



# NTCIP SMI

---

<https://ite-org.github.io/NTCIP-8004/latest/>

AASHTO / ITE / NEMA

CC-BY-4.0 (<https://creativecommons.org/licenses/by/4.0/>)

## Table of contents

---

Front Matter	4
	4
Notices	5
Acknowledgements	8
Foreword	9
Introduction	11
General	12
Scope	12
References	12
General Statements	16
Terms	16
Abbreviations	20
Background [Informative]	22
Introduction	22
Management Information Base (MIB) and Module	23
Types of Object Types	24
Document Overview	28
Object Identification [Normative]	29
International Object Identifier Tree	29
Registered Nodes	30
Rules for Module Development [Normative]	33
General	33
Rules for Modules	35
Rules for Values	41
Rules for Macros	45
Additional NTCIP Requirements	106
Object Identifier Layout	113
MIB Design Considerations	115
Requirements for Agent Implementations [Normative]	122
Agent Capabilities	122
Requirements for Non-Standard MIBs	122
Default Values	122
Extensions of Standardized Enumerations	122
Guidelines for Operating Agencies [Informative]	124

NTCIP Structure of Management Information (SMI) [Normative]	125
NEMA Module	125
Transportation Module	126
Conversion From SMIv1 To SMIv2 [Normative]	133
Macro Examples [Informative]	136
Module Identity Macro	136
Object Identity Macro	136
Conceptual Table Object Type	136
Conceptual Row Object Type	137
Row Status Object Type	137
Enumeration	137
Integer32	137
Gauge32	138
Counter32	138
Unsigned32	138
Object Identifier	138
BITS	138
Internet Address	139
Text	139
Textual Convention Macro	139
Block Object Using Identifier and Type	140
Block Object Using Value Reference and a Defined Value	140
Block Object Using Value Dereferencing	141
Notification Type Macro	141
Object Group Macro	142
Notification Group Macro	142
Module Compliance Macro	142
History of Changes [Informative]	143

## Front Matter

---

*Working Group Draft*  
**NTCIP 8004 v3.0.0-a.19**

---

**National Transportation Communications for ITS Protocol**  
**Structure and Identification of Management Information (SMI)**

 June 18, 2025

## Notices

---

### Copyright

---

This document, including the management information base (MIB) defined herein, is licensed under the [Creative Commons Attribution 4.0 International License](#) (CC-BY 4.0) with additional terms and conditions MIB. By accessing, using, or contributing to the document, you agree to be bound by the terms of this Agreement, which includes the CC-BY 4.0 License and the additional provisions herein. By contributing to the document, you grant the NTCIP a perpetual, worldwide, non-exclusive, royalty-free license to use, modify, and incorporate your contributions into the document.

Except for the MIB, you are free to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon the material) for any purpose, even commercially, as long as you give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

### Additional Terms and Conditions

---

In addition to the terms of CC-BY 4.0, the following provisions apply to the use, distribution, and adaptation of the MIB:

#### 1. Additional Definitions

a. "MIB" refers to the Management Information Base specification, including its textual content and structure, developed as part of the standards effort. b. "NTCIP" refers to AASHTO / ITE / NEMA, who maintain and govern the MIB and its associated Object Identifiers (OIDs).

#### 2. Restrictions on OID Modifications

a. Adapted Material may not redefine existing OIDs or define new OIDs under the "nema" node (1.3.6.1.4.1.1206). b. Any proposed modifications or extensions to the MIB that involve OIDs under the "nema" node must be submitted to the NTCIP for review and approval. c. Adapted

Material must clearly indicate that it is not part of the official MIB and must not claim to represent an NTCIP MIB.

### 3. No Warranty

The MIB is provided "as is" without warranty of any kind, express or implied, including but not limited to warranties of merchantability, fitness for a particular purpose, or non-infringement. The NTCIP does not warrant that the MIB will meet your requirements or be error-free.

### 4. Limitation of Liability

In no event shall the NTCIP or its contributors be liable for any direct, indirect, incidental, special, or consequential damages arising out of the use or inability to use the MIB, even if advised of the possibility of such damages.

#### 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 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 project are the trademarks of their respective owners.

 June 18, 2025

## Acknowledgements

---

This document was prepared through an open-source standards development process with the following active contributors:

contributors **1**

Check out the full list of [contributors here](#).

In addition, the following submitted comments during the process:

- k-vaughn

The resultant document is maintained by the NTCIP Open-Source Process (OPS) Working Group (WG), 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 development organizations (SDOs) and provides guidance for NTCIP development.

🕒 June 18, 2025



## Foreword

---

### Overview

---

This document is an NTCIP Open-Source NTCIP Process, Control, and Information Management document provided as Interim for Field Release (IFR).

Open-source documents are developed using the ITS Open-Source Process, as defined in NTCIP 8008. This process provides an open standards development process that accepts issues reported by the community and resolved by peer-reviewed contributions from the community. The open source process concludes with the resultant material being approved by the defined approval process.

IFR documents are approved through a streamlined process focused on the technical experts of the community (e.g., those participating in the open-source development process) rather than through a formal ballot of industry managers.

NTCIP Process, Control, and Information Management documents define the practices and policies used by the NTCIP Joint Committee and its working groups in developing and maintaining NTCIP publications.

This document defines the structure of management information bases (MIBs) within NTCIP standards and is applicable to the NTCIP 1200 series and other NTCIP standards that deal with device data dictionaries.

### Approvals

---

IFRs are peer reviewed within the open-source process with final approval by an associated WG established by the NTCIP Joint Committee.

Approval information is provided within the online environment.

For more information about NTCIP standards, visit the NTCIP Web Site at [www.ntcip.org](http://www.ntcip.org).

### User Comment Instructions

---

Comments can be submitted at any time. In preparation of this NTCIP standards publication, input of users and other interested parties was sought and evaluated.

Comments on open-source projects can be submitted either on the [discussions](#) or [issues](#) tab of the project.

Discussions can be initiated at any time and anyone in the community can respond, all within a public environment. Responses to discussion comments are strictly informative and may not be accurate. Discussion comments can lead to the submittal of issues that need to be resolved to clarify the standard.


Issues can be submitted at any time. Issues are triaged by the project maintainer, who will evaluate their merit, classify them (e.g., as a bug, documentation issue, omission), and in most cases respond to the submitter. Once ready, issues will be available for contributors to volunteer to address. When a volunteer has a proposed solution, it can be submitted to the project and approved in a relatively short period (when compared to the traditional standards approval process). However, updates to the projects are still version controlled so that users can reference a specific version of the project without fear of it changing.

Comments should use the templates provided on the website; otherwise they may be ignored.

---

#### History

For a history of the project, see the projects [releases](#) page.

 June 18, 2025

## Introduction

---

This document specifies a set of rules for organizing, describing, and defining transportation management information to be exchanged between transportation management applications and transportation equipment. This document is based on and extends of RFC 2578, RFC 2579, and RFC 2580.

NTCIP 8004 contains mandatory requirements that are always applicable and optional and conditional requirements, which may be applicable, for the specification of transportation management information.

This document contains four annexes with Annex A and Annex B identified as normative.

The following keywords apply to NTCIP 8004: AASHTO, ITE, NEMA, IEC, NTCIP, SMI, SNMP, process and control standard.

This document only uses metric units.

🕒 June 18, 2025

## Section 1 General

---

### 1.1 Scope

---

This document specifies a set of rules for organizing, describing, and defining transportation management information to be exchanged between transportation management applications and transportation equipment. This document defines the Structure and Identification of Management Information (SMI) used in transportation-related devices. This document is applicable to the NTCIP 1200 series and other NTCIP standards that deal with device data dictionaries.

NOTE—This document relies on widely accepted conventions, generally designated as “SMIv2” and defined in the Internet Architecture Board (IAB) Standard (STD) 58.

### 1.2 References

---

The following documents are referenced by this document. At the time of publication, the editions indicated were valid.

#### 1.2.1 Normative References

---

Normative references contain provisions that, through reference in this text, constitute provisions of this document. 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.

Identifier	Title
IAB STD 58	(RFC 2578, Structure of Management Information Version 2 (SMIv2), 1999; RFC 2579, Textual Conventions for SMIv2, 1999; RFC 2580, Conformance Statements for SMIv2, 1999)
RFC 3416	Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP), 2002
RFC 3419	Textual Conventions for Transport Addresses, 2002
RFC 3584	Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework, Internet Engineering Task Force (IETF), August 2003
RFC 4001	Textual Conventions for Internet Network Addresses, 2005
RFC 4181	Guidelines for Authors and Reviewers of MIB Documents, Internet Engineering Task Force (IETF), September 2005
ITU-T X.208	Open Systems Interconnection Model and Notation – Specification of Abstract Syntax Notation One (ASN.1), 1988 <sup>1</sup>
ITU-T X.680 (ISO/IEC 8824-1:2015)	Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation
ITU-T X.690 (ISO/IEC 8825-1)	Information technology—ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER)
ITU-T X.696 (ISO/IEC 8825-7)	Information technology—ASN.1 encoding rules: Specification of Octet Encoding Rules (OER)
BIPM SI	The International System of Units (SI) – 9th edition (2019), <a href="https://www.bipm.org/documents/20126/41483022/SI-Brochure-9-EN.pdf">https://www.bipm.org/documents/20126/41483022/SI-Brochure-9-EN.pdf</a>

## 1.2.2 Other References

Other references are included to provide a more complete understanding of this document and its relationship to other documents.

### 1.2.2.1 Other Resources for Contributors

This document standardizes and tailors certain aspects of the information contained in open-sourced; however, it is not a complete replacement of that material. If you wish to learn more about open-source development, the following materials may be of interest:

Identifier	Title
IAB STD 8	(RFC 0854: 1983, <i>Telnet Protocol Specification</i> , and RFC 0855: 1983, <i>Telnet Options Specifications</i> ; J. Postel, J. Reynolds)
RFC 2021	<i>Remote Network Monitoring Management Information Base Version 2 using SMIv2</i> , 1997
RFC 2287	<i>Definitions of System-level Managed Objects for Applications</i> , 1998
RFC 2856	<i>Textual Conventions for Additional High Capacity Data Types</i> , 2000
RFC 2981	<i>Event MIB</i> , 2000
RFC 2982	<i>Distributed Management Expression MIB</i> , 2000
RFC 3014	<i>Notification Log MIB</i> , 2000
RFC 3231	<i>Definitions of Managed Objects for Scheduling Management Operations</i> , 2002
RFC 3411	<i>An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks</i> , 2002
RFC 3433	<i>Entity Sensor Management Information Base</i> , 2002
RFC 3877	<i>Alarm Management Information Base (MIB)</i> , 2001
RFC 4268	<i>Entity State MIB</i> , 2005
RFC 6933	<i>Entity MIB (Version 4)</i> , 2013
AASHTO / ITE / NEMA NTCIP 1201 v03	<i>Global Object (GO) Definitions—version 03</i> , published March 2011
AASHTO / ITE / NEMA NTCIP 1202 v03	<i>Object Definitions for Actuated Signal Controllers (ASC) Interface—version 03A</i> , published May 2019
AASHTO / ITE / NEMA NTCIP 8005 v02	<i>Procedures for Creating Management Information Base (MIB) Files</i> , Proposed
ISO 26048-1:2025 <sup>2</sup>	<i>Intelligent transport systems – Roadside modules <u>SNMP</u> data interface – Part 1: Global Objects</i>
ISO/IEC 9834-1:2012	<i>Information technology – Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree – Part 1:</i>
ISO/TS 20684-2:2021	<i>Intelligent transport systems – Roadside modules <u>SNMP</u> data interface – Part 2: Generalized field device basic management</i>
ISO/WD 20684-3	<i>Intelligent transport systems – Roadside modules <u>SNMP</u> data interface – Part 3: Triggers</i>
ISO/WD 20684-4	<i>Intelligent transport systems – Roadside modules <u>SNMP</u> data interface – Part 4: Notifications</i>
ISO/WD 20684-5	<i>Intelligent transport systems – Roadside modules <u>SNMP</u> data interface – Part 5: Logs</i>
ITU-T X.660 (07/2011)	<i>Identical to ISO/IEC 9834-1:2012. Recommended status.</i>
ISO/WD 20684-6	<i>Intelligent transport systems – Roadside modules <u>SNMP</u> data interface – Part 6: Commands</i>
ISO/WD 20684-7	<i>Intelligent transport systems – Roadside modules <u>SNMP</u> data interface – Part 7: Support Features</i>
NEMA TS_2-2016	<i>Traffic Controller Assemblies with NTCIP Requirements</i> , 2016
	<i>IETF</i> , Generic and Common Textual Conventions (TCs), <a href="https://trac.ietf.org/trac/ops/wiki/mib-common-tcs">https://trac.ietf.org/trac/ops/wiki/mib-common-tcs</a> , accessed June 10, 2021
	Perkins, D; McGinnis, E., <i>Understanding <u>SNMP</u> MIBs</i> , Prentice-Hall, Inc., 1997, ISBN 0-13-437708-7
	Stallings, William, <i><u>SNMP</u>, <u>SNMPv2</u>, and CMIP: The Practical Guide to Network-Management Standards</i> , Massachusetts, Addison-Wesley Publishing Company, 1993, ISBN 0-201-63331-0

Identifier	Title
	Larmouth, John, <a href="#">ASN.1 Complete</a> , Academic Press, a Harcourt Science and Technology Company, May 1999, ISBN 0-12233-435-3, <a href="http://www.oss.com/asn1/booksintro.html">http://www.oss.com/asn1/booksintro.html</a> (October 9, 2000)
	Dubuisson, Olivier, <a href="#">ASN.1 Communication between Heterogeneous Systems</a> , June 5, 2000, ISBN 0-12-6333361-0, <a href="http://www.oss.com/asn1/bookintro.html">http://www.oss.com/asn1/bookintro.html</a> (October 9, 2000)
	Booch, Grady, Rumbaugh, James, Jacobson, Ivar, Unified Modeling Language User Guide, September 30, 1998, ISBN 0-20157-168-4
	<a href="#">UML basics: An introduction to the Unified Modeling Language</a> , <a href="http://www-106.ibm.com/developerworks/rational/library/769.html#N10090">www-106.ibm.com/developerworks/rational/library/769.html#N10090</a> .

### 1.2.3 Contact Information

#### 1.2.3.1 Internet Documents

Obtain Request for Comment (RFC) electronic documents from several repositories online at:

[www.rfc-editor.org](http://www.rfc-editor.org)

[www.rfc-editor.org/repositories.html](http://www.rfc-editor.org/repositories.html)

#### 1.2.3.2 Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT)

The Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT) may be viewed online at:

<http://arc-it.net>

#### 1.2.3.3 ISO, IEC, and ISO/IEC Standards

ISO, IEC, and ISO/IEC standards can be purchased on-line in electronic format or printed copy from:

Techstreet  
6300 Interfirst Dr.  
Ann Arbor, MI 48108  
(800) 699-9277  
[www.techstreet.com](http://www.techstreet.com)

#### 1.2.3.4 IEEE Standards

IEEE standards can be purchased on-line in electronic format or printed copy from:

Techstreet  
6300 Interfirst Dr.  
Ann Arbor, MI 48108  
(800) 699-9277  
[www.techstreet.com](http://www.techstreet.com)

#### 1.2.3.5 NTCIP Standards

Copies of NTCIP standards may be obtained from:

NTCIP Coordinator  
National Electrical Manufacturers Association  
1300 N.17th Street, Suite 900  
Rosslyn, Virginia 22209-3801  
[www.ntcip.org](http://www.ntcip.org)  
e-mail: [ntcip@nema.org](mailto:ntcip@nema.org)

Draft amendments, which are under discussion by the relevant NTCIP Working Group, and amendments recommended by the NTCIP Joint Committee are available.

### 1.3 General Statements

---

The remainder of this document is broken into the following chapters:

- **Background:** Describes the purpose of this document
- **Object Identification:** Discusses the object identifier tree and defines the nodes related to the NTCIP effort.
- **Rules for Module Development:** Defines the rules for developing MIB modules within the NTCIP standards efforts, including the specialized fields used by the NTCIP.
- **Requirements for Agent Implementations:** Requirements for implementations of agents that claim conformance to NTCIP standards.
- **Guidelines for Agencies:** Advice for agencies on operating NTCIP devices.
- **NTCIP Structure of Management Information:** The formal MIB module that defines the structure of the NTCIP portion of the object identifier tree and key textual conventions.
- **Conversion from SMIv1 to SMIv2:** Rules for converting SNMPv1 MIBs to support SNMPv3.
- **Macro Examples:** Examples of applying the rules defined in this document.
- **History of Changes:** A list of changes made with the current version from the last major version.

### 1.4 Terms

---

For the purposes of this document, the following terms and definitions apply. Terms not defined here are used in accordance with their definitions in ISO/IEC/IEEE 24765. Words not defined



here or in ISO/IEC/IEEE 24765 are used in accordance with their definitions in *Webster's New Collegiate Dictionary*.

Term	Definition
<b>agent</b>	The entity that receives requests and transmits responses to the received requests.
<b>block object type</b>	An <u>object type</u> that represents a standardized data structure of elemental <u>object types</u> .
<b>compatibility</b>	The ability of two or more systems or components to exchange information.  [ISO/IEC/IEEE 24765]
<b>configurable object</b>	An <u>object type</u> that represents a data structure that can be configured at run-time to include elemental <u>object types</u> .
<b>context</b>	An instance of a management information base (MIB).  NOTE—A single SNMP <u>agent</u> (e.g., a unique UDP/IP address) typically represents a single <u>context</u> , but might support multiple contexts if it, for example, serves as the interface for multiple field devices (e.g., two dynamic message signs controlled by a single controller) or serves as a proxy <u>agent</u> for multiple field devices. When there are multiple contexts, there might be multiple <u>object</u> instances with the same <u>object</u> identifier and the <u>context</u> is used to disambiguate.
<b>current</b>	Reflecting the conditions at the present time (or at the time at which the data is time stamped) as determined by the Controller. Within the <u>context</u> of a status value, indicates the most up-to-date design of the element (e.g., <u>user need</u> , <u>requirement</u> , <u>object</u> ).
<b>data</b>	Information before it is interpreted.
<b>datagram</b>	A self-contained unit of data transmitted independently of other datagrams.
<b>deprecated</b>	A status value that indicates the <u>user need</u> , <u>requirement</u> , <u>dialog</u> , or <u>object</u> is valid in limited circumstances, but has been replaced by another.  NOTE—This definition is modified from “Understanding SNMP MIBS.” Procurements can require support for deprecated objects to enable multi-version <u>interoperability</u> (e.g., <u>backward compatibility</u> ) with legacy implementations.
<b>file</b>	A grouping of data into a single sequence of bytes that can be referred to by <u>file</u> operations.  NOTE—A <u>file</u> exists nominally in a directory, and can have an associated path.
<b>intelligent transportation systems (ITS)</b>	The application of advanced information processing and communications, sensing, and control technologies to surface transportation with the objective of promoting more efficient use of the existing highway and transportation network, increasing safety and mobility, and decreasing environmental impacts.  NOTE—Also known as “intelligent transport systems”
<b>International Organization for Standardization (ISO)</b>	An international standards organization.  NOTE—ANSI is the primary interface to ISO within the United States. Often thought to be International Standards Organization, because of its acronym ISO.
<b>internet</b>	A large collection of connected networks, primarily in the United States, running the Internet suite of protocols.  NOTE—Sometimes referred to as the <i>DARPA Internet</i> , <i>NSF/DARPA Internet</i> , or the <i>Federal Research Internet</i> .
<b>Interchangeable</b>	A condition which exists when two or more items possess such functional and physical characteristics as to be equivalent in performance and durability, and are capable of being exchanged one for the other without alteration of the items themselves, or adjoining items, except for adjustment, and without selection for fit and performance.  Note: See National Telecommunications and Information Administration, U.S. Department of Commerce
<b>interoperability</b>	Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Term	Definition
	[ISO/IEC/IEEE 24765]
<b>leaf object type</b>	An <u>object type</u> that does not have any sub-identifiers assigned (other than for <u>object</u> instances)
<b>management information base (MIB)</b>	A structured collection of managed information contained within a <u>SNMP context</u> . A MIB represents the instantiated data objects defined by one or more MIB modules and integrated into a device or management station.
<b>Management Station</b>	The computer system with which the device communicates. Typically, the management station commands and monitors the device.
<b>MIB module</b>	A sequence of <u>NVT ASCII</u> characters conforming to X.208 Abstract Syntax Notation One ( <u>ASN.1</u> ) and IAB STD 58. MIB modules are used to define the data objects that are to make up the instantiated objects of the MIB.  NOTE—A <u>MIB module</u> can be, and within NTCIP typically is, presented as both part of a standard and as a stand-alone, computer-readable text file.
<b>National Transportation Communications for ITS Protocol (NTCIP)</b>	A family of protocols that provide common control and data collection services as well as accommodating various system topologies and data routing duties.  NOTE—NTCIP is designed to support not only currently deployed systems, but also new systems and technologies as they become available.
<b>network</b>	A collection of devices connected by intermediate systems and populated by end systems.
<b>notification</b>	An unsolicited event message issued by an <u>SNMP agent</u> for a management station with an expectation of an acknowledgement.
<b>object</b>	An entity identified by a node on the international <u>object</u> identifier tree.  NOTE – See <u>object instance</u> and <u>object type</u> , both of which are types of objects, along with any other entity that can be identified, such as a standard. This document attempts to use the most precise term within its text to assist in distinguishing among "object type" or "object instance". The use of the term " <u>object</u> " is typically limited to registering items on the international <u>object</u> identifier tree.
<b>OBJECT IDENTIFIER</b>	A unique name (identifier) that is associated with each type of <u>object</u> in a MIB that is a defined <u>ASN.1</u> type.
<b>object instance</b>	An instance of an <u>object type</u> , which is an actual instance of data.  NOTE – This document avoids using the term " <u>object</u> " to describe this concept to prevent any ambiguity with an <u>object</u> referenced by the international <u>object</u> identifier tree or an <u>object type</u> .
<b>object type</b>	An abstract specification for a specific piece of data that can be instantiated within a device (zero or many times depending on how it is specified). It specifies the data using the <u>OBJECT-TYPE</u> macro.
<b>OBJECT-TYPE</b>	An <u>X.208 ASN.1</u> macro used to define the meta-attributes of an <u>object type</u> in an <u>SNMP MIB module</u> .
<b>obsolete</b>	A status value that indicates the definition is no longer valid, was found to be flawed, was redundant, or was not useful.  NOTE—In the next (or some future) edition of a standard, any element (e.g., <u>requirement</u> , <u>object</u> ) with a status of " <u>obsolete</u> " can be removed. This definition is modified from "Understanding <u>SNMP MIBS</u> ."
<b>protocol</b>	A specific set of rules, procedures, and conventions defining the format and timing of data transmissions between devices that are accepted and used to understand each other.
<b>Requirement</b>	A <u>requirement</u> describes a condition or capability to which a system shall conform; either derived directly from <u>user needs</u> , or stated in a contract, standard, specification, or other normative document. A desired feature, property, or behavior of a system.
<b>Requirements Traceability</b>	The ability to follow or study the logical progression among the needs, requirements, and design details in a step-by-step fashion.


Term	Definition
<b>Simple Network Management Protocol (SNMP)</b>	A communications protocol developed by the Internet Engineering Task Force (IETF), used for configuration and monitoring of network devices.
<b>Trap</b>	An unsolicited event message issued by an SNMP agent for a management station without any expectation of an acknowledgement.
<b>User</b>	A person using the system that is developed.
<b>User Need</b>	The business or operational problem (opportunity) that is to be fulfilled to justify procurement or use.  Note: While this is termed a “user need” within the NTCIP community, it reflects needs of all stakeholders.

## 1.5 Abbreviations

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

<b>ASCII</b>	<b>American National Standard Code for Information Interchange</b>
<b>ASN.1</b>	Abstract Syntax Notation One, as defined by either ITU-T X.208 or ITU-T X.680 (ISO/IEC 8824-1) (within the context used, both apply)
<b>BER</b>	Basic Encoding Rules, as defined by ITU-T X.690 (ISO/IEC 8825-1)
<b>IAB STD</b>	Internet Architecture Board Standard
<b>IANA</b>	Internet Assigned Numbers Authority
<b>IEC</b>	International Electrotechnical Commission
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>IETF</b>	Internet Engineering Task Force
<b>IP</b>	Internet Protocol
<b>ITU-T</b>	International Telecommunication Union – Telecommunication Standardization Sector
<b>MVI</b>	Multi-Version Interoperability (formerly Backward Compatibility)
<b>NVT</b>	Network Virtual Terminal
<b>OER</b>	Octet Encoding Rules, as defined by ITU-T X.696 (ISO/IEC 8825-7)
<b>OSI</b>	Open Systems Interconnection
<b>PDU</b>	Protocol Data Unit
<b>PRL</b>	Protocol Requirements List
<b>RFC</b>	Request for Comments
<b>SMI</b>	Structure and Identification of Management Information
<b>SNMP</b>	Simple Network Management Protocol version 3
<b>TCP</b>	Transport Control Protocol
<b>UML</b>	Unified Modeling Language
<b>X.208 ASN.1</b>	Abstract Syntax Notation One, as defined by ITU-T X.208
<b>X.680 ASN.1</b>	Abstract Syntax Notation One, as defined by ITU-T X.680 (ISO/IEC 8824-1)

- 
- <sup>1</sup>. [IAB STD 58](#) is still based on [ITU-T X.208](#), which is now withdrawn, but defines the fundamental concept of a macro and [MIB module](#) as used for [SNMP](#). ←
  - <sup>2</sup>. See the [ISO maintenance portal for ISO 26048-1](#) for maintained electronic files. ←

 June 18, 2025

## Section 2 Background [Informative]

---

### 2.1 Introduction

---

Within NTCIP, transportation equipment is managed primarily using the Simple Network Management Protocol (SNMP) to exchange transportation management information with transportation management applications (e.g., within a central system). This type of exchange is commonly called Center-to-Field (C2F) communications. Within SNMP terminology, the transportation equipment includes an SNMP agent that responds to requests received from a transportation management application, which is known as an SNMP management application (or manager). SNMP requests and responses share a similar structure, which can be depicted in the Unified Modeling Language (UML), as shown in Figure 1 (with slight simplifications for the purpose of this discussion).

```

---
title: SNMP Message
---
classDiagram
    class Message {
        msgSecurityParameters
        msgVersion
    }
    class HeaderData {
        msgFlags
        msgID
        msgMaxSize
        msgSecurityModel
    }
    class ScopedPDU {
        contextEngineID
        contextName
        data
    }
    class PDU {
        error-index
        error-status
        request-id
    }
    class VarBind {
        name
        value
    }
    Message *-- "1" HeaderData : msgGlobalData
    Message *-- "1" ScopedPDU : msgData
    ScopedPDU *-- "1" PDU : data
    PDU *-- "1..*" VarBind : variable-bindings
    note for ScopedPDU "Might be encrypted"

```

Starting in the upper-left corner of Figure 1, a message is the logical SNMP structure that is serialized as a data packet (i.e., byte-stream) and exchanged by lower-layer protocol entities (e.g., TCP/IP). It consists of a variety of information, such as protocol header, version information, security parameters, and a potentially encrypted scoped protocol data unit (PDU). The scoped PDU includes the context for the message, error information, the request identifier, and a series of variable bindings represented as a name/value ordered pair. Within this structure, each variable binding (i.e., name/value ordered pair) is the serialized representation of an object

instance stored within the identified SNMP context. An object instance is an instance of a defined object type, which is a formal abstract definition of a piece of data.

For example (in reverse order), NTCIP 1204, which defines data for environmental sensor stations, defines an object type called essAirTemperature. An environmental sensor station might support multiple temperature sensors; each sensor would be associated with its own instance of the object type, identified as `essAirTemperature.<sensor number>`. An SNMP agent could report the value of any object instance by creating an ordered pair of the name and value of the object instance and enclosing it in an SNMP message, potentially with other variable bindings.

In order for the SNMP manager to understand the data contained within each message, the SNMP manager and SNMP agent must agree on:

1. a mechanism to unambiguously identify the object type and object instance,
2. the precise semantics (i.e., meaning) of the data value,
3. the way in which the data will be encoded,
4. the operations that are allowed (e.g., can a manager alter the value or is it read-only), and
5. the data that should be supported.

These details are specified using a formalized NVT-ASCII text file known as a Management Information Base (MIB) module coupled with the definition of SNMP, as defined in RFCs 2578-2580 and RFCs 3411- 3418.

## 2.2 Management Information Base (MIB) and Module

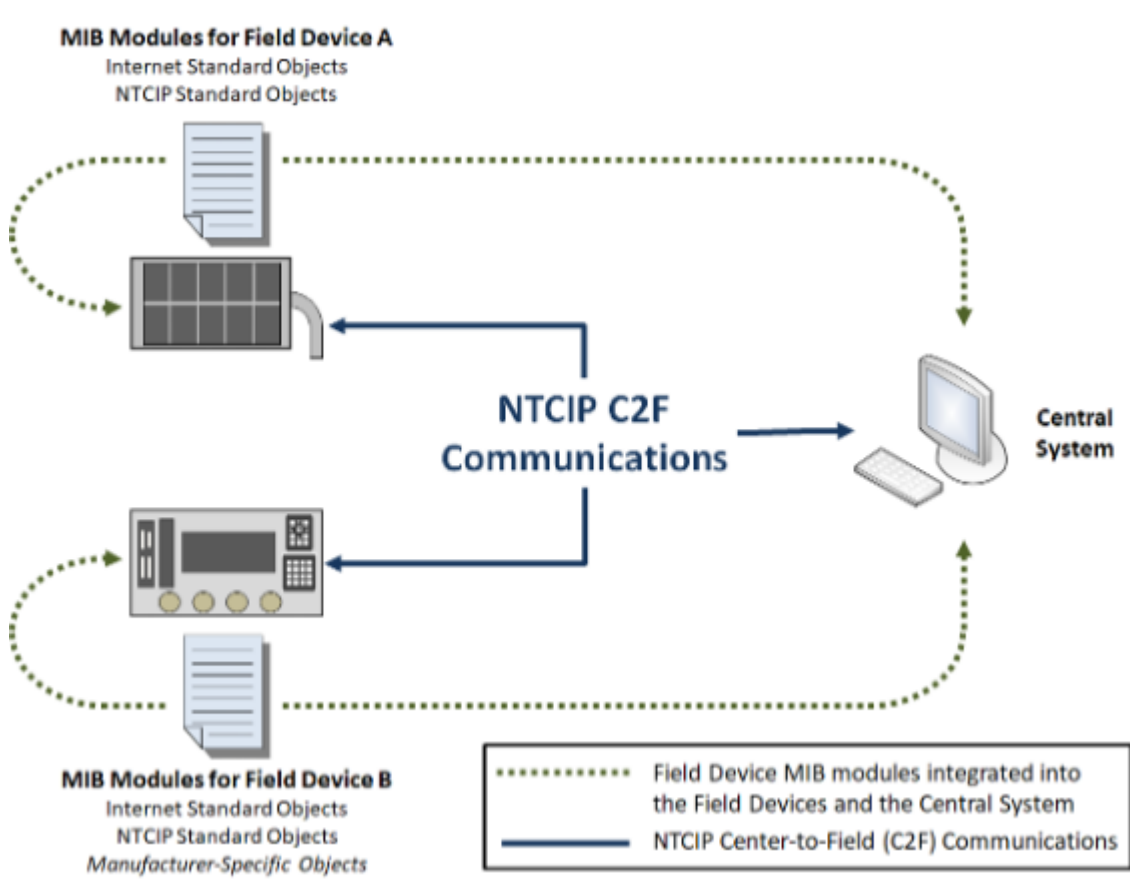
---

A Management Information Base (MIB) is the information that a specific SNMP context (typically a device) can make available for management operations. A MIB is defined by a text-based "MIB Module" (or collection of MIB Modules) that specifies the collection of object types that that the device supports. A MIB module is written using formal conventions in a structure that can read by both humans and computers. Object types include controls, configuration parameters, and status information (including sensor values and internal device status). Any data that is to be accessible via SNMP must be defined in a MIB module.

The standardized format of a MIB module allows automating much of the syntactic processing of data with off-the-shelf SNMP tools and allows developers to focus on implementing the semantics according to the documented definitions contained with the MIB module. The rules for the structure and management of this information also allow multiple MIB modules (e.g., NTCIP standard, Internet standard, manufacturer-specific extensions) to be used for a single

device. The MIB modules are integrated into the device software and its management stations using various methods (e.g, implemented directly as data structures in the software, compiled into binary tables that facilitate use by operational software). It is common for the MIB modules to be called the MIB but, technically, this is incorrect. The MIB is the instantiated collection of data objects available to management stations to configure, control, and monitor the device. It is not a database in the traditional sense.

Figure 2 illustrates how using MIB modules makes NTCIP C2F communications possible. In this example, the field devices are shown on the left of the figure with their associated MIB modules. The dynamic message sign uses only objects defined in standards while the traffic signal controller includes manufacturer-specific objects. The management station is commonly called a "central system" in transportation systems.



**Figure 2: Integrated MIB Modules Make NTCIP C2F Communications Possible**

## 2.3 Types of Object Types

Within SNMP, each piece of management information that can be exchanged is called an object instance. Object instances are abstracted into object types and formally defined within the MIB



module. However, when reading a MIB module, it is important to understand the different types of object types that might occur.

### 2.3.1 Leaf Object Types

---

Leaf object types do not have any globally unique sub-identifiers specified and are the only object types that can be instantiated in a device and retrieved via SNMP operations. The sub-identifiers for leaf object types are reserved for the identification of specific instances of the object type; they are only unique within the indicated device context (e.g., the object type `sysName` is a leaf object type and has a globally unique identifier; however, every SNMP device that instantiates this object type will use the same instance identifier). Leaf objects can be further classified in three different ways:

1. Based on whether multiple instances are allowed,
2. Based on the complexity of the underlying data, and
3. Based on the types of operations that are permitted.

#### 2.3.1.1 Classification by Instantiation Rules

If a leaf object type is defined as a part of a conceptual row of a table, multiple instances of the object are allowed, one for each row that has been conceptually instantiated within the table. This type of leaf object type is known as a "columnar object type". The different instances of the columnar object type are defined according to the mechanism defined by the `INDEX` clause of the conceptual row using one or more sub-identifiers.

If a leaf object type is not defined as a part of a conceptual table, only one instance of the object can exist; this object type is called a "scalar object type". The singular instance of a scalar object type is always identified by the single sub-identifier instance number of "0".

#### 2.3.1.2 Complexity of Object Types

SNMP was designed to exchange elemental data; as such, leaf object types are not allowed to directly use any ASN.1 data structure (e.g., `SEQUENCE`, `SEQUENCE OF`, `CHOICE`) for its syntax. This allows SNMP to provide very flexible data exchanges based on the need of the user. However, within a message, each elemental object instance must be uniquely identified, which adds overhead.

Within NTCIP, there are often needs to frequently exchange the same set of object instances in environments where the amount of data that is exchanged is a concern (e.g., communication environments with data usage limitations). These environments result in a need for a more

efficient solution than always exchanging data in its most elemental form. To overcome these issues, the NTCIP classifies every leaf object type as a simple, block, or configurable object type.

Simple object types represent elemental data, as traditionally defined with normal SNMP operations. Simple object types include integers, octet strings, object identifiers, bit strings, and other types that derive from these basic types.

Block object types represent data structures (e.g., SEQUENCE , SEQUENCE OF , CHOICE ) that are normally prohibited as SNMP object types. NTCIP gets around the SNMP prohibition by setting the SYNTAX clause of the OBJECT-TYPE macro to resolve to an OCTET STRING (typically by using a textual convention). It then defines the value of that OCTET STRING to be a serialization of a defined data structure. From the SNMP perspective, the data is an elemental OCTET STRING , but the sender and receiver perform additional encoding and decoding to allow for the exchange of a more complex structure. Within NTCIP, the preferred way to define the serialization of these data structures is by using an ITS0erString , which is defined by ISO 20684-1 as a data structure specified using X.680 ASN.1 and serialized using the Octet Encoding Rules (OER).

Configurable object types are similar to block objects in that each configurable object represents a data structure that can be serialized into an OCTET STRING and then be exchanged as a single SNMP object type. However, block object types have static definitions of their data structures defined within the MIB module. By comparison, configurable object types allow for a manager to configure the definition of the data structure after the implementation has been deployed and while the SNMP agent is running. Within NTCIP, configurable object types are typically SEQUENCE structures and the configuration is typically defined in a table where each ordered row of the table represents an ordered ComponentType (i.e., field) within the SEQUENCE structure. The

serialization of the configurable object typically uses OER to produce an OCTET STRING that can be exchanged by SNMP.

### 2.3.1.3 Object Type Permissions

SNMP object types can also be classified based on the types of data represented and the operations that are allowed under different scenarios. There are four basic types of object types from this perspective, as follows:

1. Status object types. Read-only objects that report the conditions that can be monitored by the SNMP agent.
2. Control object types. Writeable object types used to request real-time activation of a feature of a device. In some cases, control objects can also be used to report status.
3. Parameter object types. Writeable object types used to configure the SNMP agent where the parameter can be set and validated using a single SNMP set operation.
4. Interrelated parameter object types. Writeable object types used to configure the SNMP agent where the parameter has sufficient interrelationships with other object types to typically require multiple SNMP set operations using multiple SetRequest messages or a complex validation check that might consume more time than is reasonable for a traditional SNMP response.

When a device includes interrelated parameter object types, it defines a mechanism by which the object types can be safely configured. One such approach is to use the database transaction mode as specified in NTCIP 1201.

### 2.3.2 Conceptual Objects

In addition to the leaf object types described above, MIB modules can define three types of conceptual object types: conceptual tables, conceptual rows, conceptual leafs. Conceptual objects can theoretically be instantiated but because they have a MAX-ACCESS value of not-accessible, the rules of SNMP do not allow their exchange and they do not really exist as protocol entities. An implementation can only exchange instances of the accessible object types.

A conceptual table defines a set of object types that might traditionally be represented in a written document using a table format. Logically, each conceptual table consists of a number of columns and rows, where each column describes a particular type of data and each row represents a unique instance of data for each column. A conceptual table describes the information contained in the table and the rows that the table should have. It has a SYNTAX of SEQUENCE OF <EntryType>, where <EntryType> is the name of the conceptual row's data structure.

A conceptual row defines a set of object types that represent a logical unit of inter-related management information. The row typically supports multiple columns and the object types

referenced by that row logically has one instance for each defined row. The row object type defines the rules for the number of rows and the rules for creation and deletion of rows.

A conceptual leaf is a simple object type that has been identified as `not-accessible`; for example an index to a table. While SMIv1 allowed for index object types to be accessible, it provides no real value and increases the time it takes to walk through the data (i.e., by using `GetNextRequests`) supported by the SNMP agent. As a result, SMIv2 prohibits index object types from being accessible, unless they have been imported from an SMIv1 module. Conceptual leaves can also be defined when a standard wants to declare data that should be supported by a device even if it is never exchanged. For example, the trigger function is required to monitor data within the device and needs to be able to demonstrate that it has appropriate security credentials to access the data. The security credential information is stored in non-accessible objects to indicate that the controller needs to maintain this information but that the information should never be exchanged.

## 2.4 Document Overview

---

The remaining sections of this document define:

1. The mechanism used to allow multiple independent entities to define object types while maintaining globally unique names
2. Requirements for the development of MIB modules
3. Requirements for agent implementations
4. Considerations for operating agencies

In addition, Annex A provides the formal definition of the NTCIP 8004 MIB, which must be imported by all other NTCIP MIBs and Annex B provides the rules for converting NTCIP MIBs that are based on SMIv1 into the SMIv2 format required by this document.

## Object Identification [Normative]

---

SNMP allows managers and agents to implement MIB modules from multiple independent sources. Allowing this combination of object types presents a potential for naming conflicts (i.e., two sources defining different pieces of data using the same name). SNMP overcomes this challenge by identifying object types using the international object identifier tree, as defined by the ISO/IEC 9834-1 standard.

### International Object Identifier Tree

---

The international object identifier tree consists of three root nodes<sup>1</sup>. Each root node can be assigned multiple sub-nodes. Each sub-node may, in turn, have sub-nodes of its own.

Each node, whether a root node or a sub-node, is managed by some organization. For example, the three root nodes are managed by ISO, the International Telecommunication Union Telecommunication Standardization Sector (ITU-T), and jointly by ISO and ITU-T. The manager of any node may delegate the management responsibility for any sub-node underneath its branch. In this case, the delegated node is termed a subtree. Just as any sub-node may have its own set of sub-nodes, subtrees can have their own subtrees.

Except for the first two root nodes (which are limited to 40 sub-nodes each), the number of sub-nodes that any node may have is unlimited. Likewise, the number of sub-node levels is unlimited. As a result, the international object identifier tree can contain an unlimited number of nodes using a tree structure managed by multiple organizations.<sup>2</sup>

### Node Identification

---

Each node is assigned an integral number and optionally a label. The label is called the **OBJECT DESCRIPTOR** and provides a user-friendly textual name that can concisely express what the node is intended to represent. While the **OBJECT DESCRIPTOR** provides a useful human-language term to describe the node and should be designed to be unique within the scope of other sibling nodes of the same branch and level, there is no guarantee that it is unique within the entire international object identifier tree. Nonetheless, any node can be uniquely defined by following the sequence of nodes from the root to the specific node that needs to be identified. Unfortunately, this can produce a rather lengthy description since each name might consist of several characters. Alternatively, the same identifier can be produced by using the integral numbers assigned to each node rather than the **OBJECT DESCRIPTOR** s. This representation

typically results in a much more compact identifier that can more easily be processed by computers. This ordered list of integral values is called an `OBJECT IDENTIFIER`, which is known to identify a globally unique position on the identifier tree when properly registered.

The globally unique identifier can be used for any purpose for which an identifier may be useful. For example, most standards organizations have been assigned an `OBJECT IDENTIFIER` for the purposes of identification. The tree can also be used for managing groups of related data. For example, all objects related to an Actuated Signal Controller are organized under a node defined as 'asc'. In short, these attributes are a means for identifying some object, regardless of the semantics associated with the object.

#### Object Identifiers and SNMP

---

Within SNMP, the international object identifier tree is used to register object types with globally unique identifiers. The formal assignment of an `OBJECT IDENTIFIER` to each object type is contained in the MIB module.

SNMP also identifies object instances by using the international object identifier tree, but with a slight twist in that the instance portion of the identifier is only unique within its SNMP agent and context<sup>3</sup> -- not globally unique. For example, while the MIB provides a globally unique identifier for the object type of `essAirTemperature`, there might be multiple environmental sensor stations each containing multiple air temperature sensors. To identify a specific object instance within a specific environmental sensor station, SNMP extends the object type object identifier with an instance identifier (e.g., `essAirTemperature.1` to identify the reading from the first sensor). However, the instance value is not unique across different SNMP agents (i.e., every environmental sensor station will use `essAirTemperature.1` to identify the reading from its first air temperature sensor). The SNMP manager is responsible for distinguishing among these readings by combining the object instance identifier with the network address of the device and the context used within the message.

#### Registered Nodes

---

##### Root Nodes

---

The first two root nodes of the naming tree are administered by ITU-T (`itu-t`) and ISO (`iso`). The third node is jointly administered by ISO and ITU-T (`joint-iso-itu-t`, or sometimes shortened to `joint`).

NOTE---Until 1991, the U.S. name-registration authority conducted its business under the `{iso(1) member-body(2) us(840)}` arc, registering names for ANSI standards, private organizations with U.S. national standing, and the names of U.S. states and "state equivalents." In 1991, changes in the registration authority procedures standard ISO/IEC 9834 (ITU-T X.660) invalidated this procedure, requiring private organization names with national standing to be registered under the `{joint-iso-itu-t(2) country(16) us(840)}` arc. The existing register of private organization names moved, in fact, from the `{1 2 840}` arc to the `{2 16 840 1}` arc, with ANSI serving as the registration authority. Therefore, two equivalent prefixes exist (in perpetuity) for currently registered organization names. Post-1991 registrations are made only under the `{2 16 840 1}` arc, and organizations with pre-1991 registrations are encouraged (but not required) to construct no new identifiers under the `{1 2 840}` arc. Various sources provide current OID descriptions, including [www.oid-info.com/index.htm](http://www.oid-info.com/index.htm).

NOTE--- The International Telecommunication Union Telecommunication Standardization Sector (ITU-T) is the part of ITU (an agency of the United Nations) that provides standards for telecommunication equipment and systems. The Consultative Committee for International Telephony and Telegraphy (CCITT) was renamed ITU-T in 1993. The `itu-t(0)` arc is also named `ccitt` in remembrance that CCITT was previously an organization independent from ITU-T. Similarly, the `joint-iso-itu-t(2)` arc is also named `joint-iso-ccitt`.

#### NEMA Node

Under the `iso` node, ISO has designated one subtree for use by other (inter)national organizations (`org`). Under that subtree, one of the U.S. National Institute of Standards and Technology nodes is assigned to the U.S. Department of Defense (`dod`). The initial development of the Internet was a Department of Defense project and, therefore, the Internet community was assigned a node in the `dod` subtree. The Internet Architecture Board (IAB) administers the `internet` node. The descriptive name `internet` is defined as:

```
internet OBJECT IDENTIFIER ::= { iso org dod 1 }
```

(also known as 1.3.6.1)

Because of the ease of obtaining a node from IAB, NEMA requested and received a node that NEMA administers. This node is defined as:

```
nema OBJECT IDENTIFIER ::= { iso org dod internet private enterprise 1206 }
```

(also known as 1.3.6.1.4.1.1206)

All NTCIP-defined data related to device data dictionaries or protocols shall be defined under the NEMA branch of the tree. The organization of the naming tree down to the `nema` node is shown in Figure 3. The figure also shows the node for the ITS industry as used by the ISO 20684 standard series.

#### Portion of ISO Global Tree Showing Location of NEMA Node

The subtree for the NEMA node is shown in Figure 4. The description of each of the nodes are found in Annex A.1.

- 
- <sup>1</sup>. ISO 9834-1 uses the term "arc" instead of "node". ←
  - <sup>2</sup>. SNMP limits access to the first  $2^{32}-1$  nodes at any one level and to 128 levels, but from a practical perspective, this is still nearly limitless. ←
  - <sup>3</sup>. A context represents a coherent "collection of management information". Often, a physical controller will have a single context, but a single physical controller could control multiple NTCIP devices (e.g., multiple message signs, multiple traffic signals, etc.) and/or serve as a proxy agent to multiple non-NTCIP devices. In this case, each device should be assigned a separate context within the controller so that the management information for each device is kept as distinctly separate instances. ←

🕒 June 18, 2025



## Rules for Module Development [Normative]

---

Except as otherwise stated within this section and in Annex B, all NTCIP MIB modules shall conform to:

1. IAB STD 58, which includes:
  - a. RFC 2578,
  - b. RFC 2579, and
  - c. RFC 2580.
2. RFC 3584
3. RFC 4181
4. The additional constraints and conventions defined in this section

Annex B defines allowed exceptions that only apply when converting NTCIP MIB modules from the IAB STD 16 (SMIv1) format to the IAB STD 58 (SMIv2) format.

NOTE -- SMIv2 is the second version of the SMI but is used as the MIB module format for SNMPv3.

### General

---

The original text of the RFCs is indispensable for a full understanding of the MIB module format; however, RFCs are typically written in a conversational/explanatory tone which can result in redundancies and ambiguities as to the conformance status of each requirement. Further, some requirements in the referenced RFCs are written to only be applicable to IETF standard MIB modules while others are for non-standard MIB modules and tend to be from a technical perspective of what is allowed as opposed to what should be done. The conformance requirements adopted by NTCIP are intended to establish a policy for writing NTCIP standard MIB modules and as a result reflect a slightly different mix of conformance statuses from what the IETF has defined. Finally, the requirements for developing a MIB module are scattered over multiple RFCs, which describe not only the rules for writing the MIB modules but also on implications to implementations.

The tables presented in this section clarify and highlight the conformance status of certain requirements and is limited to those rules related to writing standard NTCIP MIB modules. Rules that are specified within X.208 and within the information modules defined within the referenced RFCs are omitted as they are unambiguous, and no changes are made. This document focuses on the requirements that are only defined within the text (non-module) sections of the RFCs.

This document groups requirements by topic, regardless of the source of the requirement. Each requirement is listed in a tabular format with an item number that allows for easy referencing. The tabular format consists of the following columns:

- Item: provides a unique identifier for the requirement.
- Requirement: A requirement stated as a "\"shall\" statement. This almost always requires a rewording of the statement compared with original RFC text.
- RFC: When the requirement represents an additional constraint or convention of this document, this column is blank; otherwise, it identifies the RFC number from which the requirement originates.
- Clause: When the requirement represents an additional constraint or convention of this document, this column is blank; otherwise, it identifies the location of the requirement within the referenced RFC in the following format:

```
<clause>"p"<paragraph>
```

where,

- `<clause>` indicates the clause number of the RFC where the requirement appears<sup>1</sup>
- `<paragraph>` indicates an ordered integral position of the paragraph within the indicated clause where the requirement appears (numbered and bulleted items are considered to be a part of the prior paragraph)
- RFC status: An indication of the conformance status of the requirement as interpreted from the RFC for RFC standard MIB modules.
- NTCIP status: An indication of the conformance status of the requirement for NTCIP standard MIB modules.

The codes used within the status columns are defined as follows:

**Table 1: Status Codes**

Code	Meaning
m	Mandatory
r	Recommended
o	Optional
o.#	An option group where at least one of the items with the same number (#) must be supported
d	Discouraged
x	Prohibited
c	Conditional

Status codes may be supplemented with footnotes that provide further details.

Any violations of the requirements contained within this document shall not negate the intent of any statement contained in a document or MIB module based on this document.

Non-standard (e.g., manufacturer-specific) MIB modules are outside the scope of this document.

## Rules for Modules

---

### General Rules for Modules

---

Requirements for the construction of X.208 modules within NTCIP standards are shown in Table 6. Most X.208 modules defined within NTCIP standards are MIB modules (i.e., they use the OBJECT-TYPE and/or the NOTIFICATION-TYPE macros). NTCIP standards also include a small number of other module types, such as the NTCIP8004-NEMA module, contained in Annex A of this document, which does not include any OBJECT-TYPE or NOTIFICATION-TYPE macros. The requirements presented in Table 2 apply to all X.208 modules contained in NTCIP standards.

**Table 2: General Rules for Modules**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.2.1.a	X.208 modules shall conform to the adapted subset of X.208 ASN.1 as specified in IAB STD 58	2578	1p1	m	m
4.2.1.b	Each modulerefererence (i.e., module name) shall be unique across all standard modules	2578	3p6	m	m
4.2.1.c	Each NTCIP module shall have a modulereference (i.e., name) that follows the form: NTCIP\<StandardNumber>-\<NodeName> where, \<StandardNumber> is the four-digit standard number \<NodeName> is the name of the MODULE-IDENTITY descriptor in UpperCamelCase Example: NTCIP1201-RecMechV2				r
4.2.1.d	Each revision <sup>2</sup> of a module (e.g., as contained in a new version of a standard) shall use the same modulereference (i.e., module name)	2578	3p6	r	m <sup>2</sup>
4.2.1.e	Modules shall use an empty AssignedIdentifier (i.e., there shall not be an OID between the module name and the "DEFINITIONS" keyword)	2578	3p6	m	m
4.2.1.f	Modules shall not IMPORT any "UNIVERSAL" types defined by ASN.1	2578	3.2p4	m	m
4.2.1.g	Modules shall not IMPORT the BITS construct	2578	3.2p4	m	m
4.2.1.h	Modules shall not IMPORT items not used	4181	4.4p4	r	r
4.2.1.i	Modules shall not define additional X.208 ASN.1 macros	2578	3p5	m	m
4.2.1.j	Modules shall not IMPORT or use SMIv1 macros	2578	3p5	m	m
4.2.1.k	Normative text shall not be placed in module comments	2578	3.4p1	r	m
4.2.1.l		4181	3p2	m	m <sup>3</sup>

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	A standard containing a <u>MIB module</u> shall contain certain defined sections				
4.2.1.m	A text-only, electronic version of the <u>MIB module</u> shall be produced per the rules of NTCIP 8005	-	-	-	m
4.2.1.n	Clause numbers included in the standardized NTCIP document shall be converted into comments in the electronic version of the <u>MIB module</u>	-	-	-	m
4.2.1.o	A standards-track <u>MIB module</u> shall be reviewed by an approved steward	4181	1p1	m	m
4.2.1.p	The rules defined in this document shall be waived when justification is fully documented	4181	4.9p9	o	o <sup>4</sup>

#### Rules for Module Components

The components contained within a module should follow a consistent order to facilitate use. The requirements for components of an NTCIP module are shown in Table 3.

**Table 3: Rules for Module Components**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.2.2.a	Modules shall not use the EXPORTS statement	2578	3.3p1	m	m
4.2.2.b	Modules shall use the IMPORTS statement to import all referenced external objects, including the referenced types and macros defined in [RFC 2578], [RFC 2579], and [RFC 2580] as needed	2578	3.2p1	m	m
4.2.2.c	The module shall include exactly one MODULE-IDENTITY macro immediately after the IMPORTS statement	2578	3p7	m	m
4.2.2.d	The module shall define any local TEXTUAL-CONVENTIONS immediately after the MODULE-IDENTITY macros			-	r
4.2.2.e	The module shall present any OBJECT-IDENTITY, OBJECT-TYPE, and NOTIFICATION-TYPE macros after and after all TEXTUAL-CONVENTIONS contained in the module			-	r
4.2.2.f	The module shall define at least one MODULE-COMPLIANCE macro	2580	5p2	c <sup>5</sup>	c <sup>6</sup>
4.2.2.g	The module shall define the MODULE-COMPLIANCE macro after all OBJECT-TYPES and			-	r



Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	NOTIFICATION-TYPEs contained in the module				
4.2.2.h	The module shall define OBJECT-GROUPs and NOTIFICATION-GROUPs after all MODULE-COMPLIANCE statements contained in the module			-	r

## Rules for Updating a Module

**Table 4: Rules for Updating a Module**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.2.3.a	Macro invocations (i.e., definitions) shall not be moved from one module to another	2578	10p3	r	r <sup>7</sup>
4.2.3.b	Macro invocations (e.g., even with a status of “obsolete”) shall not be removed from an updated module	4181	10p4	m	m
4.2.3.c	Normative changes shall not be made to macro invocations (when necessary, a new macro invocation will be used instead)	4181	10.1p1	m	m
4.2.3.d	New macro invocations shall be defined	4181	10.2p1	o	o

## Rules for Values

## Rules for Descriptors

RFC 2578 uses the term “descriptor” to refer to the concepts that X.208 terms a “valuereference” and it appears to be intended to apply to “typereferences” as well.

X.208 requires a valuereference to begin with a lowercase letter and the statement takes the form \<valuereference> \<macro> ::= \<value>. X.208 requires a typereference to begin with an uppercase letter and the statement takes the form \<typereference> ::= \<macro>. For example, the name “essAirTemperature” is a valuereference; the name “DisplayString” is a typereference.

This document defines “descriptor” to mean “either an X.208 valuereference or typereference” to avoid some ambiguities and misleading statements that exist in RFC 2578. The specific NTCIP requirements for a descriptor are defined in Table 5.

**Table 5: Rules for Descriptors**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.3.1.a	Descriptors shall be unique within a module	2578	3.1p3	m	m
4.3.1.b	Descriptors shall be mnemonic	2578	3.1p3	m	m
4.3.1.c	Descriptors shall not be longer than 64 characters in length	2578	3.1p3	m	m
4.3.1.d	Descriptors shall not be longer than 32 characters in length	2578	3.1p3	o <sup>8</sup>	o
4.3.1.e	Descriptors shall be unique across all standard modules that are likely to be included within NTCIP devices <sup>9</sup>	2578	3.1p4	m	m <sup>10</sup>
4.3.1.f	Descriptors shall start with the name of the device-type descriptor or an abbreviation thereof to promote understanding and uniqueness				r
4.3.1.g	Descriptors shall include a suffix that indicates a version number of any <u>object type</u> that is a replacement <u>object type</u> (e.g., <code>essSurfaceConductivityV2</code> )				r <sup>11</sup>
4.3.1.h	If an imported descriptor is duplicated (e.g., same term imported from two different modules) the importing module shall use the <code>Externalvaluereference</code> (or <code>Externaltypereference</code> ) format when referring to it (e.g., “\<modulereference>.\<descriptor>”)	2578	3.2p2	m	m
4.3.1.i	A descriptor (i.e., the term itself) shall not change after it is standardized	2578	3.6p1	m	m
4.3.1.j	A descriptor shall not be associated with more than one meaning	2578	3.6p1	m	m
4.3.1.k	A descriptor shall not be associated with more than one <u>OBJECT IDENTIFIER</u>	2578	3.6p1	m	m
4.3.1.l		2578	3.6p1	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	A revision shall not change the semantics of any standardized descriptor				

Rules for OBJECT IDENTIFIER Values

Table 6 defines additional requirements related to the assignment of OBJECT IDENTIFIER values.

Table 6: Rules for OBJECT IDENTIFIER Values

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.3.2.a	OBJECT IDENTIFIER values shall not exceed 128 sub-identifiers	2578	3.5p1	m	m
4.3.2.b	The value of each sub-identifier within an OBJECT IDENTIFIER value shall not exceed $2^{32}-1$	2578	3.5p1	m	m
4.3.2.c	An assignment of an OBJECT IDENTIFIER value shall not contain a NameForm component	2578	3.6p1	m	m

Rules for Macros

Rules for MODULE-IDENTITY Macro

The MODULE-IDENTITY macro is used to identify who is responsible for the MIB and to show the history of revisions. Requirements for the completion of the MODULE-IDENTITY macro within an NTCIP MIB module are as shown in Table 7.

**Table 7: Rules for the MODULE-IDENTITY Macro**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.1.a	The LAST-UPDATED clause shall indicate the date and time that this module was last edited	2578	5.1p1	m	m
4.4.1.b	The ORGANIZATION clause shall provide the full name of the working group	4181	4.5p2	m	m
4.4.1.c	The CONTACT-INFO clause shall include the name of the person to whom technical queries should be sent	2578	5.3p1	m	m <sup>12</sup>
4.4.1.d	The CONTACT-INFO clause shall include the postal address of the person to whom technical queries should be sent	2578	5.3p1	m	m <sup>14</sup>
4.4.1.e	The CONTACT-INFO clause shall include the telephone number of the person to whom technical queries should be sent	2578	5.3p1	m	x
4.4.1.f	The CONTACT-INFO clause shall include an email address	4181	4.5p2	m	m <sup>14</sup>
4.4.1.g	The CONTACT-INFO clause shall include the working group's website URL	4181	4.5p2	r	o
4.4.1.h	The DESCRIPTION shall contain a high-level textual description of the MIB module	2578	5.3p1	m	m
4.4.1.i	The REVISION clause shall be	4181	4.5p2	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	present for each standardized revision of the module				
4.4.1.j	The time of the most recent REVISION clause shall match the value of the LAST-UPDATED clause	4181	4.5p2	m	m
4.4.1.k	The DESCRIPTION clause associated with each REVISION clause shall indicate the version of the standard in which it is defined	4181	4.5p2	m	m
4.4.1.l	The DESCRIPTION clause associated with each REVISION clause shall provide a list of all significant changes	4181	4.5p2	r	m
4.4.1.m	The value assigned to the MODULE-IDENTITY descriptor shall be unique for the standardized module	4181	4.5p2	m	m
4.4.1.n	The MODULE-IDENTITY macro shall be updated for each revision	2578	10p2	m	m

#### Rules for the OBJECT-IDENTITY macro

The OBJECT-IDENTITY macro is the preferred method to declare a node on the international object identifier tree for administrative use. For example, the NEMA “devices” node is defined in NTCIP 8004 v03 using an OBJECT-IDENTITY macro. Requirements for the completion of the MODULE-IDENTITY macro within an NTCIP MIB module are as shown in Table 8.



Table 8: Rules for the OBJECT-IDENTITY macro

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.2.a	All administrative OIDs below the nema devices node shall be defined via the OBJECT-IDENTITY macro	4181	6p1	r	m
4.4.2.b	All normative text about an object identity shall be included within its DESCRIPTION clause	2578	7.5p1	m	m

Rules for the OBJECT-TYPE Macro

General Rules for the OBJECT-TYPE Macro

The OBJECT-TYPE macro is used to specify the object types contained in a MIB module. There are several groups of requirements related to the OBJECT-TYPE macro. General requirements for the OBJECT-TYPE macro are provided in Table 9.

**Table 9: General Rules for the OBJECT-TYPE Macro**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.3.1.a	Each unit of management information shall be specified using an instance of the OBJECT-TYPE macro	2578	7p1	m	m
4.4.3.1.b	The value assigned in the STATUS clause shall not be updated to restore an items status (e.g., do not change “deprecated” to “current”)	2578	10.2p1	r	r
4.4.3.1.c	All normative text about an object type shall be included within its DESCRIPTION clause (but this may include references to other text)	2578	7.5p1	m	m
4.4.3.1.d	The DESCRIPTION clause of each read-write and read-create object type shall define its behavior in response to an agent reboot	4181	4.6.2p3	r	m <sup>13</sup>
4.4.3.1.e	Clarifications and additional explanations shall be added to the DESCRIPTION clause as needed	2578	10.2p1	o	o
4.4.3.1.f	The length of any binary string appearing within a DEFVAL clause shall be divisible by 8	2578	7.9p7	m	m
4.4.3.1.g	The length of any hexadecimal string appearing within a DEFVAL	2578	7.9p7	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	clause shall be divisible by 2				
4.4.3.1.h	Any character string appearing within a DEFVAL clause shall not include any tab or line terminator characters	2578	7.9p7	m	m
4.4.3.1.i	A DEFVAL clause shall be revised as needed	2578	10.2p1	o <sup>14</sup>	d
4.4.3.1.j	The <u>object</u> identifier assigned to an invocation of the <u>OBJECT-TYPE</u> macro shall not end with a sub-identifier of zero	4181	4.6.5p1	m	m
4.4.3.1.k	The <u>object</u> identifier assigned to an invocation of the <u>OBJECT-TYPE</u> macro shall be unique	4181	4.6.5p1	m	m

#### Rules for Leaf Object Types

Leaf object types are the only object types that have instances that can be accessed via SNMP. Leaf object types include both scalar objects (i.e., with only one instance) and columnar objects (i.e., where there is one instance per conceptual row of a table). Table 10 provides additional requirements for the specification of leaf object types.

**Table 10: Rules for Leaf Object Types**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.3.2.a	The value of the SYNTAX clause shall be a base type, a BITS construct, or a textual convention	2578	7.1p1	c	c
4.4.3.2.b	When a standard textual convention provides the desired semantics, it shall be used in the SYNTAX clause for a new <u>object type</u>	4181	4.6.1.4p2	r	m
4.4.3.2.c	When a standard textual convention provides the desired semantics and uses a consistent syntax, it shall be used to replace the SYNTAX clause in a revised <u>object type</u>	4181	4.6.1.4p2	r	m
4.4.3.2.d	Object types with Boolean values shall be assigned the SYNTAX of the TruthValue textual convention	4181	4.6.1.9p2	r	r <sup>15</sup>
4.4.3.2.e	Constraints shall be added to the SYNTAX of an <u>object type</u> when the restrictions are implicit in the original definition but not defined	4181	4.9p3	o	m
4.4.3.2.f	The SYNTAX of <u>OBJECT IDENTIFIER</u> shall be used when the set of identification values need to be independently extensible	4181	4.6.1.5p2	r	m <sup>16</sup>

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	without a centralized registry				
4.4.3.2.g	The “Opaque” SYNTAX defined in SMIV1 shall not be used	2578	7.1.9p4	m <sup>22</sup>	m <sup>22</sup>
4.4.3.2.h	The SYNTAX of IpAddress shall not be used <sup>17</sup>	4181	4.6.1.7p1	m <sup>18</sup>	m <sup>22</sup>
4.4.3.2.i	The UNITS clause shall be present for object types with syntaxes that express values in units (e.g., time, distance, weight, volume)	2578	72p1	o	r
4.4.3.2.j	When possible, the text string for the UNITS clause shall conform to the International System of Units (SI) as defined by the International Bureau of Weights and Measures (BIPM)				r
4.4.3.2.k	The text string of the UNITS clause shall indicate the decimal position (e.g., “hundredths of seconds”, “centiseconds”)				m
4.4.3.2.l	The IndexPart of the OBJECT-TYPE macro shall not be included	2578	7.7p1	m	m
4.4.3.2.m	No nodes shall be defined underneath an OBJECT IDENTIFIER that is associated with a leaf object type.	2578	7.10p2	m	m

#### Rules for Table Object Types

Table object types represent conceptual tables, which is the mechanism defined by SNMP to support multiple instances of management information. They are referred to as “conceptual” since an instance of the object type is not directly accessible; SNMP only allows access to cells within the table. Table 11 provides additional requirements for the specification of object types that represent a conceptual table.



**Table 11: Rules for Table Object Types**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.3.3.a	The descriptor for a conceptual table object type shall end with the term "Table"	4181	Cp1	r	m
4.4.3.3.b	The SYNTAX shall be "SEQUENCE OF \<EntryType>" <sup>19</sup>	2578	7.1.12p1	m	m
4.4.3.3.c	The MAX-ACCESS clause shall be "not-accessible"	2578	7.1.12p2	m	m
4.4.3.3.d	The DESCRIPTION clause shall define the purpose of the table and summarize its contents			-	m
4.4.3.3.e	Static tables shall be associated with a scalar object type that indicates the number of rows in the table				m
4.4.3.3.f	An object type that indicates the number of rows in a table shall have a SYNTAX with a lower limit equal to or greater than one (1) <sup>20</sup>				r
4.4.3.3.g	The IndexPart of the OBJECT-TYPE macro shall not be included	2578	7.7p1	m	m
4.4.3.3.h	The OBJECT IDENTIFIER associated with a table object type shall have exactly one sub-node	2578	7.10p2	m	m
4.4.3.3.i	The sub-node to the table shall	2578	7.10p2	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	have a sub-identifier of 1				
4.4.3.3.j	The sub-node to the table shall be the row <u>object type</u> that defines a conceptual row within the table	2578	7.10p2	m	m

**Rules for Row Object Types**

Within SNMP, each conceptual table type has exactly one conceptual row type. Similar to the conceptual table, SNMP does not allow direct access to an instance of a row, which is why it is referred to as “conceptual”. Row object types identify the columns contained within each conceptual row instance of the table and provides information about managing the information within a row. Table 12 provides additional requirements for the specification of object types that represent a conceptual row within a table.

**Table 12: Rules for Row Object Types**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.3.4.a	The descriptor for a conceptual row object type shall be the descriptor for the conceptual table object type with the suffix "Table" replaced with the suffix "Entry"	4181	Cp1	r	m
4.4.3.4.b	The SYNTAX shall be "\<EntryType>" <sup>28</sup>	2578	7.1.12p1	m	m
4.4.3.4.c	The descriptor used for "\<EntryType>" shall be identical to the descriptor for the conceptual row, except the first letter will be in upper case	4181	Cp1	r	m
4.4.3.4.d	A revision shall add columns to the "\<EntryType>" as needed			-	<sup>o</sup> <sup>21</sup>
4.4.3.4.e	The MAX-ACCESS clause shall be "not-accessible"	2578	7.1.12p2	m	m
4.4.3.4.f	The DESCRIPTION clause shall explain how any INDEX objects that are not from the table are used to uniquely identify instances of objects	2578	7.7p8	m	m
4.4.3.4.g	The IndexPart of the OBJECT-TYPE macro shall be included	2578	7.7p1	m	m
4.4.3.4.h	The IndexPart shall use the AUGMENTS clause if the table has a one-to-one	2578	7.8.1p1	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	correspondence to the conceptual rows of an existing table <sup>22</sup>				
4.4.3.4.i	The IndexPart shall use the INDEX clause if the table does not have a one-to-one correspondence to the conceptual rows of an existing table	2578	7.8.1p1	m	m
4.4.3.4.j	The INDEX clause of a conceptual row that extends another table with a row relationship that is not one-to-one <sup>23</sup> shall use an INDEX clause where the initial indices are the indices of the parent table listed in the same order.	4181	4.6.4p1	r	m
4.4.3.4.k	The INDEX clause shall specify <u>object</u> types that allow unambiguous identification of a conceptual row	2578	7.7p2	m	m
4.4.3.4.l	The INDEX clause shall not reference scalar objects	2578	7.7p2	m	m
4.4.3.4.m	The INDEX clause shall not reference the same <u>object type</u> more than once	2578	7.7p2	r	m
4.4.3.4.n	If the INDEX clause references an <u>object type</u> that is external to the table, the Entry shall identify which rows will	4181	4.6.4p1	m	m

Item	Requirement exist within the local table	RFC	Clause	RFC Status	NTCIP Status
4.4.3.4.o	The INDEX clause shall not reference <u>object</u> types that have a SYNTAX of Counter32 or Counter64	2578	7.7p5	m	m
4.4.3.4.p	The INDEX clause shall not reference an <u>object type</u> that allows the value zero	4181	4.6.1.1p3	r	r
4.4.3.4.q	The INDEX clause shall not reference an <u>object type</u> that allows negative values	4181	4.6.1.1p3	m	m
4.4.3.4.r	The IMPLIED keyword shall not be used with an <u>object type</u> having a fixed length	2578	7.7p4	m	m
4.4.3.4.s	The IMPLIED keyword shall not be used with any <u>object type</u> other than the last <u>object type</u> listed in the INDEX clause	2578	7.7p4	m	m
4.4.3.4.t	The IMPLIED keyword shall not be used with an <u>object type</u> of variable length string that might have zero-length	2578	7.7p4	m	m
4.4.3.4.u	The IMPLIED keyword shall not be used within an INDEX clause for a conceptual row that might be expanded (i.e., extended with additional index columns)	4181	4.6.4p1	r	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.3.4.v	The MAX-ACCESS clause of each <u>object type</u> listed in the INDEX clause of its same table shall be “not-accessible”	2578	7.7p7	m <sup>24</sup>	m <sup>28</sup>
4.4.3.4.w	The SYNTAX and number of <u>object types</u> listed in the INDEX clause shall be designed to ensure that the OBJECT IDENTIFIERS of columnar <u>object instances</u> do not exceed 128 sub-identifiers	4181	4.6.6p2	r	m
4.4.3.4.x	Each subordinate columnar <u>object</u> shall be assigned an OBJECT IDENTIFIER that is a direct sub-identifier of the conceptual row.	2578	7.10p2	m	m
4.4.3.4.y	Each subordinate columnar <u>object</u> shall be assigned a descriptor that starts with the conceptual row descriptor minus the “Entry” suffix	4181	Cp1	r	m

#### Rules for Tables that Support Creation/Deletion of Rows

Some conceptual tables allow for row creation and deletion through SNMP operations or internally by the agent. These operations require proper management to ensure that the data within them do not produce logical inconsistencies. Table 13 provides requirements specific to conceptual tables that allow for row creation or deletion internally or by an SNMP operation (i.e., dynamic and managed tables). Table 14 provides requirements specific to conceptual tables that allow for row creation or deletion by an SNMP operation.



**Table 13: Rules for Tables that Support Creation/Deletion of Rows**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.3.5.a	The DESCRIPTION clause of the conceptual row shall define any conditions under which an <u>agent</u> can independently create/delete rows	4181	4.6.4p1	m	m
4.4.3.5.b	The DESCRIPTION clause of the conceptual row shall define what happens to dynamically created rows upon reboot	4181	4.6.4p1	o.1	o.1

**Table 14: Rules for Tables that Support Creation/Deletion of Rows**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.3.5.c	The conceptual row shall contain a <u>columnar object type</u> with the SYNTAX of StorageType	4181	4.6.4p1	o.1	o.1
4.4.3.5.d	The StorageType <u>object type</u> , if present, shall have a MAX-ACCESS of read-create	4181	4.6.4p1	m	m
4.4.3.5.e	The DESCRIPTION clause of the StorageType <u>object type</u> , if present, shall specify which columns within a permanent row can be modified	4181	4.6.4p1	m	m
4.4.3.5.f	The conceptual row shall contain a <u>columnar object type</u> with the SYNTAX of RowStatus	2578	7.1.12.1p1	m	m
4.4.3.5.g	The RowStatus <u>object type</u> shall have a MAX-ACCESS of read-create	2578	7.1.12.1p1	m	m
4.4.3.5.h	No <u>object type</u> contained within the conceptual row shall have a MAX-ACCESS of "read-write" <sup>25</sup>	2578	7.3p4	m	m
4.4.3.5.i	The DESCRIPTION clause of the RowStatus <u>object type</u> shall specify the columnar objects that must be set with valid values before row activation	4181	4.6.4p1	m	m
4.4.3.5.j	The DESCRIPTION clause of the RowStatus <u>object type</u> shall specify whether it is possible to modify other columns	4181	4.6.4p1	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	when the RowStatus is active				

Rule for the Syntax Specification of the OBJECT-TYPE Macro

General Rules for SYNTAX Specification

Table 15 provides requirements for the specification of the SYNTAX clause for leaf object types.

**Table 15: General Rules for SYNTAX Specification**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.1.a	An <u>object type</u> that is represented as an integer-valued enumeration shall have a SYNTAX of "INTEGER" with a NamedNumberList	4181	4.6.1.1p1	m	m
4.4.4.1.b	An <u>object</u> with a SYNTAX of an enumerated INTEGER that might conceivably be extended shall assign value 1 to 'other' to allow manufacturer-specific objects to extend the range <sup>26</sup>				r
4.4.4.1.c	Except for enumerations, the keyword "INTEGER" shall not be used	4181	4.6.1.1p3	r	m <sup>27</sup>
4.4.4.1.d	An <u>object type</u> that is represented as an integer, needs to support negative values, and can be represented within the range of a signed 32-bit integer shall have a SYNTAX based on Integer32	4181	4.6.1.1p3	r	m
4.4.4.1.e	An <u>object type</u> whose actual value 1) can increase above the maximum reported value or decrease below the minimum reported value, 2) does not latch at the extreme value when the value normalizes, and 3) should be represented within the range of an unsigned 32-bit integer shall have a SYNTAX based on Gauge32	4181	4.6.1.1p3	r	m
4.4.4.1.f	An <u>object type</u> whose actual value 1) can increase above the maximum value or decrease below the minimum value, 2) does not latch at the	4181	4.6.1.3p2	d <sup>28</sup>	d

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	extreme value when the value normalizes, and 3) should be represented within the range of an unsigned 64-bit integer shall have a SYNTAX based on CounterBasedGauge64				
4.4.4.1.g	An <u>object type</u> that is represented as an integer that is intended to monotonically increase until a maximum value and then wrap shall have a SYNTAX based on an accepted counter type (See Section 4.4.4.3)	2578	7.1.6p1	m	m
4.4.4.1.h	An <u>object type</u> that represents a duration between two epochs and measured in hundredths of a second shall have a SYNTAX based on TimeTicks	2578	7.1.8p1	r	m
4.4.4.1.i	An <u>object type</u> that is represented as an integer; does not have the semantics of a counter, gauge, or TimeTicks; and can be represented within the range of an unsigned, 32-bit integer shall have a SYNTAX based on Unsigned32	4181	4.6.1.1p3	r	r <sup>29</sup>
4.4.4.1.j	An <u>object type</u> that is represented as an integer; does not have the semantics of a counter, gauge, or TimeTicks; and supports values greater than $2^{31}-1$ shall have a SYNTAX based on Unsigned32	4181	4.6.1.1p3	m	m
4.4.4.1.k	An <u>object type</u> with a syntax dealing with time shall use a textual convention defined in ISO 20684-1 (e.g., ITSDailyTimeStamp)			-	r

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.1.l	An object type that represents an object identifier shall have a SYNTAX based on OBJECT IDENTIFIER <sup>30</sup>	2578	3.6p1	r	m
4.4.4.1.m	An object type that represents a series of named items where each item can have a Boolean representation shall have a SYNTAX based on the BITS construct	2578	7.1.4p1	r	r <sup>31</sup>
4.4.4.1.n	Any generic network/transport address shall be represented as a pair of object types using 1) a TransportDomain or TransportAddressType and 2) a TransportAddress from RFC 3419				m
4.4.4.1.o	An IP-specific network address shall be represented with the InetAddressType and InetAddress pair from RFC 4001	4181	4.6.1.7p1	r	r
4.4.4.1.p	All other object types shall have a SYNTAX based on OCTET STRING	2578	7.1.2p1	r	r
4.4.4.1.q	Object types shall not have a SYNTAX of DisplayString unless its values are intended to be explicitly limited to NVT ASCII	4181	Bp3	m	m
4.4.4.1.r	The SYNTAX shall be constrained using range or SIZE constraints as appropriate <sup>32</sup>	4181	4.6.1.4p1	r	m
4.4.4.1.s	If the SYNTAX value references a textual convention, a (potentially additional) constraint shall be allowed to further reflect the appropriate value	2578	11.2p1	m	m



Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.1.t	The constraints on the SYNTAX of a standardized <u>object type</u> shall not be revised if such a revision changes the meaning or validity of any value <sup>33</sup>				m
4.4.4.1.u	All values that are outside the standardized named bits and enumerations are reserved by the standard <sup>34</sup>				m
4.4.4.1.v	If the SYNTAX of a standardized <u>object type</u> is revised (which is only allowed in a backwards compatible manner per other rules), an explanation shall be added to the “\<Informative>” subclause of the <u>object type</u>				m
4.4.4.1.w	If the SYNTAX of a standardized <u>object type</u> is revised and the <u>object type</u> is writeable or is used as an index to a table that supports creation and/or deletion of rows, the module's compliance statement shall indicate the values that must be supported	4181	4.6.1.10p3	m	m

#### Rules for Enumerations

Table 16 provides additional requirements for the specification of leaf object types that are based on an INTEGER with a NamedNumberList.

**Table 16: Rules for Enumerations**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.2.a	The NamedNumberList shall specify all allowed values <sup>35</sup>	2578	7.1.1p2	m	m
4.4.4.2.b	The NamedNumberList shall start with the enumeration value of 1	2578	7.1.1p2	r	r
4.4.4.2.c	The NamedNumberList shall use contiguously assigned values	2578	7.1.1p2	r	r
4.4.4.2.d	The identifier of a named number shall not exceed 64 characters in length	2578	7.1.1p3	m	m
4.4.4.2.e	The identifier of a named number shall not exceed 32 characters in length	2578	7.1.1p3	o <sup>36</sup>	o
4.4.4.2.f	New enumerations shall be added during a revision	2578	10.2p1	o	o
4.4.4.2.g	An identifier for a standardized named number shall not be changed	2578	10.2p1	m	r <sup>37</sup>
4.4.4.2.h	A standardized named number shall be marked via a comment as “deprecated” or “obsolete”, as appropriate.	-	-	-	o
4.4.4.2.i	The DEFVAL clause, if present, shall use the identifier of the named number rather than the equivalent integer	4181	4.6.1.1p3	r	m

**Rule for Counters**

Table 17 provides additional requirements for the specification of leaf object types that are based on a counter.

**Table 17: Rules for Counters**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.3.a	The valuereference assigned to the counter object type shall denote plurality	2578	3.1p5	m	m
4.4.4.3.b	If the information is ever likely to wrap in less than an hour, the SYNTAX shall be based on a 64-bit counter syntax	2578	7.1.10p4	r	r
4.4.4.3.c	Counters that satisfy the semantics of Counter32/Counter64 as defined in RFC 2578 Clause 7.1.6 (e.g., have no defined initial value and have no requirements to reset) shall have a SYNTAX of Counter32 or Counter64, as appropriate	4181	4.6.1.2p1	r	m
4.4.4.3.d	Counters that satisfy the semantics of ZeroBasedCounter32/ZeroBasedCounter64 (e.g., initialize at zero and have no requirements to reset) shall have a SYNTAX of ZeroBasedCounter32 or ZeroBasedCounter64, as appropriate	4181	4.6.1.2p1	r	m
4.4.4.3.e	Counters that satisfy the semantics of ITSCounter32 (e.g., initialize at zero and can be reset) shall have a SYNTAX of ITSCounter32				r
4.4.4.3.f	Counters that do not satisfy the semantics of the other counter types shall use another textual convention that explicitly defines the intended semantics. <sup>38</sup>	4181	4.6.1.3p2	m	m
4.4.4.3.g	The MAX-ACCESS clause shall have a value of "read-only" or "accessible-for-notify"	2578	7.1.6p3	m	m
4.4.4.3.h	For Counter32 and Counter64 object types, the DESCRIPTION clause shall not define an initial value	2578	7.1.6p2	m	m
4.4.4.3.i	For Counter32, Counter64, ZeroBasedCounter32, and ZeroBasedCounter64	4181	4.6.1.2p1	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	object types, the DESCRIPTION clause shall not require the counter to reset in response to any event				
4.4.4.3.j	The DESCRIPTION clause shall identify any events that might cause the counter to have a discontinuity and how these events can be detected	4181	4.6.1.2p1	m	m
4.4.4.3.k	If discontinuities can occur at any time other than re-initialization of the device, the MIB module shall define a discontinuity object type that provides a reference as to when the last discontinuity occurred	2578	7.1.6p2	r	m
4.4.4.3.l	The object type shall not have a DEFVAL clause	2578	7.9p8	m	m

**Rule for TimeTicks**

Table 18 provides additional requirements for the specification of leaf object types that are based on the TimeTicks syntax.

**Table 18: Rules for TimeTicks**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.4.a	The DESCRIPTION clause shall define the two reference epochs for an object type with a SYNTAX of TimeTicks	2578	7.1.8p1	m	m

**Rule for OIDs**

Table 19 provides additional requirements for the specification of leaf object types that are based on the OBJECT IDENTIFIER syntax.

**Table 19: Rules for OIDs**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.5.a	OBJECT IDENTIFIER values shall be designed to limit their structure to 128 sub-identifiers	2578	7.1.3p1	m	m
4.4.4.5.b	OBJECT IDENTIFIER values shall be designed to limit each sub-identifier value to less than $2^{32}$	2578	7.1.3p1	m	m
4.4.4.5.c	The SYNTAX of an object type that is intended to refer to an object instance shall use the VariablePointer textual convention defined in RFC 2579	4181	4.6.1.5	r	m
4.4.4.5.d	The SYNTAX of an object type that is intended to refer to a row within a conceptual table shall use the RowPointer textual convention defined in RFC 2579	4181	4.6.1.5	r	m
4.4.4.5.e	The SYNTAX of an object type that is intended to refer to any other entity registered on the international object identifier tree shall use the AutonomousType textual convention defined in RFC 2579	-	-	-	-
4.4.4.5.f	An object type that is designed to reference and access a value stored in another object instance shall be associated with security credentials that	-	-	-	m



Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	are to be used when accessing that data.				
4.4.4.5.g	The DEFVAL clause shall use a single X.208 ASN.1 identifier	2578	7.9p4	m	m

Rule for BITS

Table 20 provides additional requirements for the specification of leaf object types that are based on the BITS construct.

**Table 20: Rules for BITS**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.6.a	A SYNTAX based on the BITS construct shall specify all allowed values <sup>39</sup>	2578	7.1.4p1	m	m
4.4.4.6.b	The named bits in the BITS construct shall start at 0	4181	4.6.1.6p1	m	m
4.4.4.6.c	The named bits shall be assigned to contiguous bit positions	2578	7.1.4p2	r <sup>40</sup>	r49
4.4.4.6.d	The identifier of a named bit shall not exceed 64 characters in length	2578	7.1.4p4	m	m
4.4.4.6.e	The identifier of a named bit shall not exceed 32 characters in length	2578	7.1.4p4	o <sup>41</sup>	o
4.4.4.6.f	New bit assignments shall be added to existing BITS constructs during a revision	2578	7.1.4p2	o	o
4.4.4.6.g	Existing bit assignments shall be aged as appropriate during a revision (e.g., denoted with a "– deprecated" statement)				o
4.4.4.6.h	An identifier for a standardized named bit shall not be changed	2578	7.1.4p2	m	m

**Rule for Block Objects**

Table 21 provides additional requirements for the specification of leaf object types that are based on the OCTET STRING syntax.

**Table 21: Rules for Block Objects**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.7.a	The descriptor for a <u>block object</u> shall have a suffix of "Block"				m
4.4.4.7.b	Block <u>object</u> types shall have a SYNTAX of ITSoerString as defined by ISO 20684-1				r <sup>42</sup>
4.4.4.7.c	Block <u>object</u> types shall define the structure of their embedded data using the format defined in Clause 4.5.4.2			-	r
4.4.4.7.d	Values of block <u>object</u> instances shall be encoded using the Octet Encoding Rules (OER) as defined in ITU-T X.696 (ISO/IEC 8825-7)			-	r
4.4.4.7.e	The MAX-ACCESS of a <u>block object type</u> shall not be greater than the MAX-ACCESS of any <u>object type</u> referenced by the <u>block object type</u> data structure <sup>43</sup>			-	m

**Rule for Octet Strings**

Table 22 provides additional requirements for the specification of leaf object types that are based on the OCTET STRING syntax.

**Table 22: Rules for Octet Strings**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.4.8.a	SIZE constraints shall be defined for OCTET STRINGS if they are more limited than the maximum of 0..65535	4181	4.6.1.4p1	r	m
4.4.4.8.b	OCTET STRINGS shall be constrained to 255 octets	2578	7.1.2p1	o	o
4.4.4.8.c	The size of any OCTET STRING used as an index shall be size constrained to facilitate <u>object instance</u> identification	4181	4.6.1.4p1	r	m

Rule for the DESCRIPTION Clause of the OBJECT-TYPE Macro

The DESCRIPTION clause consists of a text string intended to define the object type. NTCIP extends this definition by defining a series of subclauses to be embedded within this text string with precise rules for each subfield as defined in Table 23.

**Table 23: Rules for the DESCRIPTION Clause**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.5.a	The DESCRIPTION clause shall be divided into the following subclauses, each with their own label as indicated: \<Definition> \<Format> \<Superseded by> \<Informative> \<Object Identifier>				m
4.4.5.b	The \<Definition> subclause shall be present for all object types.				m
4.4.5.c	The \<Definition> subclause shall contain the normative definition of the object type				m
4.4.5.d	The \<Definition> subclause shall not be revised, once standardized, except to fix typos				r
4.4.5.e	The \<Format> subclause shall be used to define additional semantics for specific values in object types (e.g., a special meaning of the value 0) <sup>44</sup>				o
4.4.5.f	The \<Format> subclause shall be present for leaf object types that use a bitmaps or enumerations for the representation of data				m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.5.g	For bitmapped formats, the \<Format> subclause shall normatively define the format per Clause 4.5.3				m
4.4.5.h	The \<Format> subclause shall not be revised, once standardized, except to fix typos or to change the layout of the information				r
4.4.5.i	The \<Superseded by> subclause shall be present for any <u>object type</u> that has a STATUS of <u>deprecated</u> or <u>obsolete</u>				o
4.4.5.j	The \<Superseded by> subclause shall contain a reference to one or more <u>object types</u> that are intended to replace the functionality of the containing <u>object type</u> or shall indicate that the <u>object</u> was not replaced				m
4.4.5.k	The \<Superseded by> subclause shall not be revised once standardized <sup>45</sup>				r
4.4.5.l	The \<Informative> subclause shall be present for all <u>object types</u>				o
4.4.5.m					m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	The \\<Informative> subclause shall contain any informative information about the <u>object</u> type that the authors might deem to be useful for implementors and other users				
4.4.5.n	The \\<Informative> subclause shall be revised as appropriate to provide further clarification of the meaning of the <u>object type</u>				o
4.4.5.o	If the STATUS of the <u>object type</u> is aged (e.g., <u>deprecated</u> or made <u>obsolete</u> ), the \\<Informative> subclause shall explain why it was aged <sup>46</sup>				m
4.4.5.p	If the <u>object type</u> is a replacement for a <u>deprecated</u> <u>object type</u> , the \\<Informative> subclause shall identify the <u>deprecated</u> <u>object type</u> and explain how the issue was addressed				r
4.4.5.q	The \\<Object Identifier> subclause shall be present for all <u>object types</u>				m
4.4.5.r	The \\<Object Identifier> subclause shall provide an informative,				m



Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	human-readable version of the full object identifier in integer dot notation from the root node				
4.4.5.s	The \<Object Identifier> subclause shall not be revised, once standardized, except to fix typos				m

#### Mapping of the NOTIFICATION-TYPE macro

Object types are typically exchanged via get and set operations that are initiated by a manager; however, SNMP also allows for the definition of notification types, which are notices initiated by a device and sent to a manager. Notification types can be sent in either an unacknowledged mode (called a “trap”) or in an acknowledged mode (called an “inform”); both mechanisms rely upon the NOTIFICATION-TYPE macro to define the content and meaning of each notification type.<sup>47</sup>

Table 24 Mapping of the NOTIFICATION-TYPE macro

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.6.a	Each <u>notification type</u> shall be specified using an instance of the NOTIFICATION-TYPE macro	2578	8p1	m	m
4.4.6.b	The OBJECTS clause shall not include any <u>object types</u> that have a MAX-ACCESS clause value of "not-accessible"	2578	8.1p1	m	m
4.4.6.c	The value assigned in the STATUS clause shall not be updated to restore an items status (e.g., do not change " <u>deprecated</u> " to "current")	2578	10.3p1b	r	r
4.4.6.d	All normative text about a <u>notification type</u> shall be included within its DESCRIPTION clause (but this may include references to other text)	2578	8.3p1	m	m
4.4.6.e	The DESCRIPTION clause shall specify the meaning conveyed by each occurrence of each <u>object type</u> in the OBJECTS clause	2578	8.1p1	m	m
4.4.6.f	The DESCRIPTION clause shall specify which instance should be included for each occurrence of each <u>object type</u> in the OBJECTS clause	2578	8.1p1	m	m
4.4.6.g	The DESCRIPTION clause shall	4181	4.7p3	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	specify or otherwise reference the throttling technique to be used for the notification <sup>48</sup>				
4.4.6.h	Clarifications and additional explanations shall be added to the DESCRIPTION clause as needed	2578	10.3p1c	o	o
4.4.6.i	The next-to-last sub-identifier of the OID for any newly defined notification type shall have a value of zero	2578	8.5p1	m	m
4.4.6.j	An OBJECT IDENTIFIER that is assigned to notification type shall not have any sub-identifiers defined.	4181	4.6.5p2	r	m

#### Mapping of the Textual Convention Macro

Section 3 and 4 of RFC 2579 defines requirements related developing a new textual convention macro,

**Table 25: Mapping the Textual Convention macro**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.7.a	Textual conventions shall be defined for any SYNTAX that is used repeatedly with similar semantics				m
4.4.7.b	Textual conventions shall be defined for any SYNTAX that would benefit from being associated with a DISPLAY-HINT				m
4.4.7.c	The descriptor for the textual convention shall not conflict with reserved words	2579	3p2	m	m
4.4.7.d	The descriptor shall not be in all upper case	2579	3p2	m	m
4.4.7.e	The descriptor shall start with "Ntcip" to ensure uniqueness from any existing or future textual conventions from RFCs or other sources				m
4.4.7.f	The DISPLAY-HINT clause shall not be defined for any textual convention with a SYNTAX clause value of OBJECT IDENTIFIER, IpAddress, Counter32, Counter64, or any enumerated integer or BITS construct syntax	2579	3.1p1	m	m
4.4.7.g	The DISPLAY-HINT clause shall be defined for all other values of the SYNTAX clause	4181	4.6.3p2	r	r
4.4.7.h	The value assigned in the STATUS clause shall not be updated to restore an item's status (e.g., do not change " <u>deprecated</u> " to "current")	2579	5p2	o	o
4.4.7.i	The SYNTAX clause of a textual convention shall not	2579	3.5p1	m	m

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	refer to another textual convention				
4.4.7.j	The SYNTAX clause of a textual convention shall be constrained (i.e., sub-typed)	2579	4p1	o	o
4.4.7.k	ASN.1 type assignments (e.g., \<MyType>::=\<Type>) shall not be used <sup>49</sup>	2579	5p2	m	m

Mapping of the OBJECT-GROUP macro

Section 3 of RFC 2580 defines requirements related to the OBJECT-GROUP macro.

**Table 26: OBJECT-GROUP mapping**



Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.8.a	The descriptor for an OBJECT-GROUP shall include a revision number ("R#")				r
4.4.8.b	The OBJECTS clause shall not contain <u>object</u> types from other modules	2580	3.1p1a	m	m
4.4.8.c	The OBJECTS clause shall not contain <u>object</u> types with MAX-ACCESS of "not-accessible"	2580	3.1p1b	m	m
4.4.8.d	A MIB module shall not define any accessible <u>object</u> types that are not referenced by the OBJECTS clause of at least one <u>object</u> group	2580	3.1p2	m	m
4.4.8.e	A revision to a standardized OBJECT-GROUP macro shall not add or delete an <u>object</u> from the OBJECTS clause	4181	4.9p6a	m	m
4.4.8.f	The value assigned in the STATUS clause shall not be updated to restore an items status (e.g., do not change " <u>deprecated</u> " to "current")	2580	7.1p2a	o	o
4.4.8.g	The STATUS clause of the OBJECT-GROUP shall change when the status of a referenced <u>object</u> type changes	2580	7.1p2f	o <sup>50</sup>	o
4.4.8.h	The DESCRIPTION clause shall				r

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
	identify the standard and version where the group was originally defined <sup>51</sup>				

Mapping of the NOTIFICATION-GROUP macro

Section 3 of RFC 2580 defines requirements related to the NOTIFICATION -GROUP macro.

**Table 27: NOTIFICATION-GROUP mapping**

Item	Requirement	RFC	Clause	RFC Status	NTCIP Status
4.4.9.a	The NOTIFICATIONS clause shall not contain notification types from other modules	2580	4.1p1	m	m
4.4.9.b	A MIB module shall not define notification types that are not referenced by the NOTIFICATIONS clause of at least one notification group	2580	4.1p2	m	m
4.4.9.c	A revision to a standardized NOTIFICATION-GROUP macro shall not add or delete a NOTIFICATION from the NOTIFICATIONS clause	4181	4.9p6a	x	x
4.4.9.d	The value assigned in the STATUS clause shall not be updated to restore an items status (e.g., do not change "deprecated" to "current")	2580	7.2p2a	o	o
4.4.9.e	The STATUS clause of the NOTIFICATION-GROUP shall change when the status of a referenced notification type changes	2580	7.2p2f	o	o

## Mapping of the MODULE-COMPLIANCE macro

Section 3 of RFC 2580 defines requirements related to the MODULE-COMPLIANCE macro. Until a practical benefit can be demonstrated in using such statements, the NTCIP community has decided to document conformance with PRLs, RTMs, MIB compliance tables, and supported range tables. PRLs and RTMs are defined in NTCIP 8002; MIB compliance tables are intended to document how the objects within a MIB have evolved over time; supported range tables are intended to precisely define the minimum required ranges for each object for each version of the

standard in a relatively concise manner. The MIB compliance and supported range tables should be presented as defined in this clause.

#### MIB Compliance Table

The top row of the MIB Compliance Table shall provide the column names. The first column shall be labeled "Group" and subsequent columns shall indicate a compliance identifier, which is typically the version number of the standard (e.g., "v03")<sup>52</sup>. The first column of second row of the table shall contain the label "Status"; subsequent columns of the second row show indicate the status of each version of the MIB as follows:

1. "c" indicates current
2. "d" indicates deprecated
3. "o" indicates obsolete

Typically, only the latest version would be shown as current, but a standard could use multiple current columns to define different profiles (e.g., NTCIP 1203 could define a vXX profile for CMS and a vXX profile VMS, both of which are current simultaneously). Columns should only be shown as obsolete once all implementations are believed to be removed from service.

Subsequent rows of the table shall list each object and notification group defined by the MIB, with the descriptor of the group provided in the left-hand column of the table. The subsequent columns shall show the conformance status of the group as defined for the indicated version of the standard; conformance shall be shown using the conformance symbols as defined for use within PRLs. Finally, any cell within the table can be associated with a note via a reference symbol (e.g., "[1]"). The text of the notes shall appear immediately after the table.

Values within a column shall not be changed, except to correct typos or to change the status of the MIB version (i.e., as recorded in row 2). MIB versions that were previously documented as obsolete in a previous version of the standard can be omitted from the table. Each version of the standard should add at least one new column.

Table 28 provides an example of a MIB compliance table.

**Table 28: Example MIB Compliance Table (based on a draft of NTCIP 1203)**

Group	v01	v02	v03	v04-VMS	v04-CMS
Version Status	o	d	d	c	c
dmsTypeGroupR1	M	M	M	M	M
dmsTechnologyGroupR1	0.1(2)	M	M	M	M
dmsDisplayCapabilitiesGroupR1	0.1(2)	Capabilities:M	Capabilities:M	Capabilities:M	Capabilities:M
dmsVmsCapabilitiesGroupR1		VMS:M	VMS:M	M	
dmsFontGroupR1	0.2	Fonts:M	Fonts:M	Fonts:M	Fonts:M
dmsFontGroupR1Ext		Fonts:M	Fonts:M	Fonts:M	Fonts:M

**Supported Range Table**

MIB modules define the standardized allowed syntax for each object type; however, there are cases where the intent is to allow implementations to subrange the standardized syntax. Within prior versions of NTCIP (i.e., those based on SMIv1), the default policy was that implementations were allowed to subrange any object unless otherwise indicated. In general, the NTCIP standards were rather lax in stating these exceptions. Within SMIv2 (and specifically RFC 2580), the default is that implementations are required to support the full standardized range unless otherwise indicated.

To conform to the SMIv2 format, all NTCIP standards should include a supported range table to indicate the intent of supported ranges so that implementations (especially manger implementations) can manage expectations of what values might be supported. Specifically, these tables shall identify the minimum requirements for claiming support for an object and shall describe any conditional support rules.

The supported range table shall indicate the following columns in the header row:

- <sup>1</sup> An "Obj" column that lists the descriptor of each object type in the MIB module for which implementations are not required to support the full range of values.
- <sup>2</sup> A "Versions" column that identifies the compliance statements to which the refinement applies (this is typically just a version number but could include a version number coupled with a profile name).
- <sup>3</sup> A "Refinement" column that identifies the allowed variations in the object type that implementations are allowed to impose.

The first row of the table shall list the column headings. Subsequent rows are presented in groups, with each group consisting of one row that displays the object descriptor across an entire row of the table followed by one or more rows where each subsequent row list one or more versions and the refinement specification that applies to the object type for the indicated versions. The refinement specification shall be documented according to the Object construct defined within the MODULE-COMPLIANCE macro in Section 2 of RFC 2580, but omitting the "OBJECT" value(ObjectName)' portion.

Table 29 provides an example of an object refinement table.

**Table 29: Example Object Refinement Table**

Obj	Versions	Refinement
dmsMessageMemoryType		
	v01	SYNTAX INTEGER { currentBuffer (5) } DESCRIPTION "The value of 'other' is reserved for manufacturer specific use and is not expected to be widely used. Implementations shall support at least one of 'permanent', 'changeable', or 'volatile' in addition to 'currentBuffer'. Support for 'schedule' is conditional based on support for dmsActionTableGroupR1. The enumeration of 'blank' was not defined in v01."
	v02, v03	SYNTAX INTEGER { currentBuffer (5) } DESCRIPTION "The value of 'other' is reserved for manufacturer specific use and is not expected to be widely used. Implementations shall support at least one of 'permanent', 'changeable', or 'volatile' in addition to 'currentBuffer' and 'blank'. Support for 'schedule' is conditional based on support for dmsActionTableGroupR1."
	v04-VMS	SYNTAX INTEGER { currentBuffer (5) } DESCRIPTION "The value of 'other' is reserved for manufacturer specific use and is not expected to be widely used. Implementations shall support at least one of 'changeable' or 'volatile' in addition to 'currentBuffer' and 'blank'. Implementations may support 'permanent'. Support for 'schedule' is conditional based on support for dmsActionTableGroupR1."
	v04-CMS	SYNTAX INTEGER { permanent (2), currentBuffer (5) } DESCRIPTION "The value of 'other' is reserved for manufacturer specific use and is not expected to be widely used. Implementations shall support 'permanent', 'currentBuffer', and 'blank'. Implementations shall not support 'changeable' or 'volatile'. Support for 'schedule' is conditional based on support for dmsActionTableGroupR1."
dmsMessageNumber		
	v01	DESCRIPTION "Range is limited based on memory type as follows: permanent: per PICS changeable: per PICS volatile: per PICS currentBuffer: (1) schedule: (1)"
	v02,v03, v04-VMS, v04-CMS	DESCRIPTION "Range is limited based on memory type as follows: permanent: per PICS changeable: per PICS volatile: per PICS currentBuffer: (1) schedule: (1) blank: (1..255) "
dmsMessageOwner		
	v01,v02, v03	DESCRIPTION "Implementations shall support only <u>NVT-ASCII</u> characters."
	v04-VMS, v04-CMS	DESCRIPTION "Implementations shall support all <u>NVT-ASCII</u> characters and any other UTF-8 characters identified in the PICS"

**Summary**

The format of the MIB Compliance and Supported Range Tables are designed to facilitate the production of corresponding MODULE-COMPLIANCE statements if necessary, in the future. However, the tabular structure is intended to facilitate identifying changes that have occurred over time in a more concise manner and to reduce redundancy within the text.

Table 30 defines additional rules for populating the MIB compliance and object refinement tables.

**Table 30: MIB Compliance Table Rules**



Item	Requirement	RFC	Clause	RFC Status <sup>53</sup>	NTCIP Status <sup>54</sup>
4.4.10.a	Each version of a MIB shall be associated with a MIB compliance statement	2580	5p2	m	m
4.4.10.b	The MIB compliance statement shall identify each required and optional <u>object</u> group and <u>notification</u> group for the indicated version	4181	4.8p2a	r	m
4.4.10.c	The MIB compliance statement shall specify any allowed implementation variances (e.g., in access permissions or in supported <u>object</u> ranges)	4181	4.8p2b	m	m
4.4.10.d	The groups listed in the MIB compliance statement shall include any and all required <u>object</u> groups and/or types from external MIBs, prefixed with their module descriptor	4181	4.8p5a 4.8p5b	r	m
4.4.9.f	When a writeable enumerated integer or a BITS construct that is referenced by the compliance statement has enumerations or named bits added, a note that specifies support for the original set of values shall be added to all previous versions of the respective	4181	4.9p7	r	m

Item	Requirement	RFC	Clause	RFC Status <sup>53</sup>	NTCIP Status <sup>54</sup>
	object group(s) (if not already present)				
4.4.10.e	The information contained within the MIB compliance statement shall be consistent with the information provided within the respective PRL(s).			-	m

## Additional NTCIP Requirements

### Required Implementation Notes

Every NTCIP standard containing a MIB module shall contain the following normative text prior to its MIB compliance table.

The MIB compliance and object refinement tables may partially duplicate and extend requirements contained within the PRL and RTM. In case of any conflict between these sources, the PRL and RTM take precedence over the MIB compliance and object refinement tables.

To claim support for any object type, implementations shall:

- support the defined MAX-ACCESS and SYNTAX of the object type, unless the object refinement table defines a refinement for the object type for the relevant compliance statement;
- when the MIB object refinement table defines a refinement for an object type for the relevant compliance statement, support at least the minimum access and values specified.

### NTCIP-Defined Table Types

NTCIP defines three types of conceptual table types: static, dynamic, and managed.

The number of conceptual rows contained within a static conceptual table shall not change during run time, irrespective of whether they have been initialized or not. As a result, an SNMP manager can successfully query any supported object from any supported conceptual row at any time but should be prepared to receive a data that is not initialized.

The number of conceptual rows contained within a dynamic conceptual table can change during run time, but only through processes internal to the SNMP agent (e.g., an agent creating a new log entry) that the manager does not directly control.

The number of conceptual rows contained within a managed conceptual table can change during run time due to SNMP management operations (and optionally due to internal operations).

In the case of dynamic and managed tables, an SNMP manager should be prepared to receive an error response if an object instance is requested before its creation or after its deletion.

#### Bitmap Format

---

The content of the format subclause is delimited by “\<Format>” and shall be defined according to the following:

1. The format clause shall identify the meaning of each bit by its bit number. Bit numbering shall start with zero (0). This may be done in textual form (e.g., “each bit shall represent a sequential sensor number with sensor 1 mapped to Bit 0”) or may use a format such as:

#### Bit Name Description

\<x> \<bitName> = \<description>

...

1 \<bitName> = \<description>

0 \<bitName> = \<description> (Least Significant Bit of the Least Significant Byte)

- <sup>1</sup>. If the SYNTAX is based on a “BITS” construct, which is required for all new bitmapped objects, bit numbers are serialized according to the rules defined by RFC 3416 where Bit 0 is the “first bit”. This means that Bit 0 is the high-order bit of the first octet in the serialization.
- <sup>2</sup>. If the SYNTAX is based on an integer type (e.g., Integer32), which is only allowed for object types converted from SMIv1, Bit 0 shall represent the low-order bit. This means that Bit 0 is the low-order bit of the last octet in the serialization.
- <sup>3</sup>. If the SYNTAX is based on one of the NTCIP octet-string-based bitmap textual conventions (e.g., NtcipOctetBitmap8, NtcipOctetBitmap16), which are only allowed for object types converted from SMIv1, Bit 0 shall be mapped to the lowest order bit of the final octet. This means that Bit 0 is the low-order bit of the last octet in the serialization.
- <sup>4</sup>. If the SYNTAX is based on an OCTET STRING, which is only allowed for object types converted from SMIv1, the mapping of bit numbers to the bits within the serialization shall be defined within the \<Format> subclause of the DESCRIPTION clause of the object type.

Figure 5 depicts the impact of the above rules for an example object type with 14 bits defined (resulting in two bits of padding that might not be explicitly defined). Although new objects using

bitmaps should use the BITS construct standardized by [RFC 2578](#), older objects converted from SMIv1 might still use one of the other constructs.

BITS						
Syntax Bit #	0	●	●	●	7	8
Encoding Bit #	8	●	●	●	1	8
Octet Order	Leading Octet	Trailing Octet				
INTEGER						
Syntax Bit #	pad	pad	14	●	8	7
Encoding Bit #	8	●	●	●	1	8
Octet Order	First Octet	Last Octet				
BITMAP16						
Syntax Bit #	pad	pad	14	●	8	7
Encoding Bit #	8	●	●	●	1	8
Octet Order	Leading Octet	Trailing Octet				
OCTET STRING						
Syntax Bit #	To be defined by <a href="#">object type definition</a>					
Encoding Bit #	8	●	●	●	1	8
Octet Order	Leading Octet	Trailing Octet				

Figure 5 Bit Ordering of a 14-Bit Value<sup>55</sup>

ITSOerString Specifications

ITSOerString Syntax

The "ITSOerString" textual convention is defined in ISO 20684-1 as an unconstrained OCTET STRING representing value of an [X.680 ASN.1](#) structure encoded using [OER](#). Within [SNMP](#), the size of an OCTET STRING is limited to 65535 octets. An [object type](#) that uses the ITSOerString should further constrain this size as tightly as appropriate for its intended use.

Within NTCIP, the definition of the X.680 ASN.1 structure shall be defined or referenced by the \<Definition> subclause of the OBJECT-TYPE macro according to the rules in Clause 4.5.4.2.

#### ITSOerString Format

An object with a SYNTAX based on the ITSOerString or similar textual convention shall define the data structure to be encoded using any valid X.680 ASN.1 type; however, the following is substituted for the definition of NamedType to add mechanisms to reference other objects contained in the MIB:

NamedType ::=

identifier Type

| valueresource & "." & IndexSuffixes ExternalReference

| "\*" & valueresource & "." & IndexSuffixes ExternalReference

IndexSuffixes ::=

IndexSuffix

| IndexSuffixes & "." & IndexSuffix

IndexSuffix ::= number | DefinedValue | "(" ValueRange ")" | "\*"

ExternalReference ::=

"-- @" modulereference

| empty

where,

1. valuereference is a descriptor assigned to an invocation of an OBJECT-TYPE macro.
2. IndexSuffixes is a list of the associated indexes as required to explicitly identify the exact instance(s) of a given object-type. Each index may be identified as either a specific value or a variable; if a variable is used, the variable shall be defined within the same module. The IndexSuffixes for a leaf object shall always be "0".
3. A non-empty ExternalReference shall be present at the end of every line containing a valuereference, if the valuereference is defined in a different module
4. modulereference, when present, indicates the MIB module where the object type referenced by valuereference is defined

The first valuereference option supports the definition of block object types. For example, within a SEQUENCE, instead of providing an "identifier Type" pair, the specification can indicate the name of an object instance that should exist within the agent. The object instance is identified using the descriptor for the defined OBJECT-TYPE along with the appropriate index value(s). The "Type" to be associated with the NamedType shall be the SYNTAX value assigned in the OBJECT-TYPE macro for the associated valuereference. If an error occurs in trying to produce such an encoding, the entire ITSOerString shall produce a zero-length string.

EXAMPLE SEQUENCE { dmsSignAccess.0 -- @NTCIP1203-Dms

dmsMessageMultiString.3.2 -- @NTCIP1203-Dms

}

This would provide a structure containing the scalar value of dmsSignAccess and the dmsMessageMultiString for the second changeable message.

The second valuereference option supports the definition of configurable object types and entails an additional level of dereferencing. For example, within a SEQUENCE, instead of providing an "identifier Type" pair, the specification can indicate the name of an object instance (called the pointer object instance) that should exist within the agent and that is designed to reference an object instance (called the data object instance). The pointer object instance is identified using the descriptor for the defined pointer OBJECT-TYPE along with the appropriate index value(s). However, the actual data being encoded is the value pointed to by the pointer object instance, which is the value of the data object instance. As such, when encoding and

decoding, the type to be used shall be derived from the SYNTAX assigned in the OBJECT-TYPE macro for the associated data object instance. If an error occurs in trying to produce such an encoding, the entire ITSOerString shall produce a zero-length string.

EXAMPLE SEQUENCE { \*fdObjectGroupFieldObject.x.y.\* }

This would provide a structure containing the ordered list of objects referenced by all fdObjectGroupFieldObjects with an initial index of x and a secondary index of y. For example, if fdObjectGroupFieldObject.x.y.1 referenced dmsMessageMultiString.3.2 and fdObjectGroupFieldObject.x.y.2 referenced dmsMessageRunTimePriority.3.2 with no other entries for fdObjectGroupFieldObject.x.y, the structure would equate to

SEQUENCE { dmsMessageMultiString.3.2 -- @NTCIP1203-Dms

dmsMessageRunTimePriority.3.2 -- @NTCIP1203-Dms

}

The "ValueRange" notation of IndexSuffix shall be interpreted as a shorthand notation for listing every instance identified within the ValueRange. The "\*" notation of IndexSuffix shall be interpreted as a shorthand notation for listing every instance contained in the device in the same order as would appear in a walk of the MIB.

EXAMPLE For example, consider a device that has the following information in a conceptual table:

indexA	indexB	sampleObject
1	1	A
1	2	B
2	1	C
3	1	D
3	2	E

The field:

sampleObject.(1..2).\*

would evaluate to

sampleObject.1.1,



sampleObject.1.2,

sampleObject.2.1

Annex C provides several examples.

## Object Identifier Layout

---

All items that NTCIP registers on the international object identifier tree shall be registered under the nema node at { iso(1) org(3) dod(6) internet(1) private(4) enterprises(1) nema(1206) }.

NTCIP standards may reference and require support for objects located elsewhere on the international object identifier tree.

The NTCIP Coordinator shall ensure the assignment of a unique node(s) to each standard that requires a X.208 module. This assignment(s) shall be documented within an X.208 module contained within the standard. Such assignments are typically used as the name of the MODULE-IDENTITY invocation within the module.

NOTE—Secondary assignments might be used for administrative or other objects that need to exist in other portions of the international object identifier tree.

## Nodes under the MODULE-IDENTITY

---

Exceptions to these rules can be allowed with the approval of the Joint Committee on the NTCIP.

### Notification Types

The sub-identifier zero (0) under the MODULE-IDENTITY node shall be assigned or reserved for a descriptor identical to that of the MODULE-IDENTITY with the suffix "Notifications" added. Any notification types defined in the module shall be defined directly under this node. This ensures that each notification type is assigned a name with the next-to-last sub-identifier equal to zero (0), which is required by Clause 4.7 of RFC 4181.

Example — ntcipSmiNotifications would have the RELATIVE-OID of { transportation(4) protocols(1) ntcipSmi(5) 0 }<sup>56</sup>

### Conformance

The sub-identifier 127 under the MODULE-IDENTITY node shall be assigned or reserved for a descriptor identical to that of the MODULE-IDENTITY with the suffix "Conformance" added.

NOTE—RFC 4181 recommends placing all object types under sub-identifier one (1) and placing conformance under sub-identifier two (2). NTCIP standards place object types directly below the MODULE-IDENTITY node and shifts the conformance node to sub-identifier 127 to allow sequential numbering of object types.

Example — ntcipSmiConformance would have the RELATIVE-OID of { transportation(4) protocols(1) ntcipSmi(5) 127 }<sup>57</sup>

Sub-identifier one (1) under the conformance node shall be reserved for a compliance node with the descriptor of the MODULE-IDENTITY with the suffix "Compliances". As of 2023, NTCIP does not define MODULE-COMPLIANCE statements, but this node is reserved for this purpose if the policy changes.

Example — ntcipSmiCompliances would have the RELATIVE-OID of { transportation(4) protocols(1) ntcipSmi(5) ntcipSmiConformance(127) 1 }

Sub-identifier two (2) under the conformance node shall be assigned to or reserved for a group node with the descriptor of the MODULE-IDENTITY with the suffix "Groups".

Example — ntcipSmiGroups would have the RELATIVE-OID of { transportation(4) protocols(1) ntcipSmi(5) ntcipSmiConformance(127) 2 }

OBJECT-GROUP and NOTIFICATION-GROUP macros included in the MIB module shall be assigned a sub-identifier under the group node. The descriptor for the group shall include the descriptor of the MODULE-IDENTITY, descriptive terms as needed, and a suffix of "GroupR#[Ext#]", where "[Ext#]" is an optional extension number used to extend a previously defined group and each "#" represents a sequential number of the revision or extension. The "R#" should indicate the sequential position of the REVISION as recorded in the MODULE-IDENTITY's revision history, or the DESCRIPTION clauses within the revision history should indicate how the "R#"s relate to the revisions.

Example — dbMgmtV2GroupR1 is assigned to the RELATIVE-OID of { transportation(4) devices(2) global(6) globalV2(9) dbMgmtV2(1) dbMgmtV2Conformance(127) dbMgmtV2Groups(2) 1 }

#### Object Types

OBJECT-TYPE macros included in the MIB module shall be assigned a sub-identifier under the standard node, perhaps with intermediate nodes to further group object types.

Example – dbMgmtV2Mode has the RELATIVE-OID of { transportation(4) devices(2) global(6) globalV2(9) dbMgmtV2(1) 1 }

## MIB Design Considerations

---

### Maintain Secure Data

---

Access to data defined within the device needs to be controlled under all scenarios without any possibility of enabling any indirect access by unauthorized users. For example, a configurable event log might monitor data within the device to create a log entry to capture data when a defined event occurs. If a user is able to configure an event to monitor or record data that the user is not authorized to access, a security vulnerability exists.

The designer of the MIB module shall ensure that every object that points to another object within the device is unambiguously associated with appropriate security credentials that are to be used when accessing that data.

### Reuse Existing Structures

---

Developers of NTCIP MIB modules should be familiar existing MIBs published by

- IETF (i.e., as RFCs), especially
  - RFC 2981, event MIB;
  - RFC 2982, expression MIB; and
  - RFC 3014, notification MIB;
  - RFC 3231, scheduling MIB;
  - RFC 3433, entity sensor MIB;
  - RFC 3877, alarm MIB;
  - RFC 4268, entity state MIB;
  - RFC 6933, entity MIB;
- ISO (especially the ISO 20684 series)
- NTCIP

Many times, structures that developers wish to create already have existing MIBs that:

- Can be used directly
- Can be used with minor changes
- Can be used as a basis of a new design, or
- Simply provide insights into issues that should be considered during a new design

In particular, developers should be aware of the general-purpose input/output capability defined in ISO 20684-2 and how it can be easily extended to support a standardized interface with virtually any generic sensor or actuator.

#### Use of Table Indexes

---

This document does not preclude the potential for two or more management applications from simultaneously accessing the same object. However, designers of MIB modules should consider such situations and use designs to minimize conflicts and maximize security. In particular, designers should consider adding an initial owner index to tables when there is a need to prevent different managers from accessing each other's data or configuration. This will allow administrators to configure access to rows within the table by simply specifying the owner identifiers to which they should have access.

Developers should consider using the ISO 20684-7 fdOwnerTable for this purpose. Developers should also be cautious in defining overall management features for the table that might allow users with restricted access to monitor activities by other users. This can be achieved by placing monitoring objects into a table that augments the fdOwnerTable so that each owner can monitor statistics related to their own rows in the table but not universal statistics. Universal statistics, if

defined, should be placed in a separate area of the international object identifier tree so that administrators can easily control access.

#### Consistency with the PRL and RTM

The module compliance statement, as defined by the module compliance table and object refinement table shall be consistent with the Protocol Requirements List (PRL) and Requirements Traceability Matrix (RTM).

#### Multi-Version Interoperability (MVI, Backward Compatibility)

With the development and publication of newer versions of various NTCIP standards, the need to address and ensure multi-version interoperability (MVI, often referred to as “backward compatibility” with a previous version, or interoperability to a defined extent with future versions). To promote MVI, the standard shall ensure that standards provide adequate documentation that defines what features are supported by each version.

#### Row Status in Static Tables

The MIB module for NTCIP 8004 defines the deprecated NtcipRowStatusStatic textual convention; the UML state transition diagrams for this textual convention are provided in Figure 6, Figure 7, and Figure 8.

```
stateDiagram-v2
    invalid --> invalid: set(row data)[error]<br>set(status=activate) / badValue<br>set(status=deactivate)/badValue
    invalid --> standby: set(row data) [no error]
    invalid --> other: set(proprietary rowStatus object = nonstandard value)
    available
    note right of available: There is no path to transition directly between invalid and available, unless the table-specific reset command provides for this.
    note: A device reset shall force the values of all columns to the default value and shall force a transition to the default state. The default value and default state shall be defined by any table using a RowStatus column.
```

Figure 6: Row Status Static—Invalid

```
stateDiagram-v2
    standby --> standby: set(row data)[no error]<br>set(status=deactivate)
    standby --> invalid: set(row data) [error]
    standby --> available: set(status=activate)
    standby --> other: set(proprietary rowStatus object = nonstandard value)
    note: A device reset shall force the values of all columns to the default value and shall force a transition to the default state. The default value and default state shall be defined by any table using a RowStatus column.
```

Figure 7: Row Status Static—Standby

```
stateDiagram-v2
    available --> available: set(row data)[no error]<br>set(status=activate)
    available --> invalid: set(row data) [error]
    available --> standby: set(status=deactivate)
    available --> other: set(proprietary rowStatus object = nonstandard value)
    note: A device reset shall force the values of all columns to the default value and shall force a transition to the default state. The default value and default state shall be defined by any table using a RowStatus column.
```

Figure 8: Row Status Static—Available

.....

1. The same requirement sometimes appears multiple times within and across RFCs. This document eliminates this redundancy and only provides a single sample reference to each requirement. ←
2. Within this context, a module using the same MODULE-IDENTITY represents a new revision of the module. A module that uses a different MODULE-IDENTITY is considered as different module. If a module is intended to be a replacement of all objects of an existing module, it should use a new MODULE-IDENTITY. ←←
3. Required sections for NTCIP data dictionary standards are defined in NTCIP 8002. ←
4. Within NTCIP, waivers shall require approval of the Joint Committee on the NTCIP ←
5. RFC 2580 requires standardized modules to either include a MODULE-COMPLIANCE macro or to be paired with a companion module that provides this definition ←
6. NTCIP standards are allowed to document module compliance with a MODULE-COMPLIANCE macro (either in the same MIB or a different MIB) or through the use of module compliance and supported range tables as defined in X.X.X ←
7. Moving an invocation can cause problems in other modules using the IMPORT statement; however, this requirement does not apply to upgrading modules from SMIv1 to SMIv2 because the module names change. ←
8. RFC 2578 recommends against exceeding 32 characters, but RFC 4181 explicitly revises the statement to allow longer names when necessary to support unique, mnemonic names. ←
9. Private MIBs are encouraged to prefix descriptors with the appropriate enterprise name ←
10. The requirement is intended to only apply to RFC documents, this profile expands its application to NTCIP documents ←
11. This suffix is optional for Version 1 descriptors ←
12. For NTCIP, the contact details shall reflect where user comments are to be sent, as defined in foreword of the document. ←
13. Alternatively, the information can be provided in another normative section of the MIB module, such as the DESCRIPTION clause of a table or of the MODULE-IDENTITY macro. ←
14. RFC 2578 indicates that the DEFVAL "may be used" and that it is a "hint" to implementors. As such, it is informative and can be changed without deprecating an object. Within NTCIP, the DEFVAL clause is recommended within NTCIP and changes to its value should be avoided. ←  
←←
15. Other formats should only be used for backwards compatibility ←
16. Or use the ITSRelativeOid textual convention ←
17. Use a pair of objects based on either 1) InetAddressType and InetAddress or 2) a) either TransportDomain or TransportAddressType and b) TransportAddress ←
18. Except when converting from SMIv1 ←
19. "<EntryType>" represents an ASN.1 SEQUENCE without the DEFAULT, OPTIONAL, or COMPONENTS OF keywords ←
20. As such, if an implementation does not support any rows, it would not support the object that indicates the number of rows ←
21. Unless explicitly prohibited within the \<Definition> of the conceptual row object type ←
22. It is preferred to revise the definition of the existing table if it is managed by the same authority ←←←←

- 23. Example 1: Table B only has rows for some of the rows in Table A. Example 2: Table B has multiple rows for each row in Table A ←
- 24. Except when importing from SMIv1 ←
- 25. Use "read-create" instead ←
- 26. See Clause 5.3 for an example. ←
- 27. Within SNMP, an `"INTEGER"` is identical to an `"Integer32"`; omitting the `"32"` has caused confusion in the past. ←
- 28. The CounterBasedGauge64 breaks some defined semantics of the base type, but is the best current solution for a 64-bit gauge and is generally accepted by the SNMP community ←←
- 29. May use Integer32 to support backward compatibility ←
- 30. `"Based on"` allows for the use of textual conventions that use the OBJECT IDENTIFIER type as its base. ←
- 31. Integer and the various NTCIP `"bitmap"` types shall only be used for backwards compatibility ←
- 32. Within SNMP, an OCTET STRING is not allowed to exceed 65535 octets; this constraint should be added to any OCTET STRING that is currently unconstrained ←
- 33. For example, the range of counters and gauges cannot be changed as doing so would change the meaning of the maximum and minimum values; however, an object can be deprecated and replaced ←
- 34. Implementations are not allowed to extend meaning. For example, if an implementation wishes to logically create new enumerated values, it cannot add enumerations to the standard object type as doing so might conflict with future versions of the standard. Instead, it could create a manufacturer-specific object type and transition the standard object type to a neutral value (e.g., "other") ←
- 35. Values not listed are reserved for future standardization and constitute a protocol error if used ←
- 36. RFC 2578 recommends against exceeding 32 characters, but RFC 4181 explicitly revises the statement to allow longer names when necessary to support unique, mnemonic names. ←
- 37. Changes should only be made when there is significant need as these names often are used within code and changes can create maintenance headaches ←
- 38. This might require defining a local textual convention ←
- 39. Values not listed are reserved for future standardization and constitute a protocol error if used ←
- 40. When adding named bits to an existing BITS construct, bit positions can be skipped to facilitate identification of different versions of the object type by forcing new bits into a new octet. ←
- 41. RFC 2578 recommends against exceeding 32 characters, but RFC 4181 explicitly revises the statement to allow longer names when necessary to support unique, mnemonic names. ←
- 42. Except for existing block objects that cannot be converted into a structure that uses OER ←
- 43. It is the responsibility of the agency deploying an SNMP agent that users are not granted greater access for block objects than they are for the objects contained within the block object. ←



- 44. For example, a ranged integer might indicate that the maximum value has a special meaning. ←
- 45. The clause should always point to the next sequential replacement; if that object has been deprecated, it will reference its replacement. ←
- 46. For example, the functionality is no longer relevant or the design was flawed/ambiguous and has been replaced with a new object type ←
- 47. NTCIP defines additional rules for the use of notifications as defined in NTCIP 2301. ←
- 48. Throttling is important in a number of scenarios, including: the same notification occurring in rapid succession, different notifications occurring in rapid succession, notifications buffering when the device is offline and flooding the network when connected, and limited buffering when offline and having a large number of devices flood the network when connectivity is restored (e.g., after a power outage). The throttling design should protect against all of these. ISO 20684-4 proposes a solution for this that should be considered. ←
- 49. They should be replaced with textual conventions ←
- 50. Because this is optional, an object group can remain \"current\", even if one of the contained object types are \"deprecated\". See RFC 4181 Clause 4.9 for more details. ←
- 51. This allows a reader to quickly identify when features were added ←
- 52. It is expected that the version ←
- 53. Implemented as a MODULE-COMPLIANCE macro. ←
- 54. Implemented as a module compliance table and object refinement table ←
- 55. Encodings that resolve to an OCTET STRING identify a leading and trailing octet, which are then sent to lower layers one octet at a time; integers are encoded as a single unit and lower protocol layers define the ordering of octets ←
- 56. NTCIP 8004 currently does not define any notifications and therefore currently does not define a notifications node; however, node zero is reserved for this purpose. ←
- 57. NTCIP 8004 currently does not have a conformance statement because currently it only defines textual conventions and object identities; however, node 127 is reserved for this purpose. ←

## Requirements for Agent Implementations [Normative]

---

### Agent Capabilities

---

Manufacturers supplying transportation equipment should make AGENT-CAPABILITIES statements available for their equipment. All manufacturers need to obtain a vendor number from either NEMA or IANA prior to creating MIB modules that have manufacturer-specific data.

### Requirements for Non-Standard MIBs

---

All MIB modules shall conform to the basic compilation rules defined for SMIv2 in addition to the requirements defined within this section. Developers of non-standard MIB modules are encouraged to adopt the requirements defined in this document to the extent to which they might apply.

### Default Values

---

The value specified in a DEFVAL clause should be used when initializing a new object instance (e.g., during reboot or row creation), if feasible unless another value is provided. <sup>1</sup>

### Extensions of Standardized Enumerations

---

Unless stated otherwise within an object type <Definition> subclause, a NamedNumber (i.e., an enumerated item in an enumeration) that is assigned an identifier of "other" shall be treated as a read-only value. An attempt to set an object instance to a read-only value shall return an error (wrongValue in the case of SNMPv3).

Unless normative text is added to specifically prohibit the use of the other state, a user- or manufacturer-specific object shall be permitted to define an object specification that extends the possible states. For example, NTCIP 1202:2005 includes the following object:

```
coordCorrectionMode OBJECT-TYPE
SYNTAX      INTEGER { other (1),
                     dwell (2),
                     shortway (3),
                     addOnly (4) }
MAX-ACCESS  read-write
STATUS      mandatory
DESCRIPTION
"<Definition> This object defines the Coord Correction
Mode. The possible modes are:
other: the coordinator establishes a new offset by
a mechanism not defined in this standard.
dwell: when changing offset, the coordinator shall
establish a new offset by dwelling in the coord
phase(s) until the desired offset is reached.
shortway (Smooth): when changing offset, the
coordinator shall establish a new offset by
adding or subtracting to/from the timings in a
```

```

manner that limits the cycle change. This
operation is performed in a device specific
manner.
addOnly: when changing offset, the coordinator
shall establish a new offset by adding to the
timings in a manner that limits the cycle
change. This operation is performed in a device
specific manner.
...

```

To define a new correction mode, something like the following proprietary object could be used:

```

xxxCoordCorrectionModeExt OBJECT-TYPE
SYNTAX      INTEGER { other (1),
                      subOnly (2) }
MAX-ACCESS  read-write
STATUS      mandatory
DESCRIPTION
"<Definition> This object defines an extension to the Coord
Correction Mode as defined in NTCIP 1202. The possible modes are:
other: the coordinator establishes a new offset according to
NTCIP-1202::coordCorrectionMode.
subOnly: when changing offset, the coordinator shall
establish a new offset by subtracting from the timings in such a
manner that limits the cycle change. This operation is performed
in a device specific manner
...

```

In this case, setting `xxxCoordCorrectionModeExt` equal to `subOnly` forces `coordCorrectionMode` equal to `other`. Setting `coordCorrectionMode` equal to `shortway` forces `xxxCoordCorrectionModeExt` equal to `other`.


---

<sup>1</sup>. Clause 7.9 of RFC 2578 indicates that implementing the default value is optional. This clause makes it recommended. ←

## Guidelines for Operating Agencies [Informative]

---

Operating agencies should be aware that there are cases where two or more management applications are potentially able to simultaneously access the same object. For example, the Global Database Management group does not prevent multiple management applications from accessing the data simultaneously. It is the responsibility of any agencies involved in inter-jurisdictional control to define procedures to ensure against this situation.

 June 18, 2025

## Annex A NTCIP Structure of Management Information (SMI) [Normative]

Annex A defines the overall structure of the NTCIP-defined management information and several textual conventions that are believed to be useful for a broad range of applications. The text provided in Annex A.1 constitutes the standard NTCIP8004-NemaV1 MIB. Annex A.2 (except the headings) constitutes the standard NTCIP8004-TransportationV1 MIB.

### A.1 NEMA Module

The NTCIP8004-Transportation module is shown below and is available on [GitHub](#).

```
NTCIP8004-NEMA DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-IDENTITY, enterprises
        FROM SNMPv2-SMI;
        -- RFC 2578

nema MODULE-IDENTITY
    LAST-UPDATED "202212120500Z"
    ORGANIZATION "NTCIP BSP2 WG"
    CONTACT-INFO
        "name:      NTCIP Coordinator
         email:     ntcip@nema.org
         postal:    National Electrical Manufacturers Association
                  1300 North 17th Street, Suite 1752
                  Rosslyn, VA 22209-3801
                  USA"
    DESCRIPTION
        "IANA delegated sub-identifier 1206 under the enterprise node to NEMA
         This MIB defines the overall structure of management information
         that is believed to be useful for a broad range of applications."
    REVISION      "202212120500Z"
    DESCRIPTION
        "Updated to SMIV2. Divided module into a separate NEMA managed parent
         module and an NTCIP managed module. "
    REVISION      "200507190500Z"
    DESCRIPTION   "MIB moved into NTCIP 8004 with updated module name."
    REVISION      "200112010500Z"
    DESCRIPTION   "NEMA TS 3.2 republished as NTCIP 1101 v01."
    REVISION      "199610010500Z"
    DESCRIPTION   "NEMA TS 3.2 approved."
    ::= { enterprises 1206 }

nemaMgmt OBJECT-IDENTITY
    STATUS      current
    DESCRIPTION "The mgmt subtree is used for standard NEMA object definitions
         that span different NEMA sections."
    ::= { nema 1 }

nemaExperimental OBJECT-IDENTITY
    STATUS      current
    DESCRIPTION "The experimental subtree is used to identify object types used
         only on an experimental basis. Changing OBJECT IDENTIFIERS once assigned
         is challenging due to an installed base. Thus, this node should only be
         used for experiments, such as research efforts that are strictly limited
         to be short-term projects to investigate and test ideas. More permanent
         node assignments should be obtained prior to any longer-term or larger-
         scale deployment to prevent complications if, and when, the solution
         becomes more widely adopted."
    ::= { nema 2 }

expGlobal OBJECT-IDENTITY
    STATUS      deprecated
    DESCRIPTION
        "<Definition> A node that contains the experimental auxiliary
         input/output objects.
         <Object Identifier> 1.3.6.1.4.1.1206.2.2"
    ::= {nemaExperimental 2}

nemaPrivate OBJECT-IDENTITY
    STATUS      current
    DESCRIPTION
        "The private subtree is used to identify object types defined unilaterally. NEMA assigns nodes to enterprises for purposes such as defining enterprise-
         specific MIB modules. A request for a node assignment can be sent to the NTCIP Coordinator at ntcip@nema.org. When a sub-node is assigned, the responsibility
         for managing that sub-node is transferred to the enterprise making the request."
```

Upon receiving a node, the enterprise may, for example, define new MIB modules and object types under this node. In addition, it is strongly recommended that the enterprise also register its transportation devices under this subtree, to provide an unambiguous identification mechanism for use in management protocols. For example, if the 'ABC, Inc.' enterprise produced transportation devices, then it could request a node under the nemaPrivate node from NEMA. Such a node might be numbered: 1.3.6.1.4.1.1206.3.99

The 'ABC, Inc.' enterprise might then register their 'Widget Controller' under the name of 1.3.6.1.4.1.1206.3.99.1, ensuring a unique identification. Thereafter, each enterprise is responsible for ensuring unique identification of information objects within their subtree. NEMA delegates the role of assigning numbers under each nemaPrivate node to those to which they are assigned, except of course for the initial enterprise number."

```
 ::= { nema 3 }
```

END -- NTCIP8004-NEMA

## A.2 Transportation Module

The `NTCIP8004-Transportation` module is shown below and is available on [GitHub](#).

```
NTCIP8004-Transportation DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-IDENTITY, Integer32
        FROM SNMPv2-SMI
        -- RFC 2578

    TEXTUAL-CONVENTION
        FROM SNMPv2-TC
        -- RFC 2579

    nema
        FROM NTCIP8004-NEMA;

transportation MODULE-IDENTITY
    LAST-UPDATED "202212120500Z"
    ORGANIZATION "NTCIP BSP2 WG"
    CONTACT-INFO
        "name: NTCIP Coordinator
        email: ntcip@nema.org
        postal: National Electrical Manufacturers Association
        1300 North 17th Street, Suite 1752
        Rosslyn, VA 22209-3801
        USA"
    DESCRIPTION
        "NEMA delegated its sub-identifier 4 to the Joint Committee on the NTCIP
        with its previously assigned descriptor of 'transportation'. This MIB
        defines the overall structure of NTCIP-defined management information
        and textual conventions that are believed to be useful for a broad range
        of applications."
    REVISION "202212120500Z"
    DESCRIPTION
        "Updated to SMIV2. Divided module into a separate NEMA managed parent
        module and an NTCIP managed module. Commented out most sub-identifiers,
        so that they can be assigned by the MODULE-IDENTITY macro within other
        NTCIP standards. Formalized textual conventions."
    REVISION "200711290000Z"
    DESCRIPTION "Added tmdd and ntcipTraps nodes."
    REVISION "200507190500Z"
    DESCRIPTION "MIB moved into NTCIP 8004 with updated module name and added
    nodes for chap, modem, and tmdd."
    REVISION "200112010500Z"
    DESCRIPTION "NEMA TS 3.2 republished as NTCIP 1101 v01."
    REVISION "199610010500Z"
    DESCRIPTION "NEMA TS 3.2 approved."
    ::= { nema 4 }

-- A.2.1 Nodal Structure
protocols OBJECT-IDENTITY
    STATUS current
    DESCRIPTION "This node is the root of a subtree for protocol-related
    management information, such as information related to 1) various layers
    of the protocol stack, 2) profiles that cover several layers, 3) dynamic
    object management, and NTCIP traps."
    ::= { transportation 1 }

devices OBJECT-IDENTITY
    STATUS current
    DESCRIPTION
        "<Definition> This node is the root of a subtree for management
        information for various transportation devices. Management of this node
        is delegated to the NTCIP Base Standards, Protocols, and Profiles (BSP2)
        WG.
        <Informative> The following assignments have been made under this node
        as formally declared (or to be declared upon upgrade to SMIV2) in the
        indicated standard:
        Node descriptor Defined in
        1 asc NTCIP 1202
        2 ramp NTCIP 1207
        3 dms NTCIP 1203
        4 tss NTCIP 1209
        5 ess NTCIP 1204
```

```

6      global      NTCIP 1201
7      cctv         NTCIP 1205
8      cctvSwitch   NTCIP 1208
9      dcm          NTCIP 1206
10     ssm          NTCIP 1210
11     scp          NTCIP 1211
12     networkCamera <reserved>
13     elms         NTCIP 1213
17     saeNtcip     NTCIP 1202
18     rsu          NTCIP 1218
19     globalV2     NTCIP 1201"
::= { transportation 2 }

tcip      OBJECT-IDENTITY
STATUS    current
DESCRIPTION "This node has been assigned to the Transit Communications
Interface Profiles Technical Working Group. Assignment of any nodes under
this subtree is delegated to that group."
::= { transportation 3 }

tmdd      OBJECT-IDENTITY
STATUS    current
DESCRIPTION "This node has been assigned to the ITE Traffic Management Data
Dictionary Steering Committee. Assignment of any nodes under this subtree
is delegated to that group."
::= { transportation 4 }

deviceAdmin OBJECT-IDENTITY
STATUS      current
DESCRIPTION "This node has been assigned for the definition of
administrative data that is related to the other other nodes under the
transportation node. Its sub-node structure is intended to parallel that
of the transporation node."
::= { devices 126 }

-- *****
-- Protocols branch of tree
-- *****
layers      OBJECT-IDENTITY
STATUS      current
DESCRIPTION
"<Definition> This node contains management information related to
various layers of the OSI protocol stack.
<Informative> The following subnodes have
been defined:
Node   Descriptor
1      chap
2      modem
7      application"
::= { protocols 1 }

application OBJECT-IDENTITY
STATUS      current
DESCRIPTION "This node contains management information related to
protocols assigned to the application layer of the OSI stack."
::= { layers 7 }

profiles    OBJECT-IDENTITY
STATUS      current
DESCRIPTION "This node contains management information related to profiles
that cover several layers of the OSI stack."
::= { protocols 2 }

ntcipSmi    OBJECT-IDENTITY
STATUS      current
DESCRIPTION "A node used to manage information related to the NTCIP SMI."
::= { protocols 5 }

-- A.2.2 Common Textual Conventions
-- MIB developers should use textual conventions to the extent possible so
-- that 1) associated semantics can be automated and code easily reused, and
-- 2) generic SNMP implementations are guided on how to present values to
-- users. The following lists/defines several textual conventions that are
-- most widely used. The list also includes some of the APPLICATION types
-- defined in RFC 2578 for a more complete reference set.
-- MIB developers are also encouraged to consider the textual conventions
-- listed at https://trac.ietf.org/trac/ops/wiki/mib-common-tcs, which is
-- maintained with all IETF defined conventions.
-- *****
-- Integers
-- *****
-- "Byte", "Ubyte", "Short", "Ushort", and "Long" were defined in
-- NTCIP8004v02, but have identical ranges to the following defined textual
-- conventions defined in ISO 20684-1, which should be used instead
-- From FIELD-DEVICE-TC-MIB
-- @ https://standards.iso.org/iso/20684/-1/ed-2/en/iso20684-1-tc.mib
-- ITSInteger8:      Replaces NTCIP8004v02 "Byte"
-- ITSInteger16:     Replaces NTCIP8004v02 "Short"
-- ITSPositive8
-- ITSPositive16
-- ITSPositive32
-- ITSUnsigned8:     Replaces NTCIP8004v02 "Ubyte"
-- ITSUnsigned16:    Replaces NTCIP8004v02 "Ushort"
```

```

-- From SNMPv2-SMI
-- @ RFC 2578
--   Integer32:      Replaces NTCIP8004v02 "Long" and "INTEGER", both
--                   of which have identical ranges
--   Unsigned32
NtcipPercent ::= TEXTUAL-CONVENTION
  DISPLAY-HINT "d"
  STATUS      deprecated
  DESCRIPTION
    "<Definition> An expression of a percentage from 0 to 100 contained in
    an Integer32 type.
    <Informative> This textual convention is defined to support values
    converted from the SNMPv1 INTEGER type; new objects should use
    ITSPercent as defined in ISO 20684-1, which is based on Unsigned32."
  SYNTAX      Integer32 (0..100)

-- *****
-- Counters and Gauges
-- *****
-- From FIELD-DEVICE-TC-MIB
-- @ https://standards.iso.org/iso/20684/-1/ed-2/en/iso20684-1-tc.mib
--   ITSCounter8:      8-bit, zero-based, resetable
--   ITSCounter16:     16-bit, zero-based, resetable
--   ITSCounter32:     32-bit, zero-based, resetable
--   ITSCounter64:     64-bit, zero-based, resetable
--   ITSGauge8:        8-bit gauge
--   ITSGauge16:       16-bit gauge
-- From SNMPv2-SMI
-- @ RFC 2578
--   Counter32         32-bit, random base, non-resetable
--   Counter64         64-bit, random base, non-resetable
--   Gauge32           32-bit gauge
-- From RMON2-MIB
-- @ RFC 2021
--   ZeroBasedCounter32 32-bit, zero-based, non-resetable
-- From HCNUM-TC
-- @ RFC 2856
--   ZeroBasedCounter64 64-bit, zero-based, non-resetable
--   CounterBasedGauge64 64-bit gauge
-- *****
-- Date/Time
-- *****
-- From FIELD-DEVICE-TC-MIB
-- @ https://standards.iso.org/iso/20684/-1/ed-2/en/iso20684-1-tc.mib
--   ITSDailyTimeStamp
--   ITSDateStamp
--   ITSDayOfMonth
--   ITSDayOfWeek
--   ITSMonth
-- *****
-- Other Enumerations
-- *****
-- From SNMPv2-TC
-- @ RFC 2579
--   TruthValue
--   TestAndIncr
--   RowStatus
--   StorageType
-- From FIELD-DEVICE-TC-MIB
-- @ https://standards.iso.org/iso/20684/-1/ed-2/en/iso20684-1-tc.mib
--   ITSPduErrorStatus
--   ITSUnits

NtcipRowStatusStatic ::= TEXTUAL-CONVENTION
  STATUS      deprecated
  DESCRIPTION
    "<Definition>RowStatusStatic has four states and two commands associated
    with it.
    <Format> The four possible states are defined as:
    1) other: The status is controlled or defined by a user- or
      manufacturer-specific object.
    2) standby: All columnar data in the row have passed (any) defined
      consistency checks but are not to be used by the end application.
    3) available: All columnar data in the row have passed (any) defined
      consistency checks and are available to be used by the end
      application.
    4) invalid: One or more columnar objects has a value that caused the
      row to fail at least one defined consistency check.

    Setting RowStatusStatic equal to one of the above values shall return
    an error (e.g., 'wrongValue' in SNMPv3).

    The only two possible command values that may be set by a management
    application are:
    5) activate: Makes the columnar data in the row available for use by
      the end application.
    6) deactivate: Makes the columnar data in the row unavailable for use
      by the end application.

    <Informative> In a static table, all rows exist irrespective of whether
    the columnar objects contain appropriate values. The value of a
    columnar object within a row may be inappropriate when the values of
    other columnar objects in the row are considered. An entire row itself

```



may also be considered inappropriate under some circumstances. If this is the case, a static table shall include an additional columnar object that defines row status and has the SYNTAX of RowStatusStatic.

UML state transition diagrams for RowStatusStatic are defined in NTCIP 8004.

RowStatusStatic was deprecated in NTCIP8004v03.01 in favor of RowStatus, which allows implementations to create rows when needed and to destroy them when not needed. This allows the potential for better memory management in implementations that wish to provide this capability."

```
SYNTAX      INTEGER          { other (1),          standby (2),          available (3),
                             invalid (4),
                             activate (5),
                             deactivate (6)}
```

```
-- *****
-- Bitmaps
-- *****
-- New objects should use "BITS", unless 1) there is a need to conserve the
-- extra bits and 2) there is confidence that the object will never need to
-- expand to a larger value.
```

NtcipOctetBitmap8 ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1x"

STATUS current

DESCRIPTION

"<Definition> A bitmap of up to 8 bits with Bit 0 in the low-order bit.

When a bit is on (1), the indicated feature is on or supported; when a bit is off (0), the indicated feature is off or not supported.

<Informative> This object orders the bits in the reverse order of the BITS construct but in the same order as used by INTEGERS."

```
SYNTAX      OCTET STRING (SIZE (1))
```

NtcipOctetBitmap16 ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1x "

STATUS current

DESCRIPTION

"<Definition>A bitmap of up to 16 bits with Bit 0 in the low-order bit of the trailing octet and Bit 15 in the high-order bit of the first octet.

When a bit is on (1), the indicated feature is on or supported; when a bit is off (0), the indicated feature is off or not supported.

<Informative> This object orders the bits in the reverse order of the BITS construct but in the same order as used by INTEGERS."

```
SYNTAX      OCTET STRING (SIZE (2))
```

NtcipOctetBitmap32 ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1x "

STATUS deprecated

DESCRIPTION

"<Definition>A bitmap of up to 32 bits with Bit 0 in the low-order bit of the trailing octet and Bit 31 in the high-order bit of the first octet.

When a bit is on (1), the indicated feature is on or supported; when a bit is off (0), the indicated feature is off or not supported.

<Informative>This has been deprecated in favor of the SNMP-standard BITS construct, which orders the bits in the reverse order and allows for growth in a bit field."

```
SYNTAX      OCTET STRING (SIZE (4))
```

NtcipOctetBitmap512 ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1x "

STATUS deprecated

DESCRIPTION

"<Definition>A bitmap of up to 512 bits (64 octets) with the meaning of bits defined by the object type but in no case shall there be more than seven pad bits (i.e., the length of the octet string may be less than 64 octets).

When a bit is set (1), the indicated feature/error is on, active, supported (as defined by the object using this textual convention); when a bit is not set (0), the indicated feature/error is off, not active, or not supported.

<Informative>This has been deprecated in favor of the SNMP-standard BITS construct, which orders the bits in the reverse order and allows for growth in a bit field."

```
SYNTAX      OCTET STRING (SIZE (0..64))
```

NtcipOctetBitmap2040 ::= TEXTUAL-CONVENTION

DISPLAY-HINT "1x "

STATUS deprecated

DESCRIPTION

"<Definition>A bitmap of up to 2040 bits (255 octets) with the meaning of bits defined by the object type but in no case shall there be more than seven pad bits (i.e., the length of the octet string may be less than 255 octets).

When a bit is set (1), the indicated feature/error is on, active, supported (as defined by the object using this textual convention); when a bit is not set (0), the indicated feature/error is off, not active, or not supported.

<Informative>This has been deprecated in favor of the SNMP-standard BITS

```

        construct, which orders the bits in the reverse order and allows for
        growth in a bit field."
SYNTAX      OCTET STRING (SIZE (0..255))

NtcipOctetBitmap3200 ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1x "
STATUS      deprecated
DESCRIPTION
"<Definition>A bitmap of up to 3200 bits (400 octets) with the meaning of
bits defined by the object type but in no case shall there be more than
seven pad bits (i.e., the length of the octet string may be less than
400 octets).

When a bit is set (1), the indicated feature/error is on, active,
supported (as defined by the object using this textual convention);
when a bit is not set (0), the indicated feature/error is off, not
active, or not supported.

<Informative>This has been deprecated in favor of the SNMP-standard BITS
construct, which orders the bits in the reverse order and allows for
growth in a bit field."
SYNTAX      OCTET STRING (SIZE (1..400))

NtcipIntegerBitmap8 ::= TEXTUAL-CONVENTION
DISPLAY-HINT "x"
STATUS      current
DESCRIPTION
"<Definition> A bitmap of up to 8 bits with Bit 0 in the low-order bit.
<Informative> This object orders the bits in the reverse order of the
BITS construct but in the same order as used by INTEGERS."
SYNTAX      Integer32 (0..255)

NtcipIntegerBitmap16 ::= TEXTUAL-CONVENTION
DISPLAY-HINT "x"
STATUS      current
DESCRIPTION
"<Definition>A bitmap of up to 16 bits with Bit 0 in the low-order bit of
the trailing octet.
<Informative> This object orders the bits in the reverse order of the
BITS construct but in the same order as used by INTEGERS."
SYNTAX      Integer32 (0..65535)

NtcipIntegerBitmap31 ::= TEXTUAL-CONVENTION
DISPLAY-HINT "x"
STATUS      deprecated
DESCRIPTION
"<Definition>A bitmap of up to 31 bits with Bit 0 in the low-order bit of
the trailing octet.
<Informative>This has been deprecated in favor of the SNMP-standard BITS
construct, which orders the bits in the reverse order and allows for
growth in a bit field. Prior versions of NTCIP sometimes indicated a
value for Bit 31 (i.e., the high-order bit) with an unsigned range on
INTEGER; this was a technically invalid definition and resulted in
ambiguities as to whether it should be encoded in 4 octets (which is
technically a signed value, or 5 octets, which many off-the-shelf SNMP
engines do not support. As a result, implementation response to a value
that uses Bit 31 is undefined and should be avoided."
SYNTAX      Integer32

-- *****
-- Text
-- *****
-- From SNMP-FRAMEWORK-MIB
-- @ RFC 3411
--      SnmpAdminString: UTF8 String 0..255 octets

NtcipOwnerString ::= TEXTUAL-CONVENTION
DISPLAY-HINT "127t"
STATUS      deprecated
DESCRIPTION
"<Definition>This data type is used to model an administratively assigned
name of the owner of a resource, preferably in human-readable form.
The string is encoded in UTF-8. It is suggested that this name contain
one or more of the following: management station name, manager's name,
location or phone number.
<Informative> Versions prior to 2023 limited this textual convention to
the NVT ASCII character set; however, the range was expanded to allow
for more language support and UTF-8 is fully compatible with NVT
ASCII. This textual convention has been deprecated as there is seldom a
need to force a limit of 127 characters (as this convention does)
rather than 255 characters (as SnmpAdminString does). "
SYNTAX      OCTET STRING (SIZE (0..127))

-- *****
-- Other OCTET STRING conventions
-- *****
-- From FIELD-DEVICE-TC-MIB
-- @ https://standards.iso.org/iso/20684/-1/ed-1/en/iso20684-1-1-a.mib
--      ITS0erString

NtcipTwoColors ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1dR1dG1dB1dR1dG1dB"
STATUS      current
DESCRIPTION
"A 6-octet value representing two colors, each encoded as a 3-octet RGB

```

```
value."  
SYNTAX OCTET STRING (SIZE(6))  
  
-- *****  
-- OBJECT IDENTIFIER conventions  
-- *****  
-- From SNMPv2-TC  
-- @ RFC 2579  
--     AutonomousType  
--     VariablePointer  
--     RowPointer  
  
END -- NTCIP8004-Transportation
```

 June 19, 2025




## Annex B Conversion From SMIv1 To SMIv2 [Normative]

---

The following defines additional requirements notes that apply to NTCIP modules being converted from SMIv1 to SMIv2.

- <sup>1</sup> The value assigned in the SMIv2 `MAX-ACCESS` clause shall be consistent with the previous SMIv1 `ACCESS` clause (i.e., it must provide greater or equal access)
- <sup>2</sup> The value assigned in the `MAX-ACCESS` clause shall be `read-create`, if the previous `ACCESS` clause indicated `read-write` and the table supports row creation
- <sup>3</sup> The `STATUS` clause shall be updated to reflect the SMIv2 validity of the object type rather than the SMIv1 optionality of the object type (e.g., change `mandatory` and `optional` to `current`)
- <sup>4</sup> The `<Units>` subclause from the NTCIP SMIv1 conventions shall be converted to the SMIv2 `UNITS` clause
- <sup>5</sup> Each `TRAP-TYPE` macro shall be converted to a `NOTIFICATION-TYPE` macro per the rules of RFC 3584 Clause 2.1.2
- <sup>6</sup> A `NOTIFICATION-TYPE` that is being converted from an SMIv1 `TRAP-TYPE` shall be assigned an object identifier that equates to the `TRAP-TYPE`'s `ENTERPRISE` clause followed by a zero and followed by the value assigned to the SMIv1 `TRAP-TYPE`.
- <sup>7</sup> Deprecating object types on the basis of converting to IETF-recognized textual conventions (e.g., `INTEGER` to `Gauge32`) is encouraged when the textual conventions provide significant additional semantics that off-the-shelf tools can benefit from
- <sup>8</sup> Converting from `Counter` to `Counter32` requires ensuring that the semantics defined in RFC 2578 Clause 7.1.6 are satisfied
- <sup>9</sup> The `SYNTAX` of an object type shall not be changed in a way that modifies the data type when encoded using BER or OER. When necessary, the existing object can be deprecated and replaced with a new object. The following conversions (among others) are prohibited based on this rule:
  - a. `INTEGER` to `Unsigned32`
  - b. `INTEGER` to `Counter32`
  - c. `INTEGER` to `Gauge32`
  - d. `Counter` to `ZeroBasedCounter32`
  - e. `Counter` to `Counter64`
  - f. `Counter` to `ZeroBasedCounter64`
- <sup>10</sup> The counter-based timekeeping solution used by `globalTime` is not conformant with SNMPv3 and shall be deprecated. When time-related object types need to be defined (e.g., a timestamp), they should be based on the textual conventions defined in ISO 20684-1.

- <sup>11</sup>. Developers should consider migrating object types that represent general sensor data to the general-purpose input/output feature of ISO 20684-2 if and when any backwards compatibility issues arise.

 June 18, 2025

## Annex C Macro Examples [Informative]

The following examples provide various examples of properly formatted invocations of the macros discussed by this document.

NOTE---While the macros shown are similar to real invocations, the details are frequently shortened to save space and the macros often represent translations of SMIv1 entities for which the indicated SMIv2 translations have not yet been approved. The examples within this annex are strictly informative and provided as examples, they should not be interpreted as standardized information. To prevent any confusion, the formal OIDs in this section are shown with a parent node of \"example\".

### C.1 Module Identity Macro

```
transportation MODULE-IDENTITY
LAST-UPDATED \"202212120500Z\"
ORGANIZATION \"NTCIP BSP2 WG\"
CONTACT-INFO
  \"name: NTCIP Coordinator
   email: ntcip@nema.org
   postal: National Electrical Manufacturers Association
           1300 North 17th Street, Suite 1752
           Rosslyn, VA 22209-3801
           USA\"
DESCRIPTION
  \"NEMA delegated its sub-identifier 4 to the Joint Committee on the NTCIP
  with its previously assigned descriptor of \"transportation\". This MIB
  defines the overall structure of NTCIP-defined management information
  and textual conventions that are believed to be useful for a broad range
  of applications.\"
REVISION \"202212120500Z\"
DESCRIPTION
  \"Updated to SMIv2. Divided module into a separate NEMA managed parent
  module and an NTCIP managed module. Commented out most sub-identifiers,
  so that they can be assigned by the MODULE-IDENTITY macro within other
  NTCIP standards. Formalized textual conventions.\"
REVISION \"200711290000Z\"
DESCRIPTION \"Added tmdd and ntcipTraps nodes.\"
REVISION \"200507190500Z\"
DESCRIPTION
  \"MIB moved into NTCIP 8004 with updated module name and added
  nodes for chap, modem, and tmdd.\"
REVISION \"200112010500Z\"
DESCRIPTION \"NEMA TS 3.2 republished as NTCIP 1101 v01.\"
REVISION \"199610010500Z\"
DESCRIPTION \"NEMA TS 3.2 approved.\"
:= { example 1 }
```

### C.2 Object Identity Macro

```
protocols OBJECT-IDENTITY
STATUS current
DESCRIPTION \"This node is the root of a subtree for protocol-related
management information, such as information related to 1) various layers
of the protocol stack, 2) profiles that cover several layers, 3) dynamic
object management, and NTCIP traps.\"
:= { example 2 }
```

### C.3 Conceptual Table Object Type

```
phaseTable OBJECT-TYPE
SYNTAX SEQUENCE OF PhaseEntry
MAX-ACCESS not-accessible
STATUS current
```



```

DESCRIPTION
  "<Definition> A table containing Controller Unit phase parameters.
  The number of rows in this table is equal to the maxPhases object.
  <TableType> static
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.x.1.2"
::= { example 3 }

```

## C.4 Conceptual Row Object Type

```

phaseEntry OBJECT-TYPE
  SYNTAX PhaseEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    "<Definition> Parameters for a specific Controller Unit phase.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.x.1.2.1"
  INDEX { phaseNumber }
::= { example 4 }

PhaseEntry ::= SEQUENCE {
  phaseNumber Integer32,
  phaseWalk Integer32,
  phasePedestrianClear Integer32,
  phaseMinimumGreen Integer32,
  phaseYellowChange Integer32,
  phaseRedClear Integer32 }

```

## C.5 Row Status Object Type

```

vacmAccessStatus OBJECT-TYPE
  SYNTAX RowStatus
  MAX-ACCESS read-create
  STATUS current
  DESCRIPTION
    "<Definition> The status of this conceptual row.
    The RowStatus TC [RFC2579] requires that this
    DESCRIPTION clause states under which circumstances
    other objects in this row can be modified:
    The value of this object has no effect on whether
    other objects in this conceptual row can be modified.
    <Object Identifier> 1.3.6.1.6.3.16.1.4.1.9"
::= { example 5 }

```

## C.6 Enumeration

```

dmsMemoryMgmt OBJECT-TYPE
  SYNTAX INTEGER {
    other (1), -- deprecated
    normal (2),
    clearChangeableMessages (3),
    clearVolatileMessages (4) }
  MAX-ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "<Definition> Allows the system to manage the device's memory. SNMP
    Get operations on this object should always return normal (2).

    clearChangeableMessages (3): the controller shall set dmsMessageStatus
    for all changeable messages to notUsed (1), and release all memory
    associated with changeable messages. This action does not affect any
    changeable graphics or fonts.

    clearVolatileMessages (4): the controller shall set dmsMessageStatus for
    all volatile messages to notUsed (1), and release all memory associated
    with volatile messages. This action does not affect any changeable
    graphics or fonts.

    <Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.16"
  DEFVAL {normal}
::= { example 6 }

```

## C.7 Integer32

```

maxPhases OBJECT-TYPE
  SYNTAX Integer32 (2..255)
  UNITS "phase"
  MAX-ACCESS read-only
  STATUS current

```

```

DESCRIPTION
  "<Definition> The Maximum Number of Phases this Controller Unit
    supports. This object indicates the maximum rows which shall appear in
    the phaseTable object.
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.1.1.1"
::= { example 7 }

```

## C.8 Gauge32

```

dcmBatteryVoltage OBJECT-TYPE
  SYNTAX Gauge32 (0..255)
  UNITS "tenths of volts"
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    "<Definition> Indicates the voltage of the battery at the time of the
      request. The value range shall be 00.0-25.5V in 0.1 increments.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.9.2.2"
::= { example 8 }

```

## C.9 Counter32

```

dcmVehicleSeqNum OBJECT-TYPE
  SYNTAX Counter32 (0..65535)
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    "<Definition> A number assigned to each vehicle that traverses a
      sensor array. The numbers are assigned sequentially over all sensor
      arrays. The starting value is not defined and the value shall roll over
      to a value of 0 after reaching the maximum. This number along with a
      time tag (controller-localTime) is used to uniquely identify vehicles.
    <Object Identifier> 1.3.6.1.4.1.1206.4.9.1.26"
::= { example 9 }

```

## C.10 Unsigned32

```

dmsSignWidth OBJECT-TYPE
  SYNTAX Unsigned32 (0..65535)
  UNITS "millimeters"
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    "<Definition> Indicates the sign width in millimeters including the
      border (dmsHorizontalBorder).
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.4"
::= { example 10 }

```

## C.11 Object Identifier

```

dcmVCOID OBJECT-TYPE
  SYNTAX InstancePointer
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    "<Definition> This field contains the OID of a Vehicle Criteria
      object that is supported by this device. The OID specified must be a
      full OID from the root of the iso tree, relative OID's cannot be used.
      This allows referring to objects outside of this MIB. The specified OID
      can also be an OID from a table (if the object is only defined within a
      table (e.g. Array Number), but it shall be the OID of the object from
      the table in which the specified object is defined (e.g. if Array Number
      is to be used, the OID shall be taken from the Logical IO To Array Map
      Table.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.3.1.4"
::= { example 11 }

```

## C.12 BITS

```

systemCameraEquipped OBJECT-TYPE
  SYNTAX BITS { cameraPower (0),
                heaterPower (1),
                wiper (2),

```

```

        washer (3),
        blower (4) }
MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "<Definition> A bit mapped value as defined below:
    Bit When set, denotes the availability of a controllable:
        0 Camera Power supply,
        1 Heater Power supply,
        2 Wiper,
        3 Washer,
        4 Blower
    All other bits reserved.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.7.5.3"
::= { example 12 }

```

## C.13 Internet Address

```

intersectionAddressType OBJECT-TYPE
SYNTAX InetAddressType
ACCESS read-write
STATUS current
DESCRIPTION
    "<Definition> This object identifies the type of address stored in
    intersectionAddress.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.10.2.2.1.13"
DEFVAL { ipv4 }
::= { example 13 }

intersectionAddress OBJECT-TYPE
SYNTAX InetAddress
ACCESS read-write
STATUS current
DESCRIPTION
    "<Definition> This object provides the Internet address for the entry
    and shall be a unicast address.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.10.2.2.1.2"
DEFVAL { '00000000'h }
::= { example 13 }

```

## C.14 Text

```

fontName OBJECT-TYPE
SYNTAX SnmpAdminString (SIZE (0..64))
MAX-ACCESS read-write
STATUS current
DESCRIPTION
    "<Definition> Indicates the name of the font.
    <Object Identifier> 1.3.6.1.4.1.1206.4.2.3.3.2.1.3"
::= { example 14 }

```

## C.15 Textual Convention Macro

```

MessageActivationCode ::= TEXTUAL-CONVENTION
DISPLAY-HINT "\"2dm1dp1dt2dn2xc1d.1d.1d.1d\""
STATUS current
DESCRIPTION
    "The MessageActivationCode consists of those parameters required to
    activate a message on a DMS. It is defined as an OCTET STRING containing
    the OER-encoding of the following ASN.1 structure:

    MessageActivationCodeStructure ::= SEQUENCE {
        duration INTEGER (0..65535),
        activatePriority INTEGER (0..255),
        messageMemoryType INTEGER (0..255),
        messageNumber INTEGER (0..65535),
        messageCRC OCTET STRING (SIZE (2)),
        sourceAddress OCTET STRING (SIZE (4))
    }
    where,
    duration = the maximum amount of time, in minutes, the message may be
    displayed prior to activating the dmsDefaultEndDurationMessage.
    dmsMessageTimeRemaining shall be set to this value upon successful
    display of the indicated message. A value of 65535 shall indicate an
    infinite duration.

    activatePriority = the activation priority of the message. If this value
    is greater than or equal to the dmsMessageRunTimePriority of the
    currently displayed message, the new message shall be displayed unless
    errors are detected.

    messageMemoryType = the dmsMessageMemoryType of the desired message.

```

```

messageNumber = the dmsMessageNumber of the desired message.
messageCRC = the dmsMessageCRC of the desired message.
sourceAddress = the 4-byte IPv4 address of the device that requested the
                activation.

For example, given the MULTI string '[jp3]TEST
\[f1\]Flashing[/f1\]', stored in volatile memory slot 5 with no
beacons and no pixel service, the message ID Code is '04 00 05 95 F9'.
If this message is to be displayed for 267 minutes with activation
priority 55 from IP address 103.8.9.10, the message Activation Code is
'01 0B 37 04 00 05 95 F9 67 08 09 0A' in hex and would be displayed
to a user as the following based on the DISPLAY-HINT:
'267m55p4t5n95F9c103.8.9.10'."
SYNTAX OCTET STRING (SIZE 12))

```

## C.16 Block Object Using Identifier and Type

As defined in Section 4.5.4, the definition of a Block Object consists of two parts:

1. The `OBJECT-TYPE` macro used for all object types and
2. A definition of the data structure using a slightly modified version of X.680 ASN.1, as defined in Clause 4.5.4.2; this definition can be placed in a `TEXTUAL-CONVENTION`, in the `<Definition>` subclause of the `OBJECT-TYPE` macro, or in a location external to the MIB module and referenced.

The following uses the textual convention defined in Clause C.15.

```

dmsActivateMessage OBJECT-TYPE
    SYNTAX MessageActivationCode
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "<Definition> A code indicating the active message. The value of this
        object may be SET by a management station or modified by logic internal
        to the DMS (e.g., activation of the end duration message, etc.).

        When modified by internal logic with a reference to a message ID code,
        the duration indicates 65535 (infinite), the activate priority indicates
        255, and the source address indicates an address of 127.0.0.1.

        If a GET is performed on this object, the DMS shall respond with the
        value for the last message that was successfully activated.

        <Object Identifier> 1.3.6.1.4.1.1206.4.2.3.6.3"
    ::= { example 16 }

```

## C.17 Block Object Using Value Reference and a Defined Value

In the following example, because the fields within the ASN.1 structure are optional, a more meaningful value of `DISPLAY-HINT` cannot be provided. As a result, the object type macro references the `ITSOerString`.

The reference to `"essNtcipCategory.0"` is an example of a valuereference that uses a number as an index. `essTemperatureSensorHeight.x"` is an example of a valuereference that uses a `DefinedValue` as an index.

```

essStationMetaDataBlock OBJECT-TYPE
    SYNTAX ITSoerString

```

```

MAX-ACCESS read-only
STATUS current
DESCRIPTION
  "<Definition> The OER-encoding of the following ASN.1 structure:
    MessageActivationCodeStructure ::= SEQUENCE {
      essNtcipCategory.0 OPTIONAL, -- @NTCIP1204-v03
      essTypeOfStation.0 OPTIONAL, -- @NTCIP1204-v03
      essLatitude.0 OPTIONAL, -- @NTCIP1204-v03
      essLongitude.0 OPTIONAL, -- @NTCIP1204-v03
      tempMetaData SEQUENCE OF TemperatureMetaData OPTIONAL
    }

    TemperatureMetaData ::= SEQUENCE {
      essTemperatureSensorIndex.x OPTIONAL, -- @NTCIP1204-v03
      essTemperatureSensorHeight.x OPTIONAL -- @NTCIP1204-v03
    }

  where,
  x = the essTemperatureSensorIndex for the value being reported
  For example, the following left-hand hexadecimal code would be decoded
  as follows:

  17          All optional fields other than essTypeOfStation are present
  02          Category = '\permanent\'
  02 50 AA 26 Latitude = 38.840870 degrees
  F9 BD 2E AD Longitude = -105.042259 degrees
  02          Count for SEQUENCE OF = 2
  03          Both optional fields are present
  01          Index = 1
  01          Height for index 1 = 1 meter
  03          Both optional fields are present
  02          Index = 2
  05          Height for index 2 = 5 meters
  <Object Identifier> 1.3.6.1.4.1.1206.4.2.5.2.15.4"
  ::= { example 17 }

```

## C.18 Block Object Using Value Dereferencing

In the following example, the fields within the ASN.1 structure are not known at design time. Since a more meaningful value of DISPLAY-HINT cannot be provided, the object type macro references the ITSOerString.

The reference to `essNtcipCategory.0` is an example of a `valuereference` that uses a number as an index. `essTemperatureSensorHeight.x` is an example of a `valuereference` that uses a `DefinedValue` as an index.

```

fdObjectGroupCurrentValue OBJECT-TYPE
SYNTAX ITSObjectString
MAX-ACCESS read-only
STATUS current
DESCRIPTION
  "The OER encoded string of the following ASN.1 structure using the
  refinements defined in Clause 4.5.4.2 of NTCIP 8004:
  SEQUENCE {
    *fdObjectGroupFieldObject.x.y.* -- @OBJECT-GROUP-MIB
  }
  where x and y represent the fdObjectGroupOwner and fdObjectGroupName of
  the requested fdObjectGroupCurrentValue.

  If an error occurs in retrieving any value, fdObjectGroupLastError shall
  be updated to reflect the reported error and the value of this object
  (fdObjectGroupCurrentValue) shall be a zero-length string. "
  REFERENCE "NTCIP 1103 watchBlockValue"
  ::= { example 18 }

```

## C.19 Notification Type Macro

```

sampleNotification NOTIFICATION-TYPE
OBJECTS { sampleObject }
STATUS current
DESCRIPTION

```

```
"A sample notification."
::= { example 19 }
```

## C.20 Object Group Macro

---

```
essCharacteristicsGroup OBJECT-GROUP
  OBJECTS { essNtcipCategory,
            essNtcipSiteDescription,
            essTypeOfStation,
            essLatitude,
            essLongitude,
            essReferenceHeight }
  STATUS current
  DESCRIPTION
    "Management information that characterizes the ESS."
  ::= { example 20 }
```

## C.21 Notification Group Macro

---

```
sampleNotificationGroup NOTIFICATION-GROUP
  NOTIFICATIONS { sampleNotification }
  STATUS current
  DESCRIPTION
    "Notifications included in this document."
  ::= { example 21 }
```

## C.22 Module Compliance Macro

---

sampleCompliance MODULE-COMPLIANCE STATUS current DESCRIPTION "The conformance statement for this sample" MODULE -- this module MANDATORY-GROUPS { essCharacteristicsGroup, sampleNotificationGroup } ::= {example 22}

## Annex D History of Changes [Informative]


---

Annex D summarizes the changes made between the prior version of NTCIP 8004 and NTCIP 8004 v03.

The migration of NTCIP from SNMPv1 to SNMPv3 required NTCIP 8004 to migrate from SMIv1 to SMIv2, which required fundamental changes in the way that NTCIP documents its object types. While the scope of NTCIP 8004 v03 is consistent with the scope of prior versions of NTCIP 8004, the details and references have changed considerably. The magnitude of this change resulted in a complete rewrite of the document, including:

1. Updating references
2. Rewriting Section 2 to be consistent with SNMPv3 message structure and terminology
3. Rewriting Section 3 to be consistent with the concepts and terminology contained in ISO/IEC 9834-1 and to update the node assignments under NEMA
4. Rewriting Section 4 to be a profile of the SMIv2 and related IETF RFCs
5. Restructuring the document to consolidate all requirements for implementations into Section 5
6. Restructuring the document to consolidate guidelines for operating agencies into Section 6
7. Revising Annex A to conform with SMIv2 and deprecate items that are no longer recommended
8. Adding Annex B to define special rules from converting from SMIv1 to SMIv2
9. Updating and consolidating all examples into Annex C
10. Updating Annex D.

## §

 June 18, 2025