the SEQUENCE OF only contains data for rows 3 and 5 of a table, it only considers these rows in the count). For example, if the error is contained in the third instance of a SEQUENCE OF SEQUENCE, and in the third field with the SEQUENCE having a total of five fields, the error index would be $5+5+3 = 13$.

Representation: This field shall be encoded per the following ASN.1 Construct:

```
Error-Index ::= INTEGER (0..255)
```

The value 255 shall mean the 256th field or greater.

4.3 EXAMPLES

The hexadecimal byte values in the left column are those data field bytes for the Application Level actually sent over the wire per OER. Lower layer protocols encapsulate these bytes as appropriate.

4.3.1 Get an Object Example

This an example of the SFMP data packets for a get and get-response for globalTime.0.

Bytes	SFMP Get-Request Data-Packet
80	CHOICE = [0] (i.e., context specific) = sfmp-get
	SFMP-GetRequest-PDU = SEQUENCE
14	Preamble = 0001 0100 =
	Bit 1 = 0 – extension absent
	Bit 2 = 0 – default version = version-1
	Bit 3 = 0 – default community name = “public”
	Bit 4 = 1 – request-number present
	Bit 5 = 0 – error-data absent
	Bit 6 = 1 – message-oid present
	Bit 7 = 0 – data absent
	Bit 8 = 0 – RESERVED
01	request number = 1
06 04 02 06 03 01 00	message-oid of 6 bytes at nema.4.2.6.3.1.0 = globalTime.0

NOTE—For more information on the definition of the preamble and how to encode default fields, see NTCIP 1102:2004.

SFMP Get-Response Data-Packet
C0 CHOICE = [PRIVATE 0] = sfmp-get-response
SFMP-GetResponse-PDU = SEQUENCE
12 Preamble = 0001 0010 =
Bit 1 = 0 – extension absent
Bit 2 = 0 – default version = version-1
Bit 3 = 0 – default community name = “public”
Bit 4 = 1 – request-number present
Bit 5 = 0 – error-data absent
Bit 6 = 0 – message-oid absent
Bit 7 = 1 – data present
Bit 8 = 0 – RESERVED
01 request number = 1
3A 24 63 20 data = globalTime.0 = 975463200 = November 29, 2000 at 2:00 am
UTC.

4.3.2 Get Block Object with Community Name Example

This an example of the SFMP data packets for a get and get-response for the globalTime.0 object, using a community name other than the default of “public.” The community name used in this example demonstrates that the community name may use any hexadecimal sequence and it is not restricted to printable ASCII.

SFMP Get-Request Data-Packet
80 CHOICE = [0] (i.e., context specific) = sfmp-get
SFMP-GetRequest-PDU = SEQUENCE
34 Preamble = 0011 0100 =
Bit 1 = 0 – extension absent,
Bit 2 = 0 – default version = version-1
Bit 3 = 1 – community name present
Bit 4 = 1 – request-number present
Bit 5 = 0 – error-data absent
Bit 6 = 1 – message-oid present
Bit 7 = 0 – data absent
Bit 8 = 0 – RESERVED
09 7E 6F 63 74 65 community name (first byte (09 hex) specifies length - 9 bytes) =
74 73 7E 99 “~octets~”
02 request number = 2
06 04 02 06 03 01 00 message-oid of 6 bytes at nema.4.2.6.3.1.0 = globalTime.0

SFMP Get-Response Data-Packet
C0 CHOICE = [PRIVATE 0] = sfmp-get-response
SFMP-GetResponse-PDU = SEQUENCE
12 Preamble = 0001 0010 =
Bit 1 = 0 – extension absent
Bit 2 = 0 – default version = version-1
Bit 3 = 0 – default community name = “public”
Bit 4 = 1 – request-number present
Bit 5 = 0 – error-data absent
Bit 6 = 0 – message-oid absent
Bit 7 = 1 – data present
Bit 8 = 0 – RESERVED
02 request number = 2
3A 24 63 20 data = globalTime.0 = 975463200 = November 29, 2000 at 2:00 am
UTC.

4.3.3 SFMP Set Example

This is an example of the SFMP data packets for a set and set-response for globalTime.0.

```

SFMP Set Request Data-Packet
90      CHOICE = [16] = sfmp-set
        SFMP-SetRequest-PDU = SEQUENCE
16      Preamble = 0001 0110 =
        Bit 1 = 0 – extension absent
        Bit 2 = 0 – default version = version-1
        Bit 3 = 0 – default community name = “public”
        Bit 4 = 1 – request-number present
        Bit 5 = 0 – error-data absent
        Bit 6 = 1 – message-oid present
        Bit 7 = 1 – data present
        Bit 8 = 0 – RESERVED
03      request number = 3
06 04 02 06 03 01 00  message-oid of 6 bytes at nema.4.2.6.3.1.0
3A 24 63 20          data = globalTime.0 = 975463200 = November 29, 2000 at 2:00 am UTC

```

```

SFMP Set Response Data-Packet
D0      CHOICE = [PRIVATE 16] = sfmp-set-response
        SFMP-Set-Response = SEQUENCE
10      Preamble = 0001 0000 =
        Bit 1 = 0 – extension absent
        Bit 2 = 0 – default version = version-1
        Bit 3 = 0 – default community name = “public”
        Bit 4 = 1 – request-number present
        Bit 5 = 0 – error-data absent
        Bit 6 = 0 – message-oid absent
        Bit 7 = 0 – data absent
        Bit 8 = 0 – RESERVED
03      request number = 3

```

4.3.4 SFMP Set Block Object Example

This example assumes a sample read-write block object defined under nema at node 1.1.1 with the following resolution of the OerString:

```

SampleBlockObject ::= SEQUENCE OF SEQUENCE {
    a      INTEGER,           -- set to 1, 4, 7
    b      INTEGER          DEFAULT 5, -- set to 2, 5, 8
    c      INTEGER (0..10), -- set to 3, 6, 9
    d      OCTET STRING,     -- all three set to “hi”
    e      OCTET STRING (SIZE 1) -- all three set to 0xFF
}

```

```

SFMP Set Request Data-Packet
90      CHOICE = [16] = sfmp-set
        SFMP-Set = SEQUENCE
36      Preamble = 0011 0110 =
        Bit 1 = 0 – extension absent
        Bit 2 = 0 – default version = version-1
        Bit 3 = 1 –community name present

```

Bit 4 = 1 – request-number present
Bit 5 = 0 – error-data absent
Bit 6 = 1 – message-oid present
Bit 7 = 1 – data present
Bit 8 = 0 – RESERVED

0D 61 64 6D 69 6E 69
73 74 72 61 74 6F 72
04
04 01 01 01 00

community name of 13 bytes = "administrator"
request number = 4
message-oid of 4 bytes at nema.1.1.1.0

data field = OerString
SEQUENCE OF

01 03

quantity of items = a one-byte value of 3
First SEQUENCE

80
01 01
01 02
03
02 68 69
FF

preamble = 1000 0000 = b is present
a = a one-byte value of 1
b = a one-byte value of 2
c = 3 (forced to one-byte by range)
d = a two-byte string of "hi"
e = 0xFF

Second SEQUENCE

00
01 04

preamble = 0000 0000 = b is not present
a = a one-byte value of 4
b defaults to 5
c = 6
d = a two-byte string of "hi"
e = 0xFF

Third SEQUENCE

80
01 07
01 08
09
02 68 69
FF

preamble = 1000 0000 = b is present
a = a one-byte value of 7
b = a one-byte value of 8
c = a one-byte value of 9
d = a two-byte string of "hi"
e = 0xFF

SFMP Set Response Data-Packet

D0

CHOICE = [PRIVATE 16] = sfmp-set-response
SFMP-Set-Response = SEQUENCE

10

Preamble = 0001 0000 =
Bit 1 = 0 – extension absent
Bit 2 = 0 – default version = version-1
Bit 3 = 0 – default community name = "public"
Bit 4 = 1 – request-number present
Bit 5 = 0 – error-data absent
Bit 6 = 0 – message-oid absent
Bit 7 = 0 – data absent
Bit 8 = 0 – RESERVED

04

request number = 4

4.3.5 Get Error Example

This is an example of an error when getting an object with an invalid OID.

SFMP Get Request Data-Packet

80

CHOICE = [0] (i.e., context specific) = sfmp-get

14 SFMP-GetRequest-PDU = SEQUENCE
Preamble = 0001 0100 =
Bit 1 = 0 – extension absent
Bit 2 = 0 – default version = version-1
Bit 3 = 0 – default community name = “public”
Bit 4 = 1 – request-number present
Bit 5 = 0 – error-data absent
Bit 6 = 1 – message-oid present
Bit 7 = 0 – data absent
Bit 8 = 0 – RESERVED
05 request number = 5
01 00 message-oid of 6 bytes at nema.0 = nema.0 (not an object)

SFMP Error-Response Data-Packet
E0 CHOICE = [PRIVATE 32] = sfmp-error
SFMP-ErrorResponse-PDU = SEQUENCE
18 Preamble = 0001 1000 =
Bit 1 = 0 – extension absent
Bit 2 = 0 – default version = version-1
Bit 3 = 0 – default community name = “public”
Bit 4 = 1 – request-number present
Bit 5 = 1 – error-data absent
Bit 6 = 0 – message-oid absent
Bit 7 = 0 – data present
Bit 8 = 0 – RESERVED
05 request number = 5
Error-Data = SEQUENCE
02 errorStatus = 2 = noSuchName
00 errorIndex = 0, problem with the object as a whole

4.3.6 Set Error Example

This is an example of an error when setting an object to a bad value using the structure defined in Section 4.3.4.

SFMP Set Request Data-Packet
90 CHOICE = [16] = sfmp-set
SFMP-Set = SEQUENCE
36 Preamble = 0011 0110 =
Bit 1 = 0 – extension absent
Bit 2 = 0 – default version = version-1
Bit 3 = 1 –community name present
Bit 4 = 1 – request-number present
Bit 5 = 0 – error-data absent
Bit 6 = 1 – message-oid present
Bit 7 = 1 – data present
Bit 8 = 0 – RESERVED
0D 61 64 6D 69 6E 69
73 74 72 61 74 6F 72
06 community name of 13 bytes = “administrator”
04 01 01 01 00 request number = 6
message-oid of 4 bytes at nema.1.1.1.0
data field = OerString
SEQUENCE OF
01 03 quantity of items = a one-byte value of 3
First SEQUENCE

80 preamble = 1000 0000 = b is present
01 01 a = a one-byte value of 1
01 02 b = a one-byte value of 2
03 c = 3 (forced to one-byte by range)
02 68 69 d = a two-byte string of "hi"
FF e = 0xFF

Second SEQUENCE
00 preamble = 0000 0000 = b is not present
01 04 a = a one-byte value of 4
b defaults to 5
06 c = 6
02 68 69 d = a two-byte string of "hi"
FF e = 0xFF

Third SEQUENCE
80 preamble = 1000 0000 = b is present
01 07 a = a one-byte value of 7
01 08 b = a one-byte value of 8
10 c = a one-byte value of 16 (an invalid value)
02 68 69 d = a two-byte string of "hi"
FF e = 0xFF

SFMP Error-Response Data-Packet

E0 CHOICE = [PRIVATE 32] = sfmp-error
SFMP-ErrorResponse-PDU = SEQUENCE
18 Preamble = 0001 1000 =
Bit 1 = 0 – extension absent
Bit 2 = 0 – default version = version-1
Bit 3 = 0 – default community name = "public"
Bit 4 = 1 – request-number present
Bit 5 = 1 – error-data present
Bit 6 = 0 – message-oid absent
Bit 7 = 0 – data absent
Bit 8 = 0 – RESERVED
06 request number = 6
Error-Data = SEQUENCE
03 errorStatus = 3 = badValue
0D errorIndex = 13, problem with the 13th field of the object

SECTION 5 SIMPLE TRANSPORTATION MANAGEMENT PROTOCOL (STMP)

NOTE—This definition of STMP is 100% backwards compatible with the definition provided in NEMA TS 3.2-1996. STMP requires SNMP or SFMP in order to allow a management station to configure the dynamic objects.

5.1 OVERVIEW

STMP is conceptually similar to SFMP, except that it has been designed to work with *dynamic objects*, i.e., block objects defined at run time, rather than just a set of predefined block objects. This has the benefit of providing the management station with the flexibility required to define its own messages (i.e., block object structures that avoid the need to include object identifiers and thereby significantly reduce the number of bytes), but as a result it significantly increases the complexity of the software within the agent. It also uses a number of the same truncations used by SFMP in order to decrease data packet size as compared to SNMP. This approach provides a potentially significant advantage in applications involving frequent polling on limited bandwidth links.

5.1.1 Dynamic Objects

NTCIP realized that it would be very difficult to reach consensus on a small set of fixed messages for some of the more complex devices such as traffic signal controllers. Yet, it was clear that these devices would still be called upon to frequently exchange status information over low speed communication circuits. As a result of this analysis, NTCIP developed the concept of a *dynamic object*, which is the major feature of STMP. A dynamic object is a simple sequence of specific NTCIP objects, similar to a block object, but the component objects within a dynamic object are defined at run-time by the management station rather than being defined in a static MIB.

The management station shall only configure the dynamic objects using SNMP or SFMP messages to set the values in the dynamic object configuration table, defined in Section 5.1.1.1, and the dynamic object definition table, defined in Section 5.1.1.2.

5.1.1.1 Dynamic Object Configuration Table

The *dynObjConfigTable* is a table indicating the owner and status of each dynamic object. Due to historic reasons explained in 5.1.1.2, its index, the *dynObjNumber*, is located under a different node on the ISO Naming Tree, but this has no operational impact. The *dynObjConfigTable* has conceptual rows that contain the objects shown in Table 2.

Table 2 Columns of the Dynamic Object Configuration Table

dynObjNumber	dynObjConfigOwner	dynObjConfigStatus
--------------	-------------------	--------------------

The INDEX for a particular row in the *dynObjConfigTable* is defined by *dynObjNumber*. It identifies with which of the 13 dynamic objects this row of the table is associated.

The intent of the *dynObjConfigOwner* object is to indicate the identity of the owner that defined the dynamic object.

The *dynObjConfigStatus* indicates the status of the dynamic object. The allowed states of each dynamic object are defined by the **ConfigEntryStatus** type, as defined in Section 5.2.4.1. The status may be *valid*, *invalid*, or *underCreation*.

5.1.1.2 Dynamic Object Definition Table

In addition to defining the state (and optionally the owner) of each dynamic object, the management station also defines the desired contents of the dynamic object. One approach to solving this problem would have been to define a series of additional fields in the *dynObjConfigTable*, such as *dynObjField1*, *dynObjField2*, *dynObjField3*, etc. where each was an OBJECT IDENTIFIER pointing to the desired object. However, this would have resulted in a large number of very similar object types. Instead, NTCIP defined an embedded table to contain the list of objects. The embedded table, called the *dynObjDef* table uses the same *dynObjNumber* as the primary index; it then uses a secondary index, *dynObjIndex*, to indicate the position of the referenced variable in the dynamic object. Finally, the *dynObjVariable* references the specific object to be included in the indicated field in the indicated dynamic object. Because the table is embedded, the editing rules imposed by the *dynObjConfigStatus* parameter affect the access to the cells of this table as well. See Table 3.

Table 3 Composite Table for Dynamic Object Configuration and Definition

dynObjNumber	dynObjConfigOwner	dynObjConfigStatus	dynObjIndex	dynObjVariable
1	<Owner of Dynamic Object #1>	<Status of Dynamic Object #1>	1	<OID of 1 st object in dynObj 1>
			2	<OID of 2 nd object in dynObj 1>
			3	<OID of 3 rd object in dynObj 1>
		
			255	<OID of 255 th obj. in dynObj 1>
2	<Owner of Dynamic Object #2>	<Status of Dynamic Object #2>	1	<OID of 1 st object in dynObj 2>
			2	<OID of 2 nd object in dynObj 2>
			3	<OID of 3 rd object in dynObj 2>
		
			255	<OID of 255 th obj. in dynObj 2>
3	<Owner of Dynamic Object #3>	<Status of Dynamic Object #3>	1	<OID of 1 st object in dynObj 3>
			2	<OID of 2 nd object in dynObj 3>
			3	<OID of 3 rd object in dynObj 3>
		
			255	<OID of 255 th obj. in dynObj 3>
...
13	<Owner of Dynamic Object #13>	<Status of Dynamic Object #13>	1	<OID of 1 st obj in dynObj 13>
			2	<OID of 2 nd obj in dynObj 13>
			3	<OID of 3 rd obj in dynObj 13>
		
			255	<OID of 255 th obj - dynObj 13>

NOTE—Version 1 of the Transportation Management Information Base (TMIB) defined in NEMA TS 3.2/NTCIP 1101:1996 had an owner parameter and a status parameter for each dynamic object variable. Deployment experience indicated that this design was less than ideal and thus NEMA TS 3.2/NTCIP 1101:1996 was changed with Amendment 1 in 1998. The original objects, *dynObjOwner* and *dynObjStatus*, were deprecated in that Amendment and replaced with the *dynObjConfigOwner* and *dynObjConfigStatus* objects as described. This ensured that each dynamic object would only have a single owner and status at any one time.

5.1.1.3 Dynamic Objects and System Operation

STMP supports 13 dynamic objects for each agent. In theory, the management station could configure each device with a different set of dynamic objects, but in practice, most management stations are likely to configure similar devices with similar dynamic object definitions.

5.1.2 Other Truncations

Because there are a small number of dynamic objects defined by the protocol, the message identifier only requires four bits rather than multiple bytes. STMP also takes advantage of other encoding and design truncations to minimize the data packet size. For example, a password is not required because the dynamic objects are defined at run-time; a low level of security is already provided by the fact that the structure of the data stream is not published in a standard.

5.2 DEFINITION

All STMP implementations shall be declared to be a management station, agent, or both.

Management stations may transmit STMP-GetRequest-PDU's, STMP-GetNextRequest-PDU's, STMP-SetRequest-PDU's, and STMP-SetRequest-NoReply-PDU's as necessary in order to manage agents. Management stations shall be able to receive and process STMP-GetResponse-PDU's, STMP-SetResponse-PDU's, and STMP-ErrorResponse-PDU's per the rules defined within Section 5.2.

Agents shall be able to receive and process STMP-GetRequest-PDU's, STMP-GetNextRequest-PDU's, STMP-SetRequest-PDU's, and STMP-SetRequest-NoReply-PDU's per the rules defined within Section 5.2, including the ability to transmit STMP-GetResponse-PDU's, STMP-SetResponse-PDU's, and STMP-ErrorResponse-PDU's as required.

The transmission of STMP data packets shall be governed by the rules defined in Section 5.2.1 and shall conform to the structures defined in Section 5.2.2.

5.2.1 Rules

Similar to SNMP and SFMP, STMP models all device functions as alterations (i.e., SETs) or inspections (i.e., GETs) of variables (i.e., objects). This strategy avoids the complexities of defining a different message type as a part of the protocol definition for each command desired.

Communication among protocol entities is accomplished by the exchange of protocol messages, each of which may be entirely and independently represented within a single datagram. An implementation of this protocol shall accept any valid message whose length does not exceed 484 octets. However, implementations may support larger messages.

A management station must take into account the variable binding list processing nature of TMP. In TMP, all objects contained in a single set-request data packet appear to be set to their new values simultaneously. Therefore, a management station shall not combine a state change request with a request to set an instance value associated with that state change. If such an operation is attempted, the operation may not be correctly processed. This rule applies to both the process to configure a dynamic object as well as the execution of a dynamic object.

5.2.2 Elements of Procedure

Section 5.2.2 describes the actions of a protocol entity implementing the STMP; however, it is not intended to constrain the internal architecture of any conformant implementation.

An STMP protocol entity transmits an STMP message by passing the serialized message to a transport service capable of sending the message to the peer protocol entity.

An STMP protocol entity receives an STMP message as follows:

- a) It performs a rudimentary parse of the incoming data packet to build a structure containing the Message Type, the Object Identifier and the associated data as contained in the Information Field. If the parse fails, e.g., if one of the fields contained invalid data, the protocol entity discards the data

packet and performs no further actions.

- b) The protocol entity then processes the message according to the rules defined in Sections 5.2.2.2 and 5.2.2.3.

5.2.2.1 Initiating a Request

An STMP management station may, at any time, initiate a get or set operation by generating an STMP-Data-Packet containing an STMP-GetRequest-PDU, an STMP-GetNextRequest-PDU, an STMP-SetRequest-PDU, or an STMP-SetRequest-NoReply-PDU. The conditions that may result in the management station initiating such a request are the subject of the end-application functionality and are beyond the scope of NTCIP 1103 v02.

An STMP agent shall not issue an STMP-Data-Packet containing an STMP-GetRequest-PDU, an STMP-GetNextRequest-PDU, an STMP-SetRequest-PDU, or an STMP-SetRequest-NoReply-PDU. This restriction shall not preclude a single device from acting as both a management station and an agent.

5.2.2.2 Processing a Request

An STMP management station shall silently drop any STMP-GetRequest-PDU, STMP-GetNextRequest-PDU, SFMP-SetRequest-PDU, or SFMP-SetRequest-NoReply-PDU.

The STMP agent shall process all requests in accordance with Sections 5.2.2.2.1 through, including processing the request sufficiently to generate the transmission of the appropriate response (assuming that the STMP agent has permission to transmit) within the maximum Response Time. If the specification does not indicate the maximum Response Time, the maximum Response Time shall be 100 milliseconds plus one millisecond for each byte in the response STMP PDU Information field.

5.2.2.2.1 Processing an STMP Get Request

Upon receipt of an STMP-GetRequest-PDU, an STMP agent shall respond according to the following rules, in order:

- a) If the STMP-GetRequest-PDU contains an information field, the agent shall silently drop the data packet.
- b) The Subject Dynamic Object shall be defined to be the Dynamic Object that has a dynObjNumber which is equal to the value of the Object Identifier field.
- c) If the dynObjConfigStatus of the Subject Dynamic Object is not *valid*, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate noSuchName, and the error-index field shall indicate zero (0).
- d) If the Subject Dynamic Object contains a Referenced Object that is not currently instantiated, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate noSuchName and the error-index field shall indicate the dynObjIndex number of the problem Referenced Object.
- e) If the size of the get-response would exceed a local limitation, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status fields shall indicate tooBig and the error-index field shall indicate zero (0).
- f) If the value of the Subject Dynamic Object cannot be retrieved for reasons not covered by any of the foregoing rules, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate genErr and the error-index field shall indicate the element within the structure that is preventing the operation, unless this is unknown, in which case it shall indicate a zero (0).

- g) If none of the foregoing rules apply, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-GetResponse-PDU such that the Object Identifier field shall indicate the dynamic object number and the data field shall contain the Dynamic Object Data.

5.2.2.2.2 Processing an STMP Get Next Request

Upon receipt of an STMP-GetNextRequest-PDU, an STMP agent shall respond according to the following rules, in order:

- a) If the STMP-GetNextRequest-PDU contains an information field, the agent shall silently drop the data packet.
- b) The Subject Dynamic Object shall be defined to be the Dynamic Object with a dynObjConfigStatus of *valid* that lexicographically follows the number contained in the Object Identifier field of the STMP data packet.
- c) If there is no such Subject Dynamic Object, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the value used in the request, the error-status field shall indicate noSuchName and the error-index field shall indicate zero (0).
- d) If the Subject Dynamic Object contains a Referenced Object that is not currently instantiated, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate noSuchName and the error-index field shall indicate the dynObjIndex number of the problem Referenced Object.
- e) If the size of the get-response would exceed a local limitation, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status fields shall indicate tooBig and the error-index field shall indicate zero (0).
- f) If the Subject Dynamic Object cannot be retrieved for reasons not covered by any of the foregoing rules, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate genErr and the error-index field shall indicate the element within the structure that is preventing the operation, unless this is unknown, in which case it shall indicate a zero (0).
- g) If none of the foregoing rules apply, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-GetResponse-PDU. The Object Identifier field shall contain the dynamic object number of the Subject Dynamic Object and the data field shall contain the Dynamic Object Data for that Dynamic Object.

5.2.2.2.3 Processing an STMP Set Request

Upon receipt of an STMP-SetRequest-PDU, an STMP agent shall respond according to the following rules, in order:

- a) The Subject Dynamic Object shall be defined to be the Dynamic Object that has a dynObjNumber which is equal to the value of the Object Identifier field.
- b) If the dynObjConfigStatus of the Subject Dynamic Object is not *valid*, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate noSuchName and the error-index field shall indicate zero (0).
- c) If the Subject Dynamic Object contains a Referenced Object that is only available for get operations, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate readOnly and the error-index field shall indicate the dynObjIndex number of the Referenced Object.
- d) If the contents of the Information field cannot be parsed to fit the SYNTAX defined for the referenced

object, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate badValue and the error-index field shall indicate the field number at which the parsing first failed.

- e) If any of the Referenced Objects of the Subject Dynamic Object cannot be altered for reasons not covered by any of the foregoing rules, the agent shall transmit to the originator of the request an STMP-Data-Packet containing an STMP-ErrorResponse-PDU. The Object Identifier field shall indicate the Subject Dynamic Object, the error-status field shall indicate genErr and the error-index field shall either indicate zero (0) or shall indicate the element within the structure that is preventing the operation.
- f) If none of the foregoing rules apply, the agent shall assign the requested values to the subject Referenced Objects. Each Referenced Object assignment specified by the request shall be effected as if simultaneously set with respect to all other assignments specified in the same message. The agent shall then transmit to the originator of the request an STMP-Data-Packet containing an STMP-SetResponse-PDU such that the request number field shall be the same as that which was used in the set request.

5.2.2.2.4 Processing an STMP Set Request-No Reply

Upon receipt of an STMP-SetRequest-NoReply-PDU, an STMP agent shall respond according to the following rules, in order:

- a) The Subject Dynamic Object shall be defined to be the Dynamic Object that has a dynObjNumber which is equal to the value of the Object Identifier field.
- b) If the dynObjConfigStatus of the Subject Dynamic Object does not equal *valid*, the agent shall discard the data packet and perform no further action.
- c) If any of the Referenced Objects in the Subject Dynamic Object is only available for get operations, the agent shall discard the data packet and perform no further action.
- d) If the contents of the data field cannot be parsed to fit the SYNTAX defined for the referenced object, the agent shall discard the data packet and perform no further action.
- e) If the value of any of the Referenced Objects cannot be altered for reasons not covered by any of the foregoing rules, the agent discard the data packet and perform no further action.
- f) If none of the foregoing rules apply, the agent shall assign the requested values to the Referenced Objects. Each Referenced Object assignment specified by the request shall be effected as if simultaneously set with respect to all other assignments specified in the same message. The agent shall not transmit any response.

5.2.2.3 Confirmation of Request

An STMP agent shall silently drop any STMP-Data-Packet containing an STMP-GetResponse-PDU, STMP-SetResponse-PDU, or STMP-ErrorResponse-PDU.

Upon receipt of a STMP-GetResponse-PDU, the management station should parse the data field into the appropriate ASN.1 structure. If the contents of the data field cannot be parsed properly, the error should be logged and the end-application notified; otherwise, the management station should provide the end-application with the parsed data.

Upon receipt of an STMP-SetResponse-PDU, the management station shall parse the data into the appropriate ASN.1 structure. If any information field is present, the PDU shall be silently dropped. It should then remove the associated request from the outstanding request list. If the management station is unable to find the associated request, it should log the error and notify the end-application.

Upon receipt of an error message, the management station should log the error, and notify the end-application.

5.2.3 STMP Data Packet Structure

The STMP-Data-Packet is defined to have a header field, defined in Section 5.2.3.1, and an information field (or PDU), defined in Section 5.2.3.2. The header field can be further subdivided into a PDU Format Bit, which is always one, a message type bit field, which is three bits in length, and a dynamic object identifier bit field, which is four bits in length. This is shown in Figure 3.

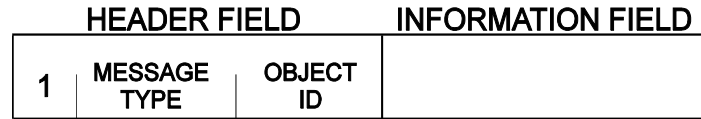


Figure 3 STMP PDU Fields

5.2.3.1 Header Field

The header field shall be one byte in length. It is the same byte used by TMP in order to multiplex the three component protocols into a single structure. As such, Table 1 provides a mapping between the possible values for this field and the proper meanings. Table 4 provides a further explanation of the Header Field specifically for STMP.

Table 4 STMP Header Field Explanation

BIT	CONTENTS	Description
7	<i>PDU Format</i>	
	0	Reserved by TMP for SNMP and any other future uses.
	1	Indicates that packet is STMP or SFMP.
6-4	<i>Message Type</i>	NOTE—The following descriptions for this (Message Type) field apply only to STMP packets (i.e., when the PDU Format is 0x1 and Object ID is between 0x0001 and 0x1101).
	000	An STMP-GetRequest-PDU is contained in the packet.
	001	An STMP-SetRequest-PDU is contained in the packet.
	010	An STMP-SetRequest-NoReply-PDU is contained in the packet.
	011	An STMP-GetNextRequest-PDU is contained in the packet.
	100	An STMP-GetResponse-PDU is contained in the packet (positive ACK).
	101	An STMP-SetResponse-PDU is contained in the packet (positive ACK).
	110	An STMP-ErrorResponse-PDU is contained in the packet.
	111	Reserved by TMP for future use.
3-0	<i>Object ID</i>	
	0000	Reserved by TMP for SFMP, See Section 4.
	0001-1101	ID of STMP "dynamic object"
	1110	Reserved by TMP for future use.
	1111	Reserved by TMP for future use.

5.2.3.2 PDU Information Field

The PDU Information field shall be empty for STMP-GetRequest-PDU's, STMP-GetNextRequest-PDU's, and STMP-SetResponse-PDU's.

The PDU Information field for STMP-GetResponse-PDU's, STMP-SetRequest-PDU's, and STMP-SetRequestNoReply-PDU's shall be the Dynamic Object Data structure as defined in Section 5.2.4.3.

The PDU Information field of an STMP-ErrorResponse-PDU shall only contain the error status and error index information according to the following structure:

```
STMP-ErrorResponse-PDU ::=
  SEQUENCE {
    error-status
      ErrorStatus,

    error-index
      ErrorIndex
  }
```

The precise definition of these fields is identical to that defined for SFMP; see Sections 4.2.4.6 and 4.2.4.7.

5.2.4 Dynamic Object Configuration

As discussed in Section 5.1.1, the dynamic object tables use the `dynObjVariable` field to define the content of each dynamic object; each object referenced by a `dynObjVariable` is termed a Referenced Object. The formal definitions of dynamic object tables are contained in Annex A.

5.2.4.1 Configuration Entry Status

The `ConfigEntryStatus` type shall be used to manage the Dynamic Object Definition (`dynObjDef` table) and Dynamic Object Configuration (`dynObjConfigTable`) tables. For each dynamic object there is a columnar object that is defined with a SYNTAX of `ConfigEntryStatus`.

All other columnar objects for the subject dynamic object shall have operations limited by the current value of the `ConfigEntryStatus` object in the row. The meaning of the values is as follows:

- a) If the current state of the `configEntryStatus` object is *invalid*, the information in the corresponding row of the `dynObjConfigTable` and the corresponding rows of the `dynObjDef` table with the same index `dynObjNumber` shall be considered undefined. Setting the status object to *invalid* has the effect of invalidating and clearing the corresponding rows of the Dynamic Object Definition Table. It is implementation specific whether the agent clears the values contained in the invalidated rows or de-allocates the memory associated with the invalidated rows. When in the *invalid* state, the agent shall reject any request to go to the *valid* state.
- b) If the current state of the `configEntryStatus` object is *underCreation*, the memory for the corresponding row of the `dynObjConfigTable` and the corresponding rows of the `dynObjDef` table with the same index `dynObjNumber` shall have been allocated, but may contain some invalid data. When in this state, the management application is allowed to modify the values of the objects contained in the associated rows of the table. Once this operation is completed, the management station may set the state to *valid*; alternatively, the management station may cancel the operation by setting the state to *invalid*.
- c) If the current state of the `configEntryStatus` object is *valid*, the corresponding row of the `dynObjConfigTable` and the corresponding rows of the `dynObjDef` table with the same index `dynObjNumber` contain information that is believed to be valid.

Table 5 indicates the actions that shall take place upon receipt of a set request to change the state of `dynObjConfigStatus`. The value of each cell in the table shows the result of receiving the indicated set request (column headings) when the device is in the indicated current state (row headings).

Table 5 State Transition for ConfigEntryStatus

CURRENT STATE	REQUESTED STATE		
	<i>Invalid</i>	<i>underCreation</i>	<i>Valid</i>
<i>invalid</i>	invalid (1)	<i>underCreation</i> (6)	invalid (3)
<i>underCreation</i>	invalid (2)	<i>underCreation</i> (3)	valid (4) or <i>underCreation</i> (5)
<i>valid</i>	invalid (2)	valid (3)	valid (1)

NOTES

(1) No action takes place and response indicates noError.

(2) The state changes to invalid, all entries associated with the ConfigEntryStatus object are deleted or cleared and response indicates noError.

(3) No action takes place but response indicates badValue.

(4) If Dynamic Object Validation succeeds then state changes to *valid* and response indicates noError. (See 5.2.4.2.)

(5) If Dynamic Object Validation fails then state remains *underCreation* and response indicates genErr. (See 5.2.4.2.)

(6) The state changes to *underCreation* and the response indicates noError.

Upon receipt of a set request for the *valid* state when in the *underCreation* state, the agent shall attempt to validate the dynamic object data contained in the associated rows of the dynObjDef table with the same *dynObjNumber* of the associated request. If the validation is successful, the state shall change to *valid*, otherwise, the state remains in the *underCreation* state and the device shall return a genErr.

5.2.4.2 Dynamic Object Validation

The configuration of a dynamic object must be validated prior to using the dynamic object; the validation process is activated as defined in Section 5.2.4.1. When validating the configuration of a dynamic object, an agent shall perform the following consistency checks:

- a) For the row where *dynObjIndex* equals 1, the *dynObjVariable* shall point to a Referenced Object.
- b) For each value of *dynObjIndex* other than 1, the associated *dynObjVariable* shall be set to its Default Value, or both the associated *dynObjVariable* and the previous *dynObjVariable* (i.e., where *dynObjIndex* is one less) shall point to a Referenced Object.

Failure to pass these consistency checks shall prevent the state from changing to *valid*. Once defined and validated, the data referenced by the dynamic object shall be accessible via STMP.

5.2.4.3 Dynamic Object Data Structure

The Dynamic Object Data Structure, as used within the PDU Information field of some STMP messages, shall consist of a series of component fields, each encoding one Referenced Object. The component fields shall be encoded in order, according to the associated *dynObjIndex*, with the first field encoding the value of the first Referenced Object of the Dynamic Object, and the last field encoding the value of the last Referenced Object of the Dynamic Object. Each component field shall consist of the OER encoding of the subject component.

5.3 EXAMPLES

The following examples demonstrate:

- a) The process to configure a dynamic object,
- b) The retrieval of the configured dynamic object through an STMP get, and
- c) The setting of the dynamic object through an STMP set.

5.3.1 Configuring a Dynamic Object

Figure 4 demonstrates the process to configure a dynamic object to consist of the following objects, in order: (1) globalTime.0, (2) controller-standardTimeZone.0, and (3) eventClassDescription.1. The selection of these three objects for the example provides a robust example of how the STMP messages are encoded in Sections 5.3.2 and 5.3.3. It is assumed that the encoding of the referenced SNMP messages is understood, as there are a variety of textbooks on this subject matter.

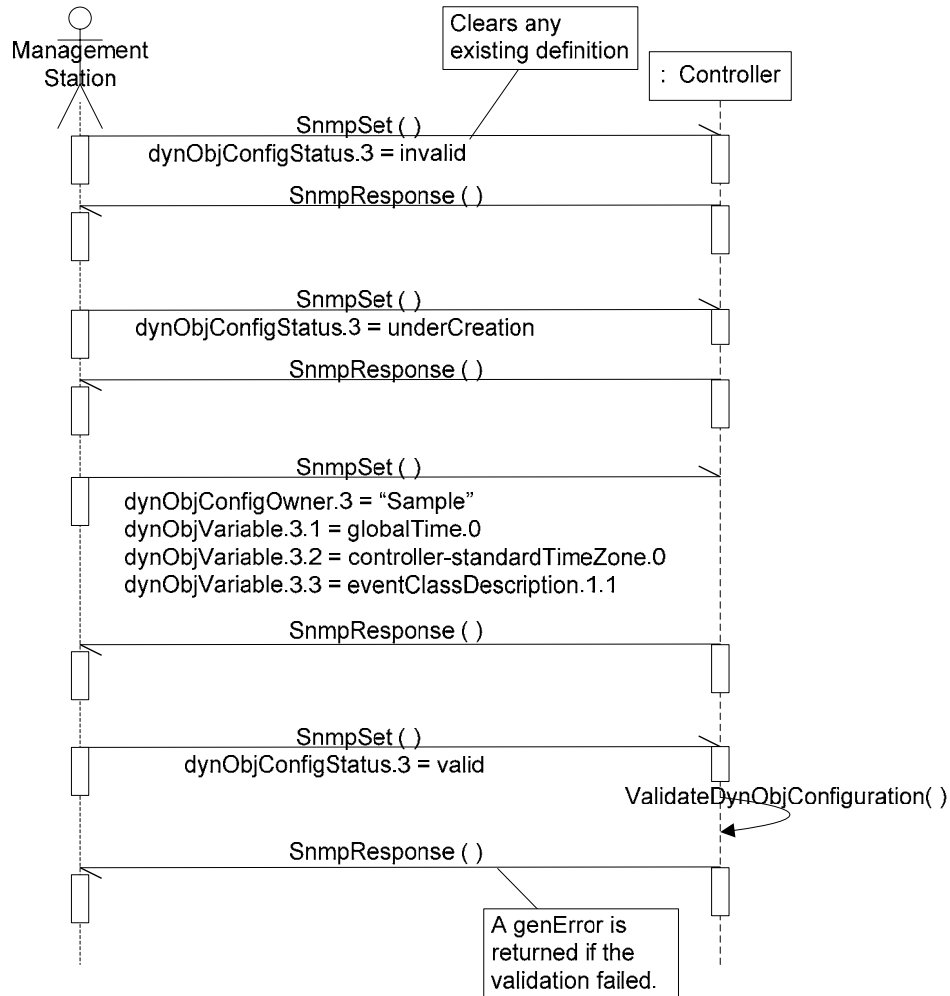


Figure 4 Configuring a Dynamic Object

5.3.2 Getting a Dynamic Object

A management station is able to retrieve the dynamic object defined above by issuing the one-byte STMP-GetRequest-PDU, as follows:

83 stmp-get for dynamic object #3

This command will cause the agent to generate an STMP-GetResponse-PDU as follows:

C3 stmp-get-response for dynamic object #3
 Information Field
3A 24 63 20 variable 1 = globalTime.0 = November 29, 2000 at 2:00 am UTC
FF FF B9 B0 variable 2 = controller-standardTimeZone.0 = -18000 = EST
06 53 61 6D 70 6C 65 variable 3 = eventClassDescription.1 (6 bytes) = "Sample"

5.3.3 Setting a Dynamic Object

Likewise a set operation would be of the following form:

93 stmp-set-request for dynamic object #3
 Information Field
3A 24 63 20 variable 1 = globalTime.0 = November 29, 2000 at 2:00 am UTC
FF FF B9 B0 variable 2 = controller-standardTimeZone.0 = -18000 = EST
06 53 61 6D 70 6C 65 variable 3 = eventClassDescription.1 (6 bytes) = "Sample"

And the response would be a single byte as follows:

D3 stmp-set-response for dynamic object #3

SECTION 6 NTCIP TRAPS

Section 6 is a placeholder.

NOTE—NTCIP 1103 v02 does not define rules regarding traps in NTCIP environments; however, a future version of NTCIP 1103 v02 is expected to do so. The future version is anticipated to include rules for TMP handling exception or event-driven reporting (i.e., to allow for an agent to send unsolicited information to the management station within a semi-controlled environment).

SECTION 7 LOGICAL NAMES

A number of NTCIP Application and Transport Profiles are based or modeled on Internet Protocols that use a 32-bit integer, called an IP Address, to identify the source and destination of messages. They typically appear in a dotted notation such as "206.239.7.229". Although this form provides a compact and efficient format when used electronically, people prefer to use pronounceable, easily remembered names such as "ftp.ntcip.org". To convert the name form to a number form, special processes are set up to provide the conversion. In the Internet, the names are referred to as domain names and the process that provides the conversion is a domain name service. If the domain name "ftp.ntcip.org" is sent to appropriate service, it will return the IP Address "206.239.7.229". This is the electronic address of the computer that hosts the NTCIP Webpage. The name and number form have been registered with the Internet Assigned Number Authority and are globally unique. No other computer on the Internet can have the same name or address.

In NTCIP center-to-center applications, there is a desire to provide a more human readable name form for IP Address and other functions. Because NTCIP does not require the setup of specialized computers to provide conversion and to avoid using the Internet term "domain" name, NTCIP refers to a readable name as a "logical name". When an implementation uses or permits identification using logical names, the Logical Name Translation Table defined in Clause A.7.2 can be used. This is a static table that, once filled out, can be used to convert one form to the other.

SECTION 8 SECURITY

TMP provides a basic level of security. However, the primary purpose of TMP security design is to prevent authorized users of the system from accessing data for which they are unauthorized.

Security against unauthorized users should be provided by lower layer services. For example, within a dedicated multi-drop system, a high degree of security is provided by the fact that the physical layer itself (i.e., the wires) is physically secure from typical hackers. Likewise, in dial-up networks, NTCIP standards recommend the use of the Challenge-Handshake Authentication Protocol (CHAP) to authenticate the remote entity. If additional levels of security are desired, off-the-shelf solutions, such as the Secure Sockets Layer, are fully compatible with NTCIP protocols.

The security mechanism provided by TMP is dependent upon which of the three component protocols are in use.

8.1 SNMP AND SFMP SECURITY

SNMP and SFMP use a common security scheme based on a simple authentication process. All SNMP data packets and all SFMP request data packets include a community name field. The community name field is an unencrypted octet string that associates the request with a user group. An agent can be configured to provide different user groups with varying levels of data access through the use of MIB views. The fact that the community name field is sent as an unencrypted octet string is a major security issue that can only be addressed by lower layer protocols.

A MIB view is a well-defined term from the SNMP community, and detailed information about MIB views is available in a variety of texts. In general, it allows objects that are defined as read-write to be viewed as if they were read-only or not-accessible when accessed via certain community names.

The mechanism to configure the visibility of data for each community name is provided by the security node of the TMP MIB as defined in Annex A. This node defines an object to hold the administrator community name. The administrator community name shall provide access to all objects defined in the device's MIB.

The security node also defines a security table that consists of columns for an index, a community name, and an access mask. Each bit of the access mask is a Boolean value that indicates whether a group of objects are read-write or read-only for a given community name. The assignment of objects to bits is manufacturer specific, except for the fact that the following objects shall not be assigned to any bit and shall be viewed as not-accessible for all community names within the table (this restriction does not apply to the administrator community name):

- a) All objects under the security node (Annex A)
{ nema transportation devices global security }
- b) All objects under the chap node (NTCIP 2301 v02 Annex B)
{ nema transportation protocols layers chap }
- c) Any objects so identified by various device standards

8.2 STMP SECURITY

The STMP provides a basic level of security based on the fact that the data packet is not self-defining. Instead, the content of each data packet requires each protocol entity to have prior knowledge of the configuration of each dynamic object. This configuration information is only accessible via SNMP or SFMP.

The following objects shall not be assigned to any dynObjVariable:

- a) All objects under the "security" node (Annex A)
{ nema transportation devices global security }
- b) All objects under the "dynObjMgmt" node (Annex A)
{nema transportation protocols dynObjMgmt}
- c) All objects under the "chap" node (NTCIP 2301 v02 Annex B)
{ nema transportation protocols layers chap }
- d) Any objects so identified by various device standards

The likely security threads for STMP at the application layer are due to the fact that SNMP or SFMP needs to be used to configure STMP dynamic objects. These unencrypted data exchanges might be captured, if the data exchanges take place over publicly accessible networks. As indicated with SNMP, the lower layer protocols will need to be employed to avoid this security thread.

SECTION 9
CONFORMANCE STATEMENT

NTCIP conformance requirements for NTCIP 1103 v02 are defined in NTCIP 2301 v02.

**Annex A
(Normative)
TRANSPORTATION MANAGEMENT PROTOCOLS
MANAGEMENT INFORMATION BASE**

Annex A defines those objects necessary to configure, manage and monitor various aspects of the Transportation Management Protocols. The objects are defined using the OBJECT-TYPE macro defined in RFC 1212. The text provided in Annex A contains several standard MIBs addressing these various aspects of configuration, monitoring, and managing functions. Each of the MIBs has their own header in order to allow use of singular or multiple MIBs within an implementation. The names of each MIB are shown following the MIB Header. All of the MIBs defined in Annex A together replace the TMIB II.

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 v01 shall apply.

Annex A presents the objects in lexicographical order of their OBJECT IDENTIFIERS which correspond to their physical location within the global naming tree. All of the objects defined in NTCIP 1103 v02 reside under the "nema" node of the global naming tree. To aid in object management, the "nema" node has been subdivided into logical categories, each defined by a node under the "nema" node. The individual objects are then located under the appropriate node.

Nodes should not be confused with conformance requirements, which are defined in profiles. The NTCIP profile for NTCIP 1103 v02 is NTCIP 2301 v02. Conformance requirements are based on logical groupings of objects that provide specific features that may be desired in a device. While the conformance requirements frequently correspond to the nodal structure, a conformance group may contain objects that are not lexicographically ordered.

NOTE—NTCIP 1103 v02 uses NTCIP 8004 v02 conventions. NTCIP 1103 v02 specifies all (non-deprecated/non-obsolete) objects to be optional according to the conventions stated in NTCIP 8004 v02; it is the responsibility of any document referring to NTCIP 1103 v02 to specify exactly which objects shall be supported under what conditions through a Protocol Requirements List (PRL).

Text preceded by a double hyphen in the MIB definitions represents normative text for NTCIP 1103 v02.

A.1. TMP SNMP MIB Header

```
-- Filename:      1103v0215-SNMP.MIB
-- Description:   This MIB defines various objects related to managing and
--               monitoring the SNMPSNMP Protocol.  Specifically, these
--               include objects related to:
--               (a) configuration of objects,
--               This MIB replaces portions of TMIB-II.
--
-- MIB Revision History:
-- 08/01/96      Original standard approved
-- 01/01/98      Preliminary Release of TS 3.2 TMIB MIB formatted for 80
--               columns and no TABs
-- 01/07/98      Replaced some missed TABs with spaces
-- 07/08/98      Added Copyright Notice
-- 10/07/98      Amendment 1
-- 03/09/00      Removed all the special edits to that were done to use the
--               SMIC Compiler
--               Defined DisplayString to eliminate reference to RFC 1212
```

-- END is still left "Dynamic Object Data" group
-- Changed filename and updated copyright years
-- Updated the MIB to Amendment 1
-- 08/09/00 Modified header format and wording of copyright and MIB
-- 11/16/01 Added objects for sfmp and stmp statistics
-- Moved security node into this MIB from NTCIP 1201
-- Added objects to support logical names
-- Renamed the module to NTCIP1103-A-2002 from TMIB-II
-- Renamed the text name to Transportation Management
-- Protocols MIB from Transportation MIB
-- Changed STATUS of all objects to optional to reflect new
-- conformance rules being defined in NTCIP 8004
-- 06/08/04 Moved report node into this MIB from NTCIP 1201
-- 09/27/04 Changed name of file.
-- 10/11/04 Per KLV e-mails 10/08/04 updated version and
-- Changed Index on logicalNameTranslation-index
-- from (0..255) to (1..255)
-- Changed all STATUS optional to mandatory
-- 06/13/05 Updated filename.
-- 08/09/05 Added object definitions for Trap Management, Watch Block
Objects
-- Updated filename; re-instantiated the OIDs for the
-- 13 dynamic objects.
-- 04/19/06 Broke the various logical groupings of objects into separate
-- MIBs to allow for separate compiling for various deployment
-- needs.
-- 10/09/07 Changed the Description fields of the objects to conform to the
-- new version of NTCIP 8004 v02.
-- 12/14/07 Changed the name of this MIB only to reflect the version number.
-- 02/13/09 Changed the name of this MIB only to reflect the version number.
--

-- DISTRIBUTION NOTICE

--Copyright 1996 - 2008 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation Engineers
--(ITE), and the National Electrical Manufacturers Association (NEMA). All
--intellectual property rights, including, but not limited to, the rights of
--reproduction in whole or in part in any form, translation into other
--languages and display are reserved by the copyright owners under the laws of
--the United States of America, the Universal Copyright Convention, the Berne
--Convention, and the International and Pan American Copyright Conventions.
--Except for the MIB, Do not copy without written permission of either AASHTO,
--ITE, or NEMA.

-- Joint NEMA, AASHTO, and ITE
-- NTCIP Management Information Base
-- DISTRIBUTION NOTICE
--

--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:

- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "-",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
--(ii) use of the MIB is restricted in that the syntax field may be modified

```
--only to reflect a more restrictive sub-range or enumerated values;  
--(iii) the description field may be modified but only to the extent that:  
--(a) only those bit values or enumerated values that are supported are  
--listed; and (b) the more restrictive subrange is expressed.  
--  
--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 OR 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, OR 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.  
--  
--NTCIP is a trademark of AASHTO/ITE/NEMA.  
--*****
```

```
NTCIP1103v0215-SNMP DEFINITIONS ::= BEGIN
```

```
IMPORTS  
    OBJECT-TYPE  
        FROM RFC-1212  
application  
    FROM NTCIP8004-A-2004;  
-- EXPORTS EVERYTHING
```

A.1.1. Objects for SNMP

```
snmpConfig OBJECT IDENTIFIER ::= {application 1}  
-- <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.1
```

A.1.2. Maximum SNMP Packet Size Parameter

```
snmpMaxPacketSize OBJECT-TYPE  
    SYNTAX  INTEGER (484..65535)  
    ACCESS  read-only  
    STATUS  mandatory  
    DESCRIPTION  
        "<Definition> Indicates the maximum packet size,  
        in octets, that the SNMP agent supports for  
        reception or transmission."  
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.1.1  
    ::= {snmpConfig 1}  
END -- NTCIP1103v0215-SNMP
```

A.2. TMP SFMP MIB Header

```
-- Filename:      1103v0215-SFMP.MIB
-- Description:   This MIB defines various objects related to managing and
--               monitoring the Simple Fixed Management Protocol (SFMP).
--               Specifically, these include objects related to:
--               (a) communication statistics,
-- This MIB replaces portions of TMIB-II.
--
-- MIB Revision History:
-- 08/01/96      Original standard approved
-- 01/01/98      Preliminary Release of TS 3.2 TMIB MIB formatted for 80
--               columns and no TABs
-- 01/07/98      Replaced some missed TABs with spaces
-- 07/08/98      Added Copyright Notice
-- 10/07/98      Amendment 1
-- 03/09/00      Removed all the special edits to that were done to use the
--               SMIC Compiler
--               Defined DisplayString to eliminate reference to RFC 1212
--               END is still left "Dynamic Object Data" group
--               Changed filename and updated copyright years
--               Updated the MIB to Amendment 1
-- 08/09/00      Modified header format and wording of copyright and MIB
-- 11/16/01      Added objects for sfmp and stmp statistics
--               Moved security node into this MIB from NTCIP 1201
--               Added objects to support logical names
--               Renamed the module to NTCIP1103-A-2002 from TMIB-II
--               Renamed the text name to Transportation Management
--               Protocols MIB from Transportation MIB
--               Changed STATUS of all objects to optional to reflect new
--               conformance rules being defined in NTCIP 8004
-- 06/08/04      Moved report node into this MIB from NTCIP 1201
-- 09/27/04      Changed name of file.
-- 10/11/04      Per KLV e-mails 10/08/04 updated version and
--               Changed Index on logicalNameTranslation-index
--               from (0..255) to (1..255)
--               Changed all STATUS optional to mandatory
-- 06/13/05      Updated filename.
-- 08/09/05      Added object definitions for Trap Management, Watch Block
Objects
--               Updated filename; re-instantiated the OIDs for the
--               13 dynamic objects.
-- 04/19/06      Broke the various logical groupings of objects into separate
--               MIBs to allow for separate compiling for various deployment
--               needs.
-- 10/09/07      Changed the Description fields of the objects to conform to the
--               new version of NTCIP 8004 v02.
-- 12/14/07      Changed the name of this MIB only to reflect the version number.
-- 02/13/09      Changed the name of this MIB only to reflect the version number.
--
--               DISTRIBUTION NOTICE
--Copyright 1996 - 2008 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation Engineers
--(ITE), and the National Electrical Manufacturers Association (NEMA). All
--intellectual property rights, including, but not limited to, the rights of
--reproduction in whole or in part in any form, translation into other
--languages and display are reserved by the copyright owners under the laws of
```

--the United States of America, the Universal Copyright Convention, the Berne
--Convention, and the International and Pan American Copyright Conventions.
--Except for the MIB, Do not copy without written permission of either AASHTO,
--ITE, or NEMA.

--
--
--
--
--

Joint NEMA, AASHTO, and ITE
NTCIP Management Information Base
DISTRIBUTION NOTICE

--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:

--

- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "--",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
- (ii) use of the MIB is restricted in that the syntax field may be modified
--only to reflect a more restrictive sub-range or enumerated values;
- (iii) the description field may be modified but only to the extent that:
--(a) only those bit values or enumerated values that are supported are
--listed; and (b) the more restrictive subrange is expressed.

--

--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 OR 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, OR 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.

--

--NTCIP is a trademark of AASHTO/ITE/NEMA.

--

NTCIP1103v0215-SFMP DEFINITIONS ::= BEGIN

IMPORTS

Counter
FROM RFC1155-SMI
OBJECT-TYPE
FROM RFC-1212
application
FROM NTCIP8004-A-2004;

```
;  
-- EXPORTS EVERYTHING  
  
sfmp OBJECT IDENTIFIER ::= {application 2}  
--   <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2  
  
sfmpStatistics OBJECT IDENTIFIER ::= { sfmp 1 }  
--   <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1
```

A.2.1. Number of Incoming SFMP Packets

```
sfmpInPkts OBJECT-TYPE  
    SYNTAX Counter  
    ACCESS read-only  
    STATUS mandatory  
    DESCRIPTION  
        "<Definition> The total number of Messages delivered to  
        the SFMP entity for processing.  
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.1  
        "  
 ::= { sfmpStatistics 1 }
```

A.2.2. Number of Outgoing SFMP Packets

```
sfmpOutPkts OBJECT-TYPE  
    SYNTAX Counter  
    ACCESS read-only  
    STATUS mandatory  
    DESCRIPTION  
        "<Definition> The total number of SFMP PDU's which were  
        generated by the SFMP protocol entity.  
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.2  
        "  
 ::= { sfmpStatistics 2 }
```

A.2.3. Number of Incoming SFMP Packets with Bad Version Numbers

```
sfmpInBadVersions OBJECT-TYPE  
    SYNTAX Counter  
    ACCESS read-only  
    STATUS mandatory  
    DESCRIPTION  
        "<Definition> The total number of SFMP Messages which  
        were delivered to the SFMP protocol entity and were for  
        an unsupported SFMP version.  
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.3  
        "  
 ::= { sfmpStatistics 3 }
```

A.2.4. Number of Incoming SFMP Packets with Bad Community Names

```
sfmpInBadCommunityNames OBJECT-TYPE  
    SYNTAX Counter  
    ACCESS read-only  
    STATUS mandatory  
    DESCRIPTION  
        "<Definition> The total number of SFMP Messages delivered  
        to the SFMP protocol entity which used a SFMP community  
        name not known to said entity.
```

```
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.4
    "
 ::= { sfmpStatistics 4 }
```

A.2.5. Number of Incoming SFMP Packets with Bad Use of a Community Name

```
sfmpInBadCommunityUses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP Messages delivered
        to the SFMP protocol entity which represented an SFMP
        operation which was not allowed by the SFMP community
        named in the Message.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.5
    "
 ::= { sfmpStatistics 5 }
```

A.2.6. Number of Incoming SFMP Packets with Parsing Errors

```
sfmpInParseErrs OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of OER errors encountered
        by the SFMP protocol entity when decoding received SFMP
        Messages.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.6
    "
 ::= { sfmpStatistics 6 }
```

A.2.7. Reserved

```
-- node 7 is reserved for bad types in order to parallel SNMP, but it does not
-- apply to SFMP
```

A.2.8. Number of Incoming SFMP Packets indicating a Too Big Error

```
sfmpInTooBigS OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDUs which were
        delivered to the SFMP protocol entity with a Message
        Type of Error and Error Number of tooBig.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.8
    "
 ::= { sfmpStatistics 8 }
```

A.2.9. Number of Incoming SFMP Packets indicating a No Such Name Error

```
sfmpInNoSuchNames OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDUs which were
```

```
        delivered to the SFMP protocol entity with a Message
        Type of Error and Error Number of noSuchName.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.9
    "
 ::= { sfmpStatistics 9 }
```

A.2.10. Number of Incoming SFMP Packets indicating a Bad Value Error

```
sfmpInBadValues OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDUs which were
        delivered to the SFMP protocol entity with a Message
        Type of Error and Error Number of badValue.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.10
    "
 ::= { sfmpStatistics 10 }
```

A.2.11. Number of Incoming SFMP Packets indicating a Read-Only Error

```
sfmpInReadOnlys OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDUs which were
        delivered to the SFMP protocol entity with a Message Type
        of Error and Error Number of readOnly.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.11
    "
 ::= { sfmpStatistics 11 }
```

A.2.12. Number of Incoming SFMP Packets indicating a General Error

```
sfmpInGenErrs OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDUs which were
        delivered to the SFMP protocol entity with a Message Type
        of Error and Error Number of genError.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.12
    "
 ::= { sfmpStatistics 12 }
```

A.2.13. Reserved

```
-- node 13 is reserved for total request vars in order to
-- parallel SNMP, but it does not apply to SFMP
```

A.2.14. Reserved

```
-- node 14 is reserved for total set vars in order to parallel
-- SNMP, but it does not apply to SFMP
```

A.2.15. Number of Incoming SFMP Get Requests

```
sfmpInGetRequests OBJECT-TYPE
```

```
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "<Definition> The total number of SFMP Get-Request PDUs
  which have been accepted and processed by the SFMP
  protocol entity.
  <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.15
  "
::= { sfmpStatistics 15 }
```

A.2.16. Reserved

```
-- node 16 is reserved for in get nexts in order to parallel
-- SNMP, but it does not apply to SFMP
```

A.2.17. Number of Incoming SFMP Set Requests

```
sfmpInSetRequests OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "<Definition> The total number of SFMP Set-Request PDUs
    which have been accepted and processed by the SFMP protocol
    entity.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.17
    "
::= { sfmpStatistics 17 }
```

A.2.18. Number of Incoming SFMP Get Responses

```
sfmpInGetResponses OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "<Definition> The total number of SFMP Get-Response PDUs
    which have been accepted and processed by the SFMP protocol
    entity.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.18
    "
::= { sfmpStatistics 18 }
```

A.2.19. Reserved

```
-- node 19 is reserved for traps to parallel SNMP, but it
-- does not apply to SFMP at present
```

A.2.20. Number of Outgoing SFMP Packets indicating a Too Big Error

```
sfmpOutTooBig OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "<Definition> The total number of SFMP PDUs which were
    generated by the SFMP protocol entity with a Message Type
    of Error and Error Number of tooBig.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.20
    "
```

```
"  
 ::= { sfmpStatistics 20 }
```

A.2.21. Number of Outgoing SFMP Packets indicating a No Such Name Error

```
sfmpOutNoSuchNames OBJECT-TYPE  
 SYNTAX Counter  
 ACCESS read-only  
 STATUS mandatory  
 DESCRIPTION  
   "<Definition> The total number of SFMP PDUs which were  
   generated by the SFMP protocol entity with a Message Type  
   of Error and Error Number of noSuchname.  
   <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.21  
   "  
 ::= { sfmpStatistics 21 }
```

A.2.22. Number of Outgoing SFMP Packets indicating a Bad Value Error

```
sfmpOutBadValues OBJECT-TYPE  
 SYNTAX Counter  
 ACCESS read-only  
 STATUS mandatory  
 DESCRIPTION  
   "<Definition> The total number of SFMP PDUs which were  
   generated by the SFMP protocol entity with a Message Type  
   of Error and Error Number of badValue.  
   <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.22  
   "  
 ::= { sfmpStatistics 22 }
```

A.2.23. Number of Outgoing SFMP Packets indicating a Read-Only Error

```
sfmpOutReadOnly OBJECT-TYPE  
 SYNTAX Counter  
 ACCESS read-only  
 STATUS mandatory  
 DESCRIPTION  
   "<Definition> The total number of SFMP PDUs which were  
   generated by the SFMP protocol entity with a Message Type  
   of Error and Error Number of readOnly.  
   <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.23  
   "  
 ::= { sfmpStatistics 23 }
```

A.2.24. Number of Outgoing SFMP Packets indicating a General Error

```
sfmpOutGenError OBJECT-TYPE  
 SYNTAX Counter  
 ACCESS read-only  
 STATUS mandatory  
 DESCRIPTION  
   "<Definition> The total number of SFMP PDUs which were  
   generated by the SFMP protocol entity with a Message Type  
   of Error and Error Number of genErr.  
   <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.24  
   "  
 ::= { sfmpStatistics 24 }
```

A.2.25. Number of Outgoing SFMP Get Requests

```
sfmpOutGetRequests OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDU's with a Message
        Type of Get-Request, which have been generated by the SFMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.25
        "
 ::= { sfmpStatistics 25 }
```

A.2.26. Reserved

```
-- node 26 is reserved for out get nexts in order to parallel SNMP,
-- but it does not apply to SFMP
```

A.2.27. Number of Outgoing SFMP Set Requests

```
sfmpOutSetRequests OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDU's with a Message
        Type of Set-Request, which have been generated by the SFMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.27
        "
 ::= { sfmpStatistics 27 }
```

A.2.28. Number of Outgoing SFMP Get Responses

```
sfmpOutGetResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDU's with a Message
        Type of Get-Response, which have been generated by the SFMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.28
        "
 ::= { sfmpStatistics 28 }
```

A.2.29. Number of Outgoing SFMP Trap Messages

```
sfmpOutTrapMessages OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition>The total number of SFMP PDUs with a message
        type of Trap that have been generated by the SFMP protocol
        entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.29
        "
 ::= {sfmpStatistics 29}
```

A.2.30. Reserved

-- node 30 is reserved for enable authentication traps to parallel
-- SNMP, but it does not apply to SFMP

A.2.31. Number of Incoming SFMP Set Requests – No Replies

```
sfmpInSetRequestsNoReply OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP Set-Request No Reply
        PDUs which have been accepted and processed by the SFMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.31
        "
 ::= { sfmpStatistics 31 }
```

A.2.32. Number of Incoming SFMP Set Responses

```
sfmpInSetResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP Set-Response PDUs
        which have been accepted and processed by the SFMP protocol
        entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.32
        "
 ::= { sfmpStatistics 32 }
```

A.2.33. Number of Incoming SFMP Error Responses

```
sfmpInErrorResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP Error-Response PDUs
        which have been accepted and processed by the SFMP protocol
        entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.33
        "
 ::= { sfmpStatistics 33 }
```

A.2.34. Number of Outgoing SFMP Set Requests – No Replies

```
sfmpOutSetRequestsNoReply OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDU's with a Message
        Type of Set-Request-No-Reply, which have been generated by
        the SFMP protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.34
        "
 ::= { sfmpStatistics 34 }
```

```
::= { sfmpStatistics 34 }
```

A.2.35. Number of Outgoing SFMP Set Responses

```
sfmpOutSetResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDU's with a Message
        Type of Set-Response, which have been generated by the SFMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.35
        "
::= { sfmpStatistics 35 }
```

A.2.36. Number of Outgoing SFMP Error Responses

```
sfmpOutErrorResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of SFMP PDU's with a Message
        Type of Error-Response, which have been generated by the SFMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.2.1.36
        "
::= { sfmpStatistics 36 }
```

```
END -- NTCIP1103v0215-SFMP
```

A.3. TMP STMP MIB Header

```
-- Filename:      1103v0215-STMP.MIB
-- Description:   This MIB defines various objects related to managing and
-- monitoring the Simple Transportation Management Protocol (STMP).
-- Specifically, these include objects related to:
-- (a) configuration of dynamic objects,
-- (b) communication statistics,
-- (c) configuration of community names,
-- (d) managing event information that can be logged in the device, and
-- (e) the mapping of logical names to network addresses
-- This MIB replaces portions of TMIB-II.
--
-- MIB Revision History:
-- 08/01/96      Original standard approved
-- 01/01/98      Preliminary Release of TS 3.2 TMIB MIB formatted for 80
--                columns and no TABs
-- 01/07/98      Replaced some missed TABs with spaces
-- 07/08/98      Added Copyright Notice
-- 10/07/98      Amendment 1
-- 03/09/00      Removed all the special edits to that were done to use the
--                SMIC Compiler
--                Defined DisplayString to eliminate reference to RFC 1212
--                END is still left "Dynamic Object Data" group
--                Changed filename and updated copyright years
--                Updated the MIB to Amendment 1
-- 08/09/00      Modified header format and wording of copyright and MIB
-- 11/16/01      Added objects for sfmp and stmp statistics
```

-- Moved security node into this MIB from NTCIP 1201
-- Added objects to support logical names
-- Renamed the module to NTCIP1103-A-2002 from TMIB-II
-- Renamed the text name to Transportation Management
-- Protocols MIB from Transportation MIB
-- Changed STATUS of all objects to optional to reflect new
-- conformance rules being defined in NTCIP 8004
-- 06/08/04 Moved report node into this MIB from NTCIP 1201
-- 09/27/04 Changed name of file.
-- 10/11/04 Per KLV e-mails 10/08/04 updated version and
-- Changed Index on logicalNameTranslation-index
-- from (0..255) to (1..255)
-- Changed all STATUS optional to mandatory
-- 06/13/05 Updated filename.
-- 08/09/05 Added object definitions for Trap Management, Watch Block
-- Objects
-- Updated filename; re-instantiated the OIDs for the
-- 13 dynamic objects.
-- 04/19/06 Broke the various logical groupings of objects into separate
-- MIBs to allow for separate compiling for various deployment
-- needs.
-- 10/09/07 Changed the Description fields of the objects to conform to the
-- new version of NTCIP 8004 v02.
-- 12/14/07 Changed the name of this MIB only to reflect the version number.
-- 02/13/09 Changed the name of this MIB only to reflect the version number.

-- DISTRIBUTION NOTICE

--Copyright 1996 - 2008 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation Engineers
--(ITE), and the National Electrical Manufacturers Association (NEMA). All
--intellectual property rights, including, but not limited to, the rights of
--reproduction in whole or in part in any form, translation into other
--languages and display are reserved by the copyright owners under the laws of
--the United States of America, the Universal Copyright Convention, the Berne
--Convention, and the International and Pan American Copyright Conventions.
--Except for the MIB, Do not copy without written permission of either AASHTO,
--ITE, or NEMA.

-- Joint NEMA, AASHTO, and ITE
-- NTCIP Management Information Base
-- DISTRIBUTION NOTICE

--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:

-- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "-",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
--(ii) use of the MIB is restricted in that the syntax field may be modified
--only to reflect a more restrictive sub-range or enumerated values;
--(iii) the description field may be modified but only to the extent that:
--(a) only those bit values or enumerated values that are supported are
--listed; and (b) the more restrictive subrange is expressed.

```
--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 OR 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, OR 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.
--
--NTCIP is a trademark of AASHTO/ITE/NEMA.
--
*****
```

```
NTCIP1103v0215-STMP DEFINITIONS ::= BEGIN
```

```
IMPORTS
    null
        FROM RFC1155-SMI
    OBJECT-TYPE
        FROM RFC-1212
    OwnerString, dynObjMgmt
        FROM NTCIP8004-A-2004;
-- EXPORTS EVERYTHING
```

A.3.1. Type Definitions

```
ConfigEntryStatus ::= INTEGER
    { valid (1),
      underCreation (2),
      invalid (3) }
-- See Clause 5.2.4.1 of NTCIP 1103 for the complete definition
-- of this Type.
```

A.3.2. Objects for STMP

```
dynObjData OBJECT IDENTIFIER ::= { dynObjMgmt 2 }
-- <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.2
```

A.3.3. Maximum Dynamic Object Table Entries

```
dynObjDefTableMaxEntries OBJECT-TYPE
    SYNTAX INTEGER (1..255)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> This object specifies the maximum number of
        rows that may be implemented in the Dynamic Object
```

```
        Definition table.
        <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.4
        "
 ::= { dynObjMgmt 4 }
```

A.3.4. Dynamic Object Definition Table

```
dynObjDef OBJECT-TYPE
    SYNTAX SEQUENCE OF DynObjEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "<Definition> A list of objects to be included in dynamic
        objects
        <TableType> static
        <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.1
        "
 ::= { dynObjMgmt 1 }
```

```
dynObjEntry OBJECT-TYPE
    SYNTAX DynObjEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "<Definition> A list of OBJECT IDENTIFIERS that make up a
        dynamic object
        <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.1.1
        "
    INDEX { dynObjNumber, dynObjIndex }
 ::= { dynObjDef 1 }
```

```
DynObjEntry ::= SEQUENCE {
    dynObjNumber INTEGER (1..13),
    dynObjIndex INTEGER,
    dynObjVariable OBJECT IDENTIFIER }
-- dynObjOwner & dynObjStatus were deprecated from the
-- DynObjEntry structure. See Annex B.
```

A.3.4.1.1. Dynamic Object Number

```
dynObjNumber OBJECT-TYPE
    SYNTAX INTEGER ( 1..13)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The dynamic object number that this entry is
        to be associated with.
        <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.1.1.1
        "
 ::= { dynObjEntry 1 }
```

A.3.4.1.2. Dynamic Object Index

```
dynObjIndex OBJECT-TYPE
    SYNTAX INTEGER (1..255)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> 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 determines the order in which objects are transmitted for the associated dynamic object. The lower dynObjIndex numbers are transmitted before larger numbers for entries within the same dynamic object.

<Object Definition> 1.3.6.1.4.1.1206.4.1.3.1.1.2
"

::= { dynObjEntry 2 }

A.3.4.1.3. Dynamic Object Variable

dynObjVariable OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-write
STATUS mandatory
DESCRIPTION

"<Definition> The complete object identifier of the particular variable to be included in the specified dynamic object number. Care must be taken when defining dynamic objects so that the maximum size of all the objects included in a dynamic object do not exceed the maximum packet size of the communications network.

When set to reference a columnar object, an agent may wish to only validate the prefix portion of the object identifier. The suffix or index portion of an object identifier need not be instantiated or exist at the time a dynObjVariable is defined.

This object shall not reference any of the objects identified in NTCIP 1103 Clause 8.2.

This object may not be modified unless the associated dynObjConfigStatus object is equal to underCreation.

<Object Definition> 1.3.6.1.4.1.1206.4.1.3.1.1.3

"

DEFVAL {null}

::= { dynObjEntry 3 }

A.3.4.1.4. Reserved

-- { dynObjEntry 4 } is a deprecated node that was a columnar object
-- of the DynObjEntry sequence. See Annex B.

A.3.4.1.5. Reserved

-- { dynObjEntry 5 } is a deprecated node that was a columnar object
-- of the DynObjEntry sequence. See Annex B.

A.3.5. Dynamic Object Data

-- { dynObjData 1 } is deprecated. See Annex B.

-- { dynObjData 2 } is deprecated. See Annex B.

-- { dynObjData 3 } is deprecated. See Annex B.

-- { dynObjData 4 } is deprecated. See Annex B.

```
-- { dynObjData 5 } is deprecated.  See Annex B.  
-- { dynObjData 6 } is deprecated.  See Annex B.  
-- { dynObjData 7 } is deprecated.  See Annex B.  
-- { dynObjData 8 } is deprecated.  See Annex B.  
-- { dynObjData 9 } is deprecated.  See Annex B.  
-- { dynObjData 10 } is deprecated.  See Annex B.  
-- { dynObjData 11 } is deprecated.  See Annex B.  
-- { dynObjData 12 } is deprecated.  See Annex B.  
-- { dynObjData 13 } is deprecated.  See Annex B.
```

A.3.6. Dynamic Object Configuration

```
dynObjConfigTable OBJECT-TYPE  
    SYNTAX SEQUENCE OF DynObjConfigEntry  
    ACCESS not-accessible  
    STATUS mandatory  
    DESCRIPTION  
        "<Definition> A table consisting of an owner and status for  
        each of the 13 dynamic object definitions.  
        <TableType> static  
        <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.3  
        "  
    ::= { dynObjMgmt 3}
```

```
dynObjConfigEntry OBJECT-TYPE  
    SYNTAX DynObjConfigEntry  
    ACCESS not-accessible  
    STATUS mandatory  
    DESCRIPTION  
        "<Definition> A table consisting of an owner and status for  
        each of the 13 dynamic object definitions.  
        <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.3.1  
        "  
    INDEX {dynObjNumber}  
    ::= {dynObjConfigTable 1}
```

```
DynObjConfigEntry ::= SEQUENCE {  
    dynObjConfigOwner OwnerString,  
    dynObjConfigStatus ConfigEntryStatus }
```

A.3.6.1.1. Dynamic Object Configuration Owner

```
dynObjConfigOwner OBJECT-TYPE  
    SYNTAX OwnerString  
    ACCESS read-write  
    STATUS mandatory  
    DESCRIPTION  
        "<Definition> The entity that configured the associated  
        dynamic object.  This object may not be modified unless  
        the associated dynObjConfigStatus object is equal to  
        underCreation.
```

```
    <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.3.1.1
    "
    DEFVAL {""}
 ::= {dynObjConfigEntry 1}
```

A.3.6.1.2. Dynamic Object Configuration Status

```
dynObjConfigStatus OBJECT-TYPE
    SYNTAX ConfigEntryStatus
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "<Definition> Indicates the state of the associated dynamic
        object. Depending on the validity checks that are performed
        on the dynamic object definition, a set request may or may
        not be honored. See Clause 5.2.4.1 for a complete
        description.
        <Object Defintion> 1.3.6.1.4.1.1206.4.1.3.3.1.2
        "
 ::= {dynObjConfigEntry 2}

END -- NTCIP1103v0215-STMP
```

A.4. TMP STMP-Statistics MIB Header

```
-- Filename:      1103v0215-STMP-Stats.MIB
-- Description:   This MIB defines various objects related to
--               monitoring the Simple Transportation Management Protocol
--               (STMP).
--               Specifically, these include objects related to:
--               (a) communication statistics,
--               This MIB replaces portions of TMIB-II.
--
-- MIB Revision History:
-- 08/01/96      Original standard approved
-- 01/01/98      Preliminary Release of TS 3.2 TMIB MIB formatted for 80
--               columns and no TABs
-- 01/07/98      Replaced some missed TABs with spaces
-- 07/08/98      Added Copyright Notice
-- 10/07/98      Amendment 1
-- 03/09/00      Removed all the special edits to that were done to use the
--               SMIC Compiler
--               Defined DisplayString to eliminate reference to RFC 1212
--               END is still left "Dynamic Object Data" group
--               Changed filename and updated copyright years
--               Updated the MIB to Amendment 1
-- 08/09/00      Modified header format and wording of copyright and MIB
-- 11/16/01      Added objects for sfmp and stmp statistics
--               Moved security node into this MIB from NTCIP 1201
--               Added objects to support logical names
--               Renamed the module to NTCIP1103-A-2002 from TMIB-II
--               Renamed the text name to Transportation Management
--               Protocols MIB from Transportation MIB
--               Changed STATUS of all objects to optional to reflect new
--               conformance rules being defined in NTCIP 8004
-- 06/08/04      Moved report node into this MIB from NTCIP 1201
-- 09/27/04      Changed name of file.
-- 10/11/04      Per KLV e-mails 10/08/04 updated version and
--               Changed Index on logicalNameTranslation-index
```

-- from (0..255) to (1..255)
-- Changed all STATUS optional to mandatory
-- 06/13/05 Updated filename.
-- 08/09/05 Added object definitions for Trap Management, Watch Block
Objects
-- Updated filename; re-instantiated the OIDs for the
-- 13 dynamic objects.
-- 04/19/06 Broke the various logical groupings of objects into separate
-- MIBs to allow for separate compiling for various deployment
-- needs.
-- 10/09/07 Changed the Description fields of the objects to conform to the
-- new version of NTCIP 8004 v2.
-- 12/14/07 Changed the name of this MIB only to reflect the version number.
-- 02/13/09 Changed the name of this MIB only to reflect the version number.
--

-- DISTRIBUTION NOTICE

--Copyright 1996 - 2008 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation Engineers
--(ITE), and the National Electrical Manufacturers Association (NEMA). All
--intellectual property rights, including, but not limited to, the rights of
--reproduction in whole or in part in any form, translation into other
--languages and display are reserved by the copyright owners under the laws of
--the United States of America, the Universal Copyright Convention, the Berne
--Convention, and the International and Pan American Copyright Conventions.
--Except for the MIB, Do not copy without written permission of either AASHTO,
--ITE, or NEMA.
--

-- Joint NEMA, AASHTO, and ITE
-- NTCIP Management Information Base
-- DISTRIBUTION NOTICE
--

--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:
--

-- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "-",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
--(ii) use of the MIB is restricted in that the syntax field may be modified
--only to reflect a more restrictive sub-range or enumerated values;
--(iii) the description field may be modified but only to the extent that:
--(a) only those bit values or enumerated values that are supported are
--listed; and (b) the more restrictive subrange is expressed.
--

--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 OR 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, OR 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.
--
--NTCIP is a trademark of AASHTO/ITE/NEMA.
--*****
```

```
NTCIP1103v0215-STMP-Stats DEFINITIONS ::= BEGIN
```

```
IMPORTS
    Counter
        FROM RFC1155-SMI
    OBJECT-TYPE
        FROM RFC-1212
    application
        FROM NTCIP8004-A-2004;
-- EXPORTS EVERYTHING
```

A.4.1. STMP Statistics

```
stmp          OBJECT IDENTIFIER ::= { application 3 }
--           <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3

stmpStatistics OBJECT IDENTIFIER ::= { stmp 1 }
--           <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1
```

A.4.1.1.1. Number of Incoming STMP Packets

```
stmpInPkts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of Messages delivered to the
        STMP entity for processing.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.1
        "
    ::= { stmpStatistics 1 }
```

A.4.1.1.2. Number of Outgoing STMP Packets

```
stmpOutPkts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDU's which were
        generated by the STMP protocol entity .
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.2
        "
    ::= { stmpStatistics 2 }
```

A.4.1.1.3. Reserved

-- node 3 is reserved for bad version in order to parallel SNMP,
-- but it does not apply to STMP

A.4.1.1.4. Reserved

-- node 4 is reserved for bad community name in order to parallel
-- SNMP, but it does not apply to STMP

A.4.1.1.5. Reserved

-- node 5 is reserved for bad community use in order to parallel
-- SNMP, but it does not apply to STMP

A.4.1.1.6. Number of Incoming STMP Packets with Parsing Errors

```
stmpInParseErrs OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of OER errors encountered by
        the STMP protocol entity when decoding received STMP Messages.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.6
        "
 ::= { stmpStatistics 6 }
```

A.4.1.1.7. Reserved

-- node 7 is reserved for bad types in order to parallel SNMP, but
-- it does not apply to STMP

A.4.1.1.8. Number of Incoming STMP Packets indicating a Too Big Error

```
stmpInTooBigs OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        delivered to the STMP protocol entity with a Message Type
        of Error and Error Number of tooBig.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.8
        "
 ::= { stmpStatistics 8 }
```

A.4.1.1.9. Number of Incoming STMP Packets indicating a No Such Name Error

```
stmpInNoSuchNames OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        delivered to the STMP protocol entity with a Message Type
        of Error and Error Number of noSuchName.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.9
        "
 ::= { stmpStatistics 9 }
```

A.4.1.1.10. Number of Incoming STMP Packets indicating a Bad Value Error

```
stmpInBadValues OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        delivered to the STMP protocol entity with a Message
        Type of Error and Error Number of badValue.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.10
        "
 ::= { stmpStatistics 10 }
```

A.4.1.1.11. Number of Incoming STMP Packets indicating a Read-Only Error

```
stmpInReadOnlys OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        delivered to the STMP protocol entity with a Message
        Type of Error and Error Number of readOnly.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.11
        "
 ::= { stmpStatistics 11 }
```

A.4.1.1.12. Number of Incoming STMP Packets indicating a General Error

```
stmpInGenErrs OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        delivered to the STMP protocol entity with a Message
        Type of Error and Error Number of genError.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.12
        "
 ::= { stmpStatistics 12 }
```

A.4.1.1.13. Reserved

```
-- node 13 is reserved for total request vars in order to parallel
-- SNMP, but it does not apply to STMP
```

A.4.1.1.14. Reserved

```
-- node 14 is reserved for total set vars in order to parallel SNMP,
-- but it does not apply to STMP
```

A.4.1.1.15. Number of Incoming STMP Get Requests

```
stmpInGetRequests OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP Get-Request PDUs which
        have been accepted and processed by the STMP protocol entity.
```

```
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.15
  "
 ::= { stmpStatistics 15 }
```

A.4.1.1.16. Number of Incoming STMP Get Next Requests

```
stmpInGetNexts OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "<Definition> The total number of STMP Get-Next PDUs which
    have been accepted and processed by the STMP protocol entity.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.16
  "
 ::= { stmpStatistics 16 }
```

A.4.1.1.17. Number of Incoming STMP Set Requests

```
stmpInSetRequests OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "<Definition> The total number of STMP Set-Request PDUs which
    have been accepted and processed by the STMP protocol entity.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.17
  "
 ::= { stmpStatistics 17 }
```

A.4.1.1.18. Number of Incoming STMP Get Responses

```
stmpInGetResponses OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "<Definition> The total number of STMP Get-Response PDUs which
    have been accepted and processed by the STMP protocol entity.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.18
  "
 ::= { stmpStatistics 18 }
```

A.4.1.1.19. Reserved

```
-- node 19 is reserved for in trap responses in order to parallel
-- SNMP, but it does not apply to STMP
```

A.4.1.1.20. Number of Outgoing STMP Packets indicating a Too Big Error

```
stmpOutTooBigs OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "<Definition> The total number of STMP PDUs which were
    generated by the STMP protocol entity with a Message Type
    of Error and Error Number of tooBig.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.20
  "

```

```
::= { stmpStatistics 20 }
```

A.4.1.1.21. Number of Outgoing STMP Packets indicating a No Such Name Error

```
stmpOutNoSuchNames OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        generated by the STMP protocol entity with a Message Type
        of Error and Error Number of noSuchName.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.21
        "
::= { stmpStatistics 21 }
```

A.4.1.1.22. Number of Outgoing STMP Packets indicating a Bad Value Error

```
stmpOutBadValues OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        generated by the STMP protocol entity with a Message Type
        of Error and Error Number of badValue.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.22
        "
::= { stmpStatistics 22 }
```

A.4.1.1.23. Number of Outgoing STMP Packets indicating a Read-Only Error

```
stmpOutReadOnly OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        generated by the STMP protocol entity with a Message Type
        of Error and Error Number of readOnly.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.23
        "
::= { stmpStatistics 23 }
```

A.4.1.1.24. Number of Outgoing STMP Packets indicating a General Error

```
stmpOutGenError OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDUs which were
        generated by the STMP protocol entity with a Message Type
        of Error and Error Number of genErr.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.24
        "
::= { stmpStatistics 24 }
```

A.4.1.1.25. Number of Outgoing STMP Get Requests

```
stmpOutGetRequests OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDU's with a Message
        Type of Get-Request, which have been generated by the STMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.25
        "
 ::= { stmpStatistics 25 }
```

A.4.1.1.26. Number of Outgoing STMP Get Next Requests

```
stmpOutGetNexts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDU's with a Message
        Type of Get-Next, which have been generated by the STMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.26
        "
 ::= { stmpStatistics 26 }
```

A.4.1.1.27. Number of Outgoing STMP Set Requests

```
stmpOutSetRequests OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDU's with a Message
        Type of Set-Request, which have been generated by the STMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.27
        "
 ::= { stmpStatistics 27 }
```

A.4.1.1.28. Number of Outgoing STMP Get Responses

```
stmpOutGetResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDU's with a Message
        Type of Get-Response, which have been generated by the STMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.28
        "
 ::= { stmpStatistics 28 }
```

A.4.1.1.29. Reserved

```
-- node 29 is reserved for in trap responses in order to parallel
-- SNMP, but it does not apply to STMP
```

A.4.1.1.30. Reserved

-- node 30 is reserved for enable authentication traps to parallel
-- SNMP, but it does not apply to STMP

A.4.1.1.31. Number of Incoming STMP Set Request – No Replies

```
stmpInSetRequestsNoReply OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP Set-Request No Reply
        PDUs which have been accepted and processed by the STMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.31
        "
 ::= { stmpStatistics 31 }
```

A.4.1.1.32. Number of Incoming STMP Set Responses

```
stmpInSetResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP Set-Response PDUs
        which have been accepted and processed by the STMP protocol
        entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.32
        "
 ::= { stmpStatistics 32 }
```

A.4.1.1.33. Number of Incoming STMP Error Responses

```
stmpInErrorResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP Error-Response PDUs
        which have been accepted and processed by the STMP protocol
        entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.33
        "
 ::= { stmpStatistics 33 }
```

A.4.1.1.34. Number of Outgoing STMP Set Request – No Replies

```
stmpOutSetRequestsNoReply OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDU's with a Message
        Type of Set-Request-No-Reply, which have been generated by
        the STMP protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.34
        "
 ::= { stmpStatistics 34 }
```

```
::= { stmpStatistics 34 }
```

A.4.1.1.35. Number of Outgoing STMP Set Responses

```
stmpOutSetResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDU's with a Message
        Type of Set-Response, which have been generated by the STMP
        protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.35
        "
::= { stmpStatistics 35 }
```

A.4.1.1.36. Number of Outgoing STMP Error Responses

```
stmpOutErrorResponses OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "<Definition> The total number of STMP PDU's with a Message
        Type of Error-Response, which have been generated by the
        STMP protocol entity.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.3.1.36
        "
::= { stmpStatistics 36 }
```

```
END -- NTCIP1103v0215-STMP-Stats
```

A.5. TMP STMP Configuration MIB Header

```
-- Filename:      1103v0215-STMP-Config.MIB
-- Description:   This MIB defines various objects related to configuring
--               the Simple Transportation Management Protocol (STMP).
--               Specifically, these include objects related to:
--               (a) configuration of objects,
--               This MIB replaces portions of TMIB-II.
--
-- MIB Revision History:
-- 08/01/96      Original standard approved
-- 01/01/98      Preliminary Release of TS 3.2 TMIB MIB formatted for 80
--               columns and no TABS
-- 01/07/98      Replaced some missed TABS with spaces
-- 07/08/98      Added Copyright Notice
-- 10/07/98      Amendment 1
-- 03/09/00      Removed all the special edits to that were done to use the
--               SMIC Compiler
--               Defined DisplayString to eliminate reference to RFC 1212
--               END is still left "Dynamic Object Data" group
--               Changed filename and updated copyright years
--               Updated the MIB to Amendment 1
-- 08/09/00      Modified header format and wording of copyright and MIB
-- 11/16/01      Added objects for sfmp and stmp statistics
--               Moved security node into this MIB from NTCIP 1201
--               Added objects to support logical names
--               Renamed the module to NTCIP1103-A-2002 from TMIB-II
--               Renamed the text name to Transportation Management
```

```
--          Protocols MIB from Transportation MIB
--          Changed STATUS of all objects to optional to reflect new
--          conformance rules being defined in NTCIP 8004
-- 06/08/04  Moved report node into this MIB from NTCIP 1201
-- 09/27/04  Changed name of file.
-- 10/11/04  Per KLV e-mails 10/08/04 updated version and
--          Changed Index on logicalNameTranslation-index
--          from (0..255) to (1..255)
--          Changed all STATUS optional to mandatory
-- 06/13/05  Updated filename.
-- 08/09/05  Added object definitions for Trap Management, Watch Block
Objects
--          Updated filename; re-instantiated the OIDs for the
--          13 dynamic objects.
-- 04/19/06  Broke the various logical groupings of objects into separate
--          MIBs to allow for separate compiling for various deployment
--          needs.
-- 10/09/07  Changed the Description fields of the objects to conform to the
--          new version of NTCIP 8004 v02.
-- 12/14/07  Changed the name of this MIB only to reflect the version number.
-- 02/13/09  Changed the name of this MIB only to reflect the version number.
--
--          DISTRIBUTION NOTICE
--Copyright 1996 - 2008 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation Engineers
--(ITE), and the National Electrical Manufacturers Association (NEMA). All
--intellectual property rights, including, but not limited to, the rights of
--reproduction in whole or in part in any form, translation into other
--languages and display are reserved by the copyright owners under the laws of
--the United States of America, the Universal Copyright Convention, the Berne
--Convention, and the International and Pan American Copyright Conventions.
--Except for the MIB, Do not copy without written permission of either AASHTO,
--ITE, or NEMA.
--
--          Joint NEMA, AASHTO, and ITE
--          NTCIP Management Information Base
--          DISTRIBUTION NOTICE
--
--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:
--
-- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "--",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
--(ii) use of the MIB is restricted in that the syntax field may be modified
--only to reflect a more restrictive sub-range or enumerated values;
--(iii) the description field may be modified but only to the extent that:
--(a) only those bit values or enumerated values that are supported are
--listed; and (b) the more restrictive subrange is expressed.
--
--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 OR 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, OR 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.
--
--NTCIP is a trademark of AASHTO/ITE/NEMA.
--*****
```

```
NTCIP1103v0215-STMP-Config DEFINITIONS ::= BEGIN
```

```
IMPORTS
    OBJECT-TYPE
        FROM RFC-1212
    profiles
        FROM NTCIP8004-A-2004;
-- EXPORTS EVERYTHING
```

A.5.1. STMP Configuration

```
profilesSTMP OBJECT IDENTIFIER ::= { profiles 2 }
--      <Object Identifier> 1.3.6.1.4.1.1206.4.1.2.2
```

-- This node is an identifier used to group all objects for support
-- of configuration functions that are common to device types that
-- support the STMP protocol. The objects under this node are placed
-- under the Protocols\Profiles\STMP subtree within the NEMA node.

A.5.1.1.1. Dynamic Object Persistence

```
dynamicObjectPersistence OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "<Definition> The maximum power outage time in minutes that
        may occur before all STMP dynamic object definitions in a
        device shall be invalidated. If this object is set to zero
        then the existing dynamic object definitions shall be
        invalidated on device power up. If this object is set to
        its maximum value (65535) then the existing dynamic object
        definitions shall nominally persist for an infinite period
        (in practice this will be limited by the non-volatile memory
        capabilities of the device) This object shall not be
        invalidated due to power outages of any duration. A device
        that supports STMP dynamic objects shall support this object.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.2.2.1
        "
```

```
DEFVAL {65535}  
::= { profilesSTMP 1 }
```

A.5.1.1.2. Dynamic Object Configuration ID

```
dynamicObjectTableConfigID 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 the current  
    values stored in the dynObjVariable and dynObjConfigOwner  
    objects for all dynamic objects with a dynObjStatus of valid.  
    This value shall be calculated on the change of any  
    dynObjStatus to or from the valid state. This value reported  
    by this object shall not change unless there has been a  
    change in the 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 configuration of dynamic objects by monitoring  
    this value after it has established a baseline.  
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.2.2.2  
    "  
::= { profilesSTMP 2 }  
  
END --NTCIP1103v0215-STMP-Config
```

A.6. TMP LogicalNames MIB Header

```
-- Filename:      1103v0215-LogicalNames.MIB  
-- Description:   This MIB defines various objects related to mapping  
--               between logical device names and network addresses.  
--               Specifically, these include objects related to:  
--               (a) the mapping of logical names to network addresses  
--               This MIB replaces portions of TMIB-II.  
-- MIB Revision History:  
-- 08/01/96      Original standard approved  
-- 01/01/98      Preliminary Release of TS 3.2 TMIB MIB formatted for 80  
--               columns and no TABs  
-- 01/07/98      Replaced some missed TABs with spaces  
-- 07/08/98      Added Copyright Notice  
-- 10/07/98      Amendment 1  
-- 03/09/00      Removed all the special edits to that were done to use the  
--               SMIC Compiler  
--               Defined DisplayString to eliminate reference to RFC 1212  
--               END is still left "Dynamic Object Data" group  
--               Changed filename and updated copyright years  
--               Updated the MIB to Amendment 1  
-- 08/09/00      Modified header format and wording of copyright and MIB  
-- 11/16/01      Added objects for sfmp and stmp statistics  
--               Moved security node into this MIB from NTCIP 1201  
--               Added objects to support logical names  
--               Renamed the module to NTCIP1103-A-2002 from TMIB-II  
--               Renamed the text name to Transportation Management  
--               Protocols MIB from Transportation MIB  
--               Changed STATUS of all objects to optional to reflect new  
--               conformance rules being defined in NTCIP 8004
```

-- 06/08/04 Moved report node into this MIB from NTCIP 1201
-- 09/27/04 Changed name of file.
-- 10/11/04 Per KLV e-mails 10/08/04 updated version and
-- Changed Index on logicalNameTranslation-index
-- from (0..255) to (1..255)
-- Changed all STATUS optional to mandatory
-- 06/13/05 Updated filename.
-- 06/22/05 Commented DEFVAL in logicalNameTranslationNetworkAddress to get
-- around incorrect syntax error. "DEFVAL {0}" for logicalName-
-- TranslationNetworkAddress is correct.
-- 08/09/05 Added object definitions for Trap Management, Watch Block
-- Objects
-- Updated filename; re-instantiated the OIDs for the
-- 13 dynamic objects.
-- 04/19/06 Broke the various logical groupings of objects into separate
-- MIBs to allow for separate compiling for various deployment
-- needs.
-- 10/09/07 Changed the Description fields of the objects to conform to the
-- new version of NTCIP 8004 v02.
-- 10/25/07 Corrected DEFVAL of logicalNameTranslationLogicalName
-- 12/14/07 Changed the name of this MIB only to reflect the version number.
-- 02/13/09 Changed the name of this MIB only to reflect the version number.

--
-- DISTRIBUTION NOTICE

--Copyright 1996 - 2008 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation Engineers
--(ITE), and the National Electrical Manufacturers Association (NEMA). All
--intellectual property rights, including, but not limited to, the rights of
--reproduction in whole or in part in any form, translation into other
--languages and display are reserved by the copyright owners under the laws of
--the United States of America, the Universal Copyright Convention, the Berne
--Convention, and the International and Pan American Copyright Conventions.
--Except for the MIB, Do not copy without written permission of either AASHTO,
--ITE, or NEMA.

--
-- Joint NEMA, AASHTO, and ITE
-- NTCIP Management Information Base
-- DISTRIBUTION NOTICE
--

--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:

--
-- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "-",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
--(ii) use of the MIB is restricted in that the syntax field may be modified
--only to reflect a more restrictive sub-range or enumerated values;
--(iii) the description field may be modified but only to the extent that:
--(a) only those bit values or enumerated values that are supported are
--listed; and (b) the more restrictive subrange is expressed.

--
--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 OR 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, OR 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.
--
--NTCIP is a trademark of AASHTO/ITE/NEMA.
--*****
```

```
NTCIP1103v0215-LogicalNames DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
    NetworkAddress
        FROM RFC1155-SMI
    OBJECT-TYPE
        FROM RFC-1212
    RowStatusStatic, application
        FROM NTCIP8004-A-2004;
```

```
-- EXPORTS EVERYTHING
```

```
logicalNames OBJECT IDENTIFIER ::= { application 4 }
--         <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.4
-- This node is used to define objects to allow a mapping between
-- logical device names and network addresses.
```

A.6.1. Maximum Logical Name Translations

```
logicalNameTranslationTableMaxEntries OBJECT-TYPE
    SYNTAX  INTEGER (1..255)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "<Definition> This object specifies the maximum number of
        rows that may be implemented in the logical name translation
        table.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.4.1
        "
::= { logicalNames 1 }
```

A.6.2. Logical Name Translation Table

```
logicalNameTranslationTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF LogicalNameTranslationEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "<Definition> This table defines the logical names of the
        other network entities with which the device may communicate
```

```
    and maps these names to the network addresses of those
    devices.
    <TableType> static
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.4.2"
 ::= { logicalNames 2 }
```

```
logicalNameTranslationEntry OBJECT-TYPE
  SYNTAX LogicalNameTranslationEntry
  ACCESS not-accessible
  STATUS mandatory
  DESCRIPTION
    "<Definition> This is one logical row of the logical name
    translation table.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.4.2.1
    "
  INDEX { logicalNameTranslationIndex }
 ::= { logicalNameTranslationTable 1 }
```

```
LogicalNameTranslationEntry ::= SEQUENCE {
  logicalNameTranslationIndex      INTEGER,
  logicalNameTranslationLogicalName OCTET STRING,
  logicalNameTranslationNetworkAddress NetworkAddress,
  logicalNameTranslationStatus     RowStatusStatic }
```

A.6.2.1.1. Index for the Logical Name Translation

```
logicalNameTranslationIndex OBJECT-TYPE
  SYNTAX INTEGER (1..255)
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "<Definition> This object provides the index into the
    logical name table.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.4.2.1.1"
 ::= { logicalNameTranslationEntry 1 }
```

A.6.2.1.2. Logical Name for the Logical Name Translation

```
logicalNameTranslationLogicalName OBJECT-TYPE
  SYNTAX OCTET STRING (SIZE (0..32))
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "<Definition> This object defines the logical name of the
    network entity for which this row is defined.
    <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.4.2.1.2"
  DEFVAL { "" }
 ::= { logicalNameTranslationEntry 2 }
```

A.6.2.1.3. Network Address of the Logical Name Translation

```
logicalNameTranslationNetworkAddress OBJECT-TYPE
  SYNTAX NetworkAddress
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "<Definition> This object defines the network address of the
    associated network entity for the given profile. If the
    transport profile is 'internet,' the network address is the
    IP address of the entity stored as an IpAddress. If the
```

```
transport profile is 't2,' there is no physical network
address, but the entity is assigned a dummy IP address in
order to abstract the mapping to the ipNetToMediaTable
defined in MIB-II.
<Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.4.2.1.3"
-- DEFVAL { 0 } See 06/22/05 Comment
::= { logicalNameTranslationEntry 3 }
```

A.6.2.1.4. Logical Name Translation Status

```
logicalNameTranslationStatus OBJECT-TYPE
    SYNTAX RowStatusStatic
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "<Definition> This object allows for the management of rows
        within the table.
        <Object Identifier> 1.3.6.1.4.1.1206.4.1.1.7.4.2.1.4"
    DEFVAL { invalid }
    ::= { logicalNameTranslationEntry 4 }

END -- NTCIP1103v0215-LogicalNames
```

A.7. TMP REPORT MIB Header

```
-- Filename: 1103v0215-Report.MIB
-- Description: This MIB defines various objects related to managing
-- event information for the purpose of logging data within the
-- device.
-- Specifically, these include objects related to:
-- (a) managing event information that can be logged in the
-- device
-- This MIB replaces portions of TMIB-II.
--
-- MIB Revision History:
-- 08/01/96 Original standard approved
-- 01/01/98 Preliminary Release of TS 3.2 TMIB MIB formatted for 80
-- columns and no TABS
-- 01/07/98 Replaced some missed TABS with spaces
-- 07/08/98 Added Copyright Notice
-- 10/07/98 Amendment 1
-- 03/09/00 Removed all the special edits to that were done to use the
-- SMIC Compiler
-- Defined DisplayString to eliminate reference to RFC 1212
-- END is still left "Dynamic Object Data" group
-- Changed filename and updated copyright years
-- Updated the MIB to Amendment 1
-- 08/09/00 Modified header format and wording of copyright and MIB
-- 11/16/01 Added objects for sfmp and stmp statistics
-- Moved security node into this MIB from NTCIP 1201
-- Added objects to support logical names
-- Renamed the module to NTCIP1103-A-2002 from TMIB-II
-- Renamed the text name to Transportation Management
-- Protocols MIB from Transportation MIB
-- Changed STATUS of all objects to optional to reflect new
-- conformance rules being defined in NTCIP 8004
-- 06/08/04 Moved report node into this MIB from NTCIP 1201
-- 09/27/04 Changed name of file.
```

-- 10/11/04 Per KLV e-mails 10/08/04 updated version and
-- Changed Index on logicalNameTranslation-index
-- from (0..255) to (1..255)
-- Changed all STATUS optional to mandatory
-- 06/13/05 Updated filename.
-- 08/09/05 Added object definitions for Trap Management, Watch Block
Objects
-- Updated filename; re-instantiated the OIDs for the
-- 13 dynamic objects.
-- 04/19/06 Broke the various logical groupings of objects into separate
-- MIBs to allow for separate compiling for various deployment
-- needs.
-- 12/14/07 Changed the name of this MIB only to reflect the version number.
-- 02/13/09 Changed the name of this MIB only to reflect the version number.
--

-- DISTRIBUTION NOTICE

--Copyright 1996 - 2008 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation Engineers
--(ITE), and the National Electrical Manufacturers Association (NEMA). All
--intellectual property rights, including, but not limited to, the rights of
--reproduction in whole or in part in any form, translation into other
--languages and display are reserved by the copyright owners under the laws of
--the United States of America, the Universal Copyright Convention, the Berne
--Convention, and the International and Pan American Copyright Conventions.
--Except for the MIB, Do not copy without written permission of either AASHTO,
--ITE, or NEMA.
--

-- Joint NEMA, AASHTO, and ITE
-- NTCIP Management Information Base
-- DISTRIBUTION NOTICE
--

--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:
--

-- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "-",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
--(ii) use of the MIB is restricted in that the syntax field may be modified
--only to reflect a more restrictive sub-range or enumerated values;
--(iii) the description field may be modified but only to the extent that:
--(a) only those bit values or enumerated values that are supported are
--listed; and (b) the more restrictive subrange is expressed.
--

--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 OR 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, OR 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.
--
--NTCIP is a trademark of AASHTO/ITE/NEMA.
--*****
```

```
NTCIP1103v0215-Report DEFINITIONS ::= BEGIN
```

```
IMPORTS
    Counter, Opaque, null
        FROM RFC1155-SMI
    OBJECT-TYPE
        FROM RFC-1212
    global
        FROM NTCIP8004-A-2004;
-- EXPORTS EVERYTHING
```

A.7.1. Report Parameter Node

```
globalReport OBJECT IDENTIFIER ::= { global 4 }
--          <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4

-- 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.
```

A.7.2. 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.
        <Unit>EventClass
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.5
        "
 ::= { globalReport 5 }
```

A.7.3. 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.
  <TableType> static
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.6
  "
 ::= { globalReport 6 }

eventClassEntry OBJECT-TYPE
  SYNTAX EventClassEntry
  ACCESS not-accessible
  STATUS mandatory
  DESCRIPTION
    "<Definition>This defines a row in the Event Class Table
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.6.1
    "
  INDEX { eventClassNumber }
 ::= { eventClassTable 1 }

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

```

A.7.3.1.1. 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.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.6.1.1
    "
 ::= { eventClassEntry 1 }

```

A.7.3.1.2. 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.
    The event cannot be logged if the eventClass has an
    eventClassLimit of zero (0).
    "

```

```
        <Unit>Event
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.6.1.2
    "
 ::= { eventClassEntry 2 }
```

A.7.3.1.3. 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.
        <Unit>second
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.6.1.3
    "
    DEFVAL {0}
 ::= { eventClassEntry 3 }
```

A.7.3.1.4. 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.
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.6.1.4
    "
 ::= { eventClassEntry 4 }
```

A.7.3.1.5. 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.
        <Unit>Event
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.6.1.5
    "
 ::= { eventClassEntry 5 }
```

A.7.3.1.6. 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.
        <Unit>Events
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.6.1.6
    "
 ::= { eventClassEntry 6 }

```

A.7.4. 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.
        <Unit>EventType
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.1
    "
 ::= { globalReport 1}

```

A.7.5. 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.
        <TableType> static
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2
    "
 ::= { 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.
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1
    "
    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 }

```

A.7.5.1.1. 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.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.1
    "
 ::= { eventLogConfigEntry 1 }
```

A.7.5.1.2. 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.

      NOTE--The event cannot be logged if the EventClass has an
      eventClassLimit of zero (0).
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.2
    "
  DEFVAL {1}
 ::= { eventLogConfigEntry 2 }
```

A.7.5.1.3. 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.

```
<Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.3
"
  DEFVAL {onChange}
 ::= { eventLogConfigEntry 3 }
```

A.7.5.1.4. Event Log Configuration Compare Value Parameter

eventConfigCompareValue OBJECT-TYPE

```
SYNTAX    INTEGER
ACCESS    read-write
STATUS    mandatory
DESCRIPTION
  "<Definition>This object contains the comparison 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).
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.4
  "
  DEFVAL    {0}
::= { eventLogConfigEntry 4 }
```

A.7.5.1.5. 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, smallerThanValue, or andedWithValue, this
  object specifies the time (in tenth of seconds, +1 tenth / -0
  tenths) for which the samples used for comparison must be
  true prior to the event condition becoming true. If the
  eventConfigMode is set to onChange or periodic, the value of
  this object shall be ignored.

  The amount of time the condition must be true is measured in
  tenths of a second. The accuracy of this timer is limited to
  +1 tenth of a second and -0 tenths of a second. If the event
  is true for at least the time shown in this parameter +1
  tenth of a second, the condition shall trigger a log entry.
  It is recognized that some designs only sample the condition
  periodically, in which case the condition must be true for
  at least the time indicated by this object before the event
  becomes true and the event shall always become true if the
  condition is true for a duration equal to the value shown in
  this object plus 1 tenth of a second.
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.5
  "
  DEFVAL    {0}
::= { eventLogConfigEntry 5 }
```

A.7.5.1.6. 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.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.6

"

DEFVAL {null}

::= { eventLogConfigEntry 6 }

A.7.5.1.7. 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 }

<Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.7

"

DEFVAL {null}

::= { eventLogConfigEntry 7 }

A.7.5.1.8. 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.

other - indicates that the action is other than defined in this standard.

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.

<Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.8

"

DEFVAL {disabled}

```
::= { eventLogConfigEntry 8 }
```

A.7.5.1.9. 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
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.2.1.9
    "
::= { eventLogConfigEntry 9 }
```

A.7.6. 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.
    <Unit>Event
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.3
    "
::= { globalReport 3 }
```

A.7.7. 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.
    <TableType> dynamic
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.4
    "
::= { globalReport 4 }
```

```
eventLogEntry OBJECT-TYPE
  SYNTAX EventLogEntry
```

```
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
  "<Definition>This object defines an entry in the event log
  table
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.4.1
  "
INDEX { eventLogClass, eventLogNumber }
::= { eventLogTable 1 }

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

A.7.7.1.1. 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.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.4.1.1
    "
  ::= { eventLogEntry 1 }
```

A.7.7.1.2. 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.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.4.1.2
    "
  ::= { eventLogEntry 2 }
```

A.7.7.1.3. 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 responsible
```

```
        for this event entry.  
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.4.1.3  
    "  
 ::= { eventLogEntry 3 }
```

A.7.7.1.4. 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.  
        <Unit>second  
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.4.1.4  
    "  
 ::= { eventLogEntry 4 }
```

A.7.7.1.5. 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.  
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.4.1.5  
    "  
 ::= { eventLogEntry 5 }
```

A.7.8. 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.
```

```
        <Unit>Events
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.4.7
    "
 ::= { globalReport 7 }

END -- NTCIP1103v0215-Report
```

A.8. TMP Security MIB Header

```
-- Filename:      1103v0215-Security.MIB
-- Description:   This MIB defines various objects related to managing and
--               monitoring the TMP security objects.
--               Specifically, these include objects related to:
--               (a) configuration of community names,
--               This MIB replaces portions of TMIB-II.
--
-- MIB Revision History:
-- 08/01/96      Original standard approved
-- 01/01/98      Preliminary Release of TS 3.2 TMIB MIB formatted for 80
--               columns and no TABs
-- 01/07/98      Replaced some missed TABs with spaces
-- 07/08/98      Added Copyright Notice
-- 10/07/98      Amendment 1
-- 03/09/00      Removed all the special edits to that were done to use the
--               SMIC Compiler
--               Defined DisplayString to eliminate reference to RFC 1212
--               END is still left "Dynamic Object Data" group
--               Changed filename and updated copyright years
--               Updated the MIB to Amendment 1
-- 08/09/00      Modified header format and wording of copyright and MIB
-- 11/16/01      Added objects for sfmp and stmp statistics
--               Moved security node into this MIB from NTCIP 1201
--               Added objects to support logical names
--               Renamed the module to NTCIP1103-A-2002 from TMIB-II
--               Renamed the text name to Transportation Management
--               Protocols MIB from Transportation MIB
--               Changed STATUS of all objects to optional to reflect new
--               conformance rules being defined in NTCIP 8004
-- 06/08/04      Moved report node into this MIB from NTCIP 1201
-- 09/27/04      Changed name of file.
-- 10/11/04      Per KLV e-mails 10/08/04 updated version and
--               Changed Index on logicalNameTranslation-index
--               from (0..255) to (1..255)
--               Changed all STATUS optional to mandatory
-- 06/13/05      Updated filename.
-- 08/09/05      Added object definitions for Trap Management, Watch Block
--               Objects
--               Updated filename; re-instantiated the OIDs for the
--               13 dynamic objects.
-- 04/19/06      Broke the various logical groupings of objects into separate
--               MIBs to allow for separate compiling for various deployment
--               needs.
-- 12/14/07      Changed the name of this MIB only to reflect the version number.
-- 02/13/09      Changed the name of this MIB only to reflect the version number.
--
--               DISTRIBUTION NOTICE
--Copyright 1996 - 2008 by the American Association of State Highway and
--Transportation Officials (AASHTO), the Institute of Transportation Engineers
```

--(ITE), and the National Electrical Manufacturers Association (NEMA). All
--intellectual property rights, including, but not limited to, the rights of
--reproduction in whole or in part in any form, translation into other
--languages and display are reserved by the copyright owners under the laws of
--the United States of America, the Universal Copyright Convention, the Berne
--Convention, and the International and Pan American Copyright Conventions.
--Except for the MIB, Do not copy without written permission of either AASHTO,
--ITE, or NEMA.

--
--
-- Joint NEMA, AASHTO, and ITE
-- NTCIP Management Information Base
-- DISTRIBUTION NOTICE
--

--To the extent and in the limited event these materials are distributed by
--AASHTO/ITE/NEMA in the form of a Management Information Base ("MIB"),
--AASHTO/ITE/NEMA extends the following permissions:

-
- (i) you may make and/or distribute unlimited copies (including derivative
--works) of the MIB, including copies for commercial distribution, provided
--that (a) each copy you make and/or distribute contains this Notice and (b)
--each derivative work of the MIB uses the same module name followed by "--",
--followed by your Internet Assigned Number Authority (IANA)-assigned
--enterprise number;
 - (ii) use of the MIB is restricted in that the syntax field may be modified
--only to reflect a more restrictive sub-range or enumerated values;
 - (iii) the description field may be modified but only to the extent that:
--(a) only those bit values or enumerated values that are supported are
--listed; and (b) the more restrictive subrange is expressed.

--
--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 OR 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, OR 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.

--
--NTCIP is a trademark of AASHTO/ITE/NEMA.
--*****

NTCIP1103v0215-Security DEFINITIONS ::= BEGIN

IMPORTS
 Gauge
 FROM RFC1155-SMI
 OBJECT-TYPE

```
FROM RFC-1212
global
FROM NTCIP8004-A-2004;
-- EXPORTS EVERYTHING
```

```
security OBJECT IDENTIFIER ::= {global 5}
-- <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.5
-- This node is an identifier used to group all objects related to the
-- assignment of community names and the access rights they provide.
-- These objects were previously defined in NTCIP 1201, but were moved
-- here as they relate to the protocols more than the end application.
```

A.8.1. 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.
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.5.1
  "
  DEFVAL { "administrator" }
  ::= { security 1 }
```

A.8.2. 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.
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.5.2
  "
  ::= { security 2 }
```

A.8.3. 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.
  <TableType> static
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.5.3
  "
  ::= { 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.
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.5.3.1
        "
    INDEX { communityNameIndex }
::= { communityNameTable 1 }
```

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

A.8.3.1.1. 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.
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.5.3.1.1
        "
::= { communityNameTableEntry 1 }
```

A.8.3.1.2. 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.
        <Object Identifier> 1.3.6.1.4.1.1206.4.2.6.5.3.1.2
        "
    DEFVAL { "public" }
::= { communityNameTableEntry 2 }
```

A.8.3.1.3. 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.

```
<Object Identifier> 1.3.6.1.4.1.1206.4.2.6.5.3.1.3
"
DEFVAL { 4294967295 }
 ::= { communityNameTableEntry 3 }

END -- NTCIP1103v0215-Security
```

A.9. TMP Trap MIB Header

NOTE THAT TRAPS WILL BE ADDRESSED IN THE NEXT VERSION. THIS SECTION WILL CONTAIN THE OBJECT DEFINITIONS NECESSARY FOR TRAP MANAGEMENT.

Annex B Deprecated Objects (Normative)

Annex B presents textual conventions (i.e., user-defined types) and object-types that were standardized in previous versions of NTCIP, but have since been deprecated. These objects are listed here because they may still be encountered in existing equipment, but they are no longer recommended for new designs.

B.1. Deprecated Type Definitions

```
EntryStatus ::= INTEGER
  { valid (1),
    createRequest (2),
    underCreation (3),
    invalid (4)
  }
-- See Annex E for a complete definition of this Type.
```

B.2. Deprecated Object Types

B.2.1. Dynamic Object Definition

```
dynObjOwner OBJECT-TYPE
  SYNTAX  OwnerString
  ACCESS  read-write
  STATUS  deprecated
  DESCRIPTION
    "This object has been replaced with dynObjConfigOwner.

    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
  ACCESS  read-write
  STATUS  deprecated
  DESCRIPTION
    "This object has been replaced with dynObjConfigStatus.

    The status of this dynamic object definition entry. See description of
    EntryStatus above for restrictions on accesses."
```

```
::= { dynObjEntry 5 }
```

B.2.2. Dynamic Object Data

```
dynObj1 OBJECT-TYPE
  SYNTAX  OCTET STRING
  ACCESS  read-write
  STATUS  deprecated
  DESCRIPTION
    "<Definition> 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 deprecated
DESCRIPTION
"<Definition> 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 deprecated
DESCRIPTION
"<Definition> 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
SYNTAX OCTET STRING
ACCESS read-write
STATUS deprecated
DESCRIPTION
"<Definition> 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 deprecated
DESCRIPTION

"<Definition> 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 deprecated
DESCRIPTION
"<Definition> 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 deprecated
DESCRIPTION
"<Definition> 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 }
```

dynObj8 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS deprecated
DESCRIPTION
"<Definition> 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 deprecated
DESCRIPTION
    "<Definition> 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 deprecated
DESCRIPTION
    "<Definition> 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 deprecated
DESCRIPTION
    "<Definition> 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 noSuchName shall be returned by the agent"
 ::= { dynObjData 11 }

dynObj12 OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS deprecated
DESCRIPTION
    "<Definition> 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 deprecated
DESCRIPTION

"<Definition> 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 }
```

Annex C (Informative) An Explanation of Relative Object Identifiers

A Relative Object Identifier is a new ASN.1 syntax that was defined in order to provide a shorthand reference to nodes on the ISO naming tree.

A node on this tree is formally identified by an Object Identifier. The Object Identifier consists of the series of node numbers encountered as one traverses the ISO naming tree from the root node to the node of interest. This has proven to be a valuable tool in producing globally unique identifiers because an owner of the node can delegate the management of any sub-node to a different entity. Thus, ISO has delegated management of the node {iso(1) org(3) dod(6)} to the US Department of Defense. Likewise, the US Department of Defense has delegated the sub-node {iso(1) org(3) dod(6) internet(1)} to the Internet Assigned Numbers Authority. This delegation of authority continues as needed to meet the needs of the user.

Thus, anyone is able to acquire a node on this tree, from a previously registered authority, and start assigning sub-nodes while having confidence that the identifiers will be globally unique under this scheme. Unfortunately, the node number that one is likely to acquire for free is likely to be several layers down on the tree. For example, NEMA obtained the number {iso(1) org(3) dod(6) internet(1) private(4) enterprises(1) nema(1206)}, and the specific object identifiers of the NTCIP objects are several more layers below this node. As a result, the entire Object Identifier of an NTCIP object is about 15 bytes long.

While this problem may be understandable if NTCIP devices had to support the entire ISO tree, the reality is that virtually every NTCIP object exchanged is likely to be under the nema node of the ISO tree. Thus, every object identifier exchanged across the NTCIP link will begin with a redundant set of seven node numbers.

The ASN.1 community has recognized this problem and has developed the Relative Object Identifier to solve this problem. A Relative Object Identifier is simply the tail end of an Object Identifier, starting at some defined location. It is encoded exactly like an Object Identifier, except that the special encoding rules of the first two sub-nodes do not apply.

NOTE—A normal Object Identifier is encoded with the first two sub-nodes combined into a single sub-identifier. This is possible because the ISO tree only has three root nodes and a limited number of first level sub-nodes such that all possible values can be encoded into a single byte. These conditions do not necessarily apply for other nodes on the ISO tree and therefore these special encoding rules do not apply.

The location of the start point of the Relative Object Identifier is defined by an ASN.1 comment associated with the Relative Object Identifier declaration. In the examples used in NTCIP 1103 v02, the start point is always identified as nema, which equates to {iso(1) org(3) dod(6) internet(1) private(4) enterprises(1) nema(1206)}.

Thus, the *devices* node, which is defined as {iso(1) org(3) dod(6) internet(1) private(4) enterprises(1) nema(1206) transportation(4) devices(2)} could be represented as either:

- a) An Object Identifier with a value of **0x2B 06 01 04 01 89 36 04 02**, or
- b) A Relative Object Identifier from the **nema** node with a value of **0x04 02**.

Annex D (Informative) Entry Status Type

The EntryStatus type, as defined in NTCIP 1101:1996 has been deprecated. Agents conforming to NTCIP 1103 v02 should not implement any objects based on this type as it has been replaced with ConfigEntryStatus. It is defined here because management stations may need to support this type in order to interoperate with agents that were designed with early versions of the NTCIP standards, specifically NTCIP 1101:1996. This type is anticipated to eventually be eliminated from NTCIP standards.

The EntryStatus type was used to manage the initial version of the dynObjDef table as defined in NTCIP 1101:1996. This original table had a separate status and owner for each indexed variable of a dynamic object. The status was intended to serve the same purpose as the ConfigEntryStatus type, but experience indicated that the design of having a different status for each indexed variable created ambiguities as to the status of the overall dynamic object. Experience also suggested a refinement to the structure of the status field operation. As a result, this object was deprecated in the 1998 Amendment. However, it is believed that some agents may exist in the field that still use the original design and therefore the original text that defined this type is provided below for reference.

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. Therefore, managers must be prepared to receive tabular information from agents that corresponds 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 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).

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.

§

< There is no text on this page .>